Gesture Recognition

Sergio Escalera, Isabelle Guyon, and Vassilis Athitsos (Eds.)

Springer Series on Challenges in Machine Learning

2017

Table of Contents

JMLR: Workshop and Conference Proceedings 13:3297–3321, 2012 Published 11/12

Human Gesture Recognition on Product Manifolds

Yui Man Lui LUI@CS.COLOSTATE.EDU
Department of Computer Science
Colorado State University
Fort Collins, CO 80523-1873, USA

Editor: Isabelle Guyon and Vassilis Athitsos

Abstract

Action videos are multidimensional data and can be naturally represented as data tensors. While tensor computing is widely used in computer vision, the geometry of tensor space is often ignored. The aim of this paper is to demonstrate the importance of the intrinsic geometry of tensor space which yields a very discriminating structure for action recognition. We characterize data tensors as points on a product manifold and model it statistically using least squares regression. To this aim, we factorize a data tensor relating to each order of the tensor using Higher Order Singular Value Decomposition (HOSVD) and then impose each factorized element on a Grassmann manifold. Furthermore, we account for underlying geometry on manifolds and formulate least squares regression as a composite function. This gives a natural extension from Euclidean space to manifolds. Consequently, classification is performed using geodesic distance on a product manifold where each factor manifold is Grassmannian. Our method exploits appearance and motion without explicitly modeling the shapes and dynamics. We assess the proposed method using three gesture databases, namely the Cambridge hand-gesture, the UMD Keck body-gesture, and the CHALEARN gesture challenge data sets. Experimental results reveal that not only does the proposed method perform well on the standard benchmark data sets, but also it generalizes well on the one-shot-learning gesture challenge. Furthermore, it is based on a simple statistical model and the intrinsic geometry of tensor space.

Keywords: gesture recognition, action recognition, Grassmann manifolds, product manifolds, one-shot-learning, kinect data

1. Introduction

Human gestures/actions are the natural way for expressing intentions and can be instantly recognized by people. We use gestures to depict sign language to deaf people, convey messages in noisy environments, and interface with computer games. Having automated gesture-based communication would broaden the horizon of human-computer interaction and enrich our daily lives. In recent years, many gesture recognition algorithms have been proposed (Mitra and Acharya, 2007; Wang et al., 2009; Bilinski and Bremond, 2011). However, reliable gesture recognition remains a challenging area due in part to the complexity of human movements. To champion the recognition performance, models are often complicated, causing difficulty for gener-

alization. Consequently, heavy-duty models may not have substantial gains in overall gesture recognition problems.

In this paper, we propose a new representation to gesture recognition based upon tensors and the geometry of product manifolds. Since human actions are expressed as a sequence of video frames, an action video may be characterized as a third order data tensor. The mathematical framework for working with high order tensors is multilinear algebra which is a useful tool for characterizing multiple factor interactions. Tensor computing has been successfully applied to many computer vision applications such as face recognition (Vasilescu and Terzopoulos, 2002), visual tracking (Li et al., 2007), and action classification (Vasilescu, 2002; Kim and Cipolla, 2009). However, the geometrical aspect of data tensors remains unexamined. The goal of this paper is to demonstrate the importance of the intrinsic geometry of tensor space where it provides a very discriminating structure for action recognition.

Notably, several recent efforts (Lui, 2012a) have been inspired by the characteristics of space and the associated construction of classifiers based upon the intrinsic geometry inherent in particular manifolds. Veeraraghavan et al. (2005) modeled human shapes from a shape manifold and expressed the dynamics of human silhouettes using an autoregressive (AR) model on the tangent space. Turaga and Chellappa (2009) extended this framework and represented the trajectories on a Grassmann manifold for activity classification. The use of tangent bundles on special manifolds was investigated by Lui (2012b) where a set of tangent spaces was exploited for action recognition. Age estimation was also studied using Grassmann manifolds (Turaga et al., 2010). The geodesic velocity from an average face to the given face was employed for age estimation where the space of landmarks was interpreted as a Grassmann manifold. Lui and Beveridge (2008) characterized tangent spaces of a registration manifold as elements on a Grassmann manifold for face recognition. The importance of the ordering on Stiefel manifolds was demonstrated by Lui et al. (2009) and an illumination model was applied to synthesize such elements for face recognition. These successes motivate the exploration of the underlying geometry of tensor space.

The method proposed in this paper characterizes action videos as data tensors and demonstrates their association with a product manifold. We focus attention on the intrinsic geometry of tensor space, and draw upon the fact that the geodesic on a product manifold is equivalent to the Cartesian product of geodesics from multiple factor manifolds. In other words, elements of a product manifold are the set of all elements inherited from factor manifolds. Thus, in our approach, action videos are factorized to three factor elements using Higher Order Singular Value Decomposition (HOSVD) in which the factor elements give rise to three factor manifolds. We further extend the product manifold representation to least squares regression. In doing so, we consider the underlying geometry and formulate least squares regression as a composite function. As such, we ensure that both the domain values and the range values reside on a manifold through the regression process. This yields a natural extension from Euclidean space to manifolds. The least squares fitted elements from a training set can then be exploited for gesture recognition where the similarity is expressed in terms of

the geodesic distance on a product manifold associated with fitted elements from factor manifolds.

We demonstrate the merits of our method on three gesture recognition problems including hand gestures, body gestures, and gestures collected from the Microsoft Kinect™ camera for the one-shot-learning CHALEARN gesture challenge. Our experimental results reveal that our method is competitive to the state-of-the-art methods and generalizes well to the one-shot-learning scheme, yet is based on a simple statistical model. The key contributions of the proposed work are summarized as follows:

- A new way of relating tensors on a product manifold to action recognition.
- A novel formulation for least squares regression on manifolds.
- The use of appearance and motion without explicitly modeling shapes or dynamics.
- A simple pixel-based representation (no silhouette or skeleton extraction).
- No extensive training and parameter tuning.
- No explicit assumption on action data.
- Competitive performance on gesture recognition.
- Applicable to other visual applications.

The rest of this paper is organized as follows: Related work is summarized in Section 2. Tensor algebra, orthogonal groups, and Grassmann manifolds are reviewed in Section 3. The formulation of the proposed product manifold is presented in Section 4 and is further elaborated with examples in Section 5. The statistical modeling on manifolds is introduced in Section 6. Section 7 reports our experimental results. Section 8 discusses the effect of using raw pixels for action recognition. Finally, we conclude this paper in Section 9.

2. Related Work

Many researchers have proposed a variety of techniques for action recognition in recent years. We highlight some of this work here, including bag-of-features models, autoregressive models, 3D Fourier transforms, tensor frameworks, and product spaces.

In the context of action recognition, bag-of-features models (Dollar et al., 2005; Wang et al., 2009; Bilinski and Bremond, 2011) may be among the most popular methods wherein visual vocabularies are learned from feature descriptors and spatiotemporal features are typically represented by a normalized histogram. While encouraging results have been achieved, bag-of-features methods have heavy training loads prior to classification. In particular, feature detection and codebook generation can consume tremendous amounts of time if the number of training samples is large. Recently, Wang et al. (2009) have evaluated a number of feature descriptors and bag-of-features models

for action recognition. This study concluded that different sampling strategies and feature descriptors were needed to achieve the best results on alternative action data sets. Similar conclusions were also found by Bilinski and Bremond (2011) where various sizes of codebooks are needed for different data sets in order to obtain peak performances.

Another school of thought for action classification is using an autoregressive (AR) model. Some of the earliest works involved dynamic texture recognition (Saisan et al., 2001) and human gait recognition (Bissacco et al., 2001). These works represented actions using AR models. The authors found that the most effective way to compare dynamics was by computing the Martin distance between AR models. Veeraraghavan et al. (2005) modeled human silhouettes based on Kendall's theory of shape (Kendall, 1984) where shapes were expressed on a shape manifold. This method modeled the dynamics of human silhouettes using an AR model on the tangent space of the shape manifold. The sequences of human shapes were compared by computing the distance between the AR models. Turaga and Chellappa (2009) investigated statistical modeling with AR models for human activity analysis. In their work, trajectories were considered a sequence of subspaces represented by AR models on a Grassmann manifold. As such, the dynamics were learned and kernel density functions with Procrustes representation were applied to density estimation.

Three-dimensional Fourier transform has been demonstrated as a valuable tool in action classification. Weinland et al. (2006) employed Fourier magnitudes and cylindrical coordinates to represent motion templates. Consequently, the action matching was invariant to translations and rotations around the z-axis. Although this method was view invariant, the training videos needed to be acquired from multiple cameras. Rodriguez et al. (2008) synthesized a filter respond using the Clifford Fourier transform for action recognition. The feature representation was computed using spatiotemporal regularity flow from the xy-parallel component. The advantage of using Clifford algebra is the direct use of vector fields to Fourier transform.

Data tensors are the multidimensional generalizations to matrices. Vasilescu (2002) modeled the joint angle trajectories on human motion as a set of factorized matrices from a data tensor. Signatures corresponding to motion and identity were then extracted using PCA for person identification. Kim and Cipolla (2009) extended canonical correlation analysis to the tensor framework by developing a Tensor Canonical Correlation Algorithm (TCCA). This method factorized data tensors to a set of matrices and learned a set of projection matrices maximizing the canonical correlations. The inner product was employed to compute the similarity between two data tensors. The use of SIFT features with CCA was also considered for gesture recognition by Kim and Cipolla (2007). Recently, nonnegative tensor factorization has been exploited for action recognition by Krausz and Bauckhage (2010) where action videos were factorized using a gradient descent method and represented as the sum of rank-1 tensors associated with a weighting factor. As a result, the appearance was captured by the basis images and the dynamics was encoded with the weighting factor.

Product spaces have received attention in applications related to spatiotemporal interactions. Datta et al. (2009) modeled the motion manifold as a collection of local linear models. This method learned a selection of mappings to encode the motion manifold from a product space. Lin et al. (2009) proposed a probabilistic framework for action recognition using prototype trees. Shape and motion were explicitly learned and characterized via hierarchical K-means clustering. The joint likelihood framework was employed to model the joint shape-motion space. Li and Chellappa (2010) investigated the product space of spatial and temporal submanifolds for action alignment. Sequential importance sampling was then used to find the optimal alignment. Despite these efforts, the geometry of the product space has not been directly considered and the geodesic nature on the product manifold remains unexamined.

3. Mathematical Background

In this section, we briefly review the background mathematics used in this paper. Particularly, we focus on the elements of tensor algebra, orthogonal groups, Stiefel manifolds, and Grassmann manifolds.

3.1. Tensor Representation

Tensors provide a natural representation for high dimensional data. We consider a video as a third order data tensor $\in \mathbb{R}^{X\times Y\times T}$ where X, Y, and T are the image width, image height, and video length, respectively. High order data tensors can be regarded as a multilinear mapping over a set of vector spaces. Generally, useful information can be extracted using tensor decompositions. In particular, a Higher Order Singular Value Decomposition (HOSVD) (De Lathauwer et al., 2000) is considered in this paper because the data tensor can be factorized in a closed-form. A recent review paper on tensor decompositions can be found in Kolda and Bader (2009). Before we describe HOSVD, we illustrate a building block operation called matrix unfolding.

3.1.1. MATRIX UNFOLDING

Let $\mathscr{A}$ be an order N data tensor $\in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$. The data tensor $\mathscr{A}$ can be converted to a set of matrices via a matrix unfolding operation. Matrix unfolding maps a tensor $\mathscr{A}$ to a set of matrices $A_{(1)}, A_{(2)}, \ldots, A_{(N)}$, where $A_{(k)} \in \mathbb{R}^{I_k\times(I_1\times\cdots\times I_{k-1}\times I_{k+1}\cdots\times I_N)}$ is a mode-k matrix of $\mathscr{A}$. An example of matrix unfolding of a third order, that is, $N = 3$, tensor is given in Figure 1. As Figure 1 shows, we can slice a third order tensor in three different ways along each axis and concatenate these slices into three different matrices $A_{(1)}$, $A_{(2)}$, and $A_{(3)}$ where the rows of an unfolded matrix are represented by a single variation of the tensor and the columns are composed by two variations of the tensor.

3.1.2. HIGHER ORDER SINGULAR VALUE DECOMPOSITION

Just as a data matrix can be factorized using a Singular Value Decomposition (SVD), a data tensor can also be factorized using Higher Order Singular Value Decomposition

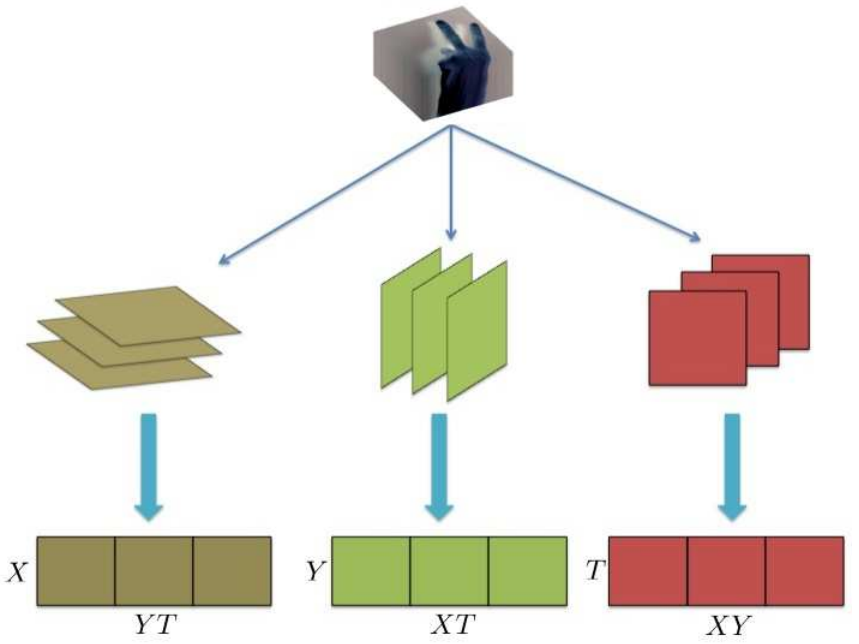


Figure 1: An example of matrix unfolding for a third order tensor. The illustration is for a video action sequence with two spatial dimensions X and Y and a temporal dimension T.

(HOSVD), also known as multilinear SVD. HOSVD operates on the unfolded matrices $A_{(k)}$, and each unfolded matrix may be factored using SVD as follows:

$$A_{(k)} = U^{(k)}\Sigma^{(k)}V^{(k)^T} \tag{1}$$

where $\Sigma^{(k)}$ is a diagonal matrix, $U^{(k)}$ is an orthogonal matrix spanning the column space of $A_{(k)}$ associated with nonzero singular values, and $V^{(k)}$ is an orthogonal matrix spanning the row space of $A_{(k)}$ associated with nonzero singular values. Then, an N order tensor can be decomposed using HOSVD as follows:

$$\mathscr{A} = \mathscr{S} \times_1 U^{(1)} \times_2 U^{(2)} \ldots \times_n U^{(N)}$$

where $\mathscr{S} \in \mathbb{R}^{(I_1 \times I_2 \times \cdots \times I_N)}$ is a core tensor, $U^{(1)}, U^{(2)}, \ldots, U^{(N)}$ are orthogonal matrices spanning the column space described in (1), and $\times_k$ denotes mode-k multiplication. The core tensor signifies the interaction of mode matrices and is generally not diagonal when the tensor order is greater than two.

3.2. Orthogonal Groups

Matrix Lie groups arise in various kinds of non-Euclidean geometry (Belinfante and Kolman, 1972). The General Linear Group[1] $\mathcal{GL}(n)$ is a set of nonsingular $n \times n$ matrices defined as:

$$\mathcal{GL}(n) = \{Y \in \mathbb{R}^{n \times n} : \det(Y) \neq 0\}.$$

The $\mathcal{GL}(n)$ is closed under a group operation, that is, matrix multiplication. This is because the product of two nonsingular matrices is a nonsingular matrix. Of practical importance here is the fact that elements of $\mathcal{GL}(n)$ are full rank and thus their row and column spaces span $\mathbb{R}^n$. A further subgroup of $\mathcal{GL}(n)$ is the orthogonal group denoted as:

$$\mathcal{O}(n) = \{Y \in \mathbb{R}^{n \times n} : Y^T Y = I\}.$$

It is known that the determinants of orthogonal matrices can be either $+1$ or -1 where the matrices with the determinant of 1 are rotation matrices and the matrices with the determinant of -1 are reflection matrices.

3.3. Stiefel Manifolds

The Stiefel manifold $\mathcal{V}_{n,p}$ is a set of $n \times p$ orthonormal matrices defined as:

$$\mathcal{V}_{n,p} = \{Y \in \mathbb{R}^{n \times p} : Y^T Y = I\}.$$

The Stiefel manifold $\mathcal{V}_{n,p}$ can be considered a quotient space of $\mathcal{O}(n)$ so we can identify an isotropy subgroup H of $\mathcal{O}(n)$ expressed as $\left\{ \begin{bmatrix} I_p & 0 \\ 0 & Q_{n-p} \end{bmatrix} : Q_{n-p} \in \mathcal{O}(n-p) \right\}$ where the isotropy subgroup leaves the element unchanged. Thus, the Stiefel manifold can be expressed as $\mathcal{V}_{n,p} = \mathcal{O}(n) \,/\, \mathcal{O}(n-p)$. From a group theory point of view, $\mathcal{O}(n)$ is a Lie group and $\mathcal{O}(n-p)$ is its subgroup so that $\mathcal{O}(n) \,/\, \mathcal{O}(n-p)$ represents the orbit space. In other words, $\mathcal{V}_{n,p}$ is the quotient group of $\mathcal{O}(n)$ by $\mathcal{O}(n-p)$.

3.4. Grassmann Manifolds

When we impose a group action of $\mathcal{O}(n)$ onto the Stiefel manifold, this gives rise to the equivalence relation between orthogonal matrices so that the elements of Stiefel manifolds are rotation and reflection invariant. In other words, elements are considered being equivalent if there exists a $p \times p$ orthogonal matrix Q_p which maps one point into the other. This equivalence relation can be written as:

$$\lfloor Y \rfloor = \{YQ_p : Q_p \in \mathcal{O}(n)\} \tag{2}$$

where $\lfloor Y \rfloor$ is an element on the Grassmann manifold. Therefore, the Grassmann manifold $\mathcal{G}_{n,p}$ is a set of p-dimensional linear subspaces of $\mathbb{R}^n$ and its isotropy subgroup composes all elements of $\left\{ \begin{bmatrix} Q_p & 0 \\ 0 & Q_{n-p} \end{bmatrix} : Q_p \in \mathcal{O}(p), Q_{n-p} \in \mathcal{O}(n-p) \right\}$. The quotient

1. In this paper, we are only interested in the field of real number $\mathbb{R}$. Unitary groups may be considered in other contexts.

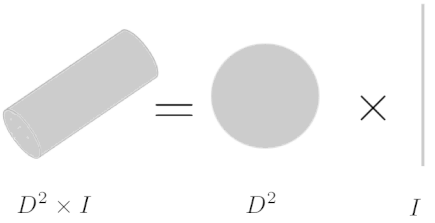


Figure 2: An example of a product manifold: A cylinder is a cross product of a circle and an interval.

representation of Grassmann manifolds is expressed as $\mathcal{G}_{n,p} = \mathcal{O}(n) / (\mathcal{O}(p) \times \mathcal{O}(n-p)) = \mathcal{V}_{n,p} / \mathcal{O}(p)$. As such, the element of the Grassmann manifold represents the orbit of a Stiefel manifold under the group action of orthogonal groups. More details on the treatment of Grassmann manifolds can be found in Edelman et al. (1998) and Absil et al. (2008).

4. Elements of Product Manifolds

This section discusses the elements of product manifolds in the context of gesture recognition. We illustrate the essence of product manifolds and the factorization of action videos. Further, we describe the realization of geodesic distance on the product manifold and its use for action classification.

4.1. Product Manifolds

A product manifold can be recognized as a complex compound object in a high dimensional space composed by a set of lower dimensional objects. For example, the product of a line with elements y in $\mathbb{R}^1$ and a solid circle with elements x in $\mathbb{R}^2$ becomes a cylinder with elements (x, y) in $\mathbb{R}^3$ as shown in Figure 2. Formally, this product topology can be expressed as:

$$\begin{aligned} I &= \{y \in \mathbb{R} : |y| < 1\}, \\ D^2 &= \{x \in \mathbb{R}^2 : |x| < 1\}, \\ D^2 \times I &= \{(x,y) \in \mathbb{R}^2 \times \mathbb{R} : |x| < 1 \text{ and } |y| < 1\} \end{aligned}$$

where D^2 and I are viewed as topological spaces.

The cylinder may be equally well interpreted as either a circle of intervals or an interval of circles. In general, a product manifold may be viewed as the cross section of lower dimensional objects. Formally, let $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_q$ be a set of manifolds. The set $\mathcal{M}_1 \times \mathcal{M}_2 \times \ldots \times \mathcal{M}_q$ is called the product of the manifolds where the manifold topology is equivalent to the product topology. Hence, a product manifold is defined

as:

$$\begin{aligned}\mathscr{M} &= \mathscr{M}_1 \times \mathscr{M}_2 \times \cdots \times \mathscr{M}_q \\ &= \{(x_1, x_2, \ldots, x_q) : x_1 \in \mathscr{M}_1, x_2 \in \mathscr{M}_2, \ldots, x_q \in \mathscr{M}_q\}\end{aligned}$$

where $\times$ denotes the Cartesian product, $\mathscr{M}_k$ represents a factor manifold (a topological space), and x_k is an element in $\mathscr{M}_k$. Note that the dimension of a product manifold is the sum of all factor manifolds (Lee, 2003).

The product manifold naturally expresses a compound topological space associated with a number of factor manifolds. For action video classification, third order data tensors are manifested as elements on three factor manifolds. As such, video data can be abstracted as points and classified on a product manifold.

4.2. Factorization in Product Spaces

As discussed in Section 3, HOSVD operates on the unfolded matrices (modes) via matrix unfolding in which the variation of each mode is captured by HOSVD. However, the traditional definition of HOSVD does not lead to a well-defined product manifold in the context of action recognition.

We observe that the column of every unfolded matrix $A_{(k)}$ is composed by multiple orders from the original data tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. This fact can also be observed in Figure 1. Let m be the dimension of the columns, $I_1 \times I_2 \times \cdots \times I_{k-1} \times I_{k+1} \cdots \times I_N$, and p be the dimension of the rows, I_k, for an unfolded matrix $A_{(k)}$. We can then assume that the dimension of the columns is greater than the dimension of the rows due to the nature of matrix unfolding for action videos, that is, $m > p$. This implies that the unfolded matrix $A_{(k)}$ only spans p dimensions.

Alternatively, one can factorize the data tensor using the right orthogonal matrices (Lui et al., 2010). From the context of action videos, the HOSVD can be expressed as:

$$\mathscr{A} = \hat{\mathscr{S}} \times_1 V^{(1)}_{\text{horizontal-motion}} \times_2 V^{(2)}_{\text{vertical-motion}} \times_3 V^{(3)}_{\text{appearance}}$$

where $\hat{\mathscr{S}}$ is a core tensor, $V^{(k)}$ are the orthogonal matrices spanning the row space with the first p rows associated with non-zero singular values from the unfolded matrices, respectively. Because we are performing action recognition on videos, the orthogonal matrices, $V^{(1)}_{\text{horizontal-motion}}$, $V^{(2)}_{\text{vertical-motion}}$, and $V^{(3)}_{\text{appearance}}$, correspond to horizontal motion, vertical motion, and appearance. Figure 3 shows some examples from the action decomposition.

From the factorization of HOSVD, each $V^{(k)}$ is a tall orthogonal matrix, thus it is an element on a Stiefel manifold. When we impose a group action of the orthogonal group, elements on the Stiefel manifold become rotation and reflection invariant. In other words, they are elements on the Grassmann manifold described in (2). As such, the action data are represented as the orbit of elements on the Stiefel manifold under the rotation and reflection actions with respect to appearance and dynamics. Section 5 will discuss how we benefit from imposing such a group action on the Stiefel manifold.

4.3. Geodesic Distance on Product Manifolds

The geodesic in a product manifold $\mathscr{M}$ is the product of geodesics in $\mathscr{M}_1, \mathscr{M}_2, \ldots, \mathscr{M}_q$ (Ma et al., 1998; Begelfor and Werman, 2006). Hence, for any differentiable curve γ parametrized by t, we have $\gamma(t) = (\gamma_i(t), \gamma_j(t))$ where γ is the geodesic on the product manifold $\mathscr{M}$, and γ_i and γ_j are the geodesics on the factor manifold $\mathscr{M}_i$ and $\mathscr{M}_j$ respectively. From this observation, the geodesic distance on a product manifold may be expressed as a Cartesian product of canonical angles computed by factor manifolds.

Just as there are alternatives to induce a metric on a Grassmann manifold (Edelman et al., 1998) using canonical angles, the geodesic distance on a product manifold could also be defined in different ways. One possible choice is the chordal distance that approximates the geodesic via a projection embedding (Conway et al., 1996). Consequently, we define the geodesic distance on a product manifold as:

$$d_{\mathscr{M}}(\mathscr{A}, \mathscr{B}) = \| \sin\Theta \|_2 \tag{3}$$

where $\mathscr{A}$ and $\mathscr{B}$ are the N order data tensors, $\Theta = (\theta_1, \theta_2, \ldots, \theta_N)$, and $\theta_k \in \mathscr{G}_k$ is a set of canonical angles (Björck and Golub, 1973) computed independently from each factor (Grassmann) manifold.

This development of geodesic distance on the product manifold can be related back to our cylinder example where a circle in $\mathbb{R}^2$ and a line in $\mathbb{R}^1$ form a cylinder in $\mathbb{R}^3$ where $\mathbb{R}^3$ is the product space. Recall that a Grassmann manifold is a set of p-dimensional linear subspaces. In analogous fashion, the product of a set of $p_1, p_2, \ldots, p_N$ linear subspaces forms a set of product subspaces whose dimension is $(p_1 + p_2 + \ldots + p_N)$. The product subspaces are the elements on a product manifold. This observation is consistent with the Θ in (3) where the number of canonical angles agrees with the dimension of product subspaces on the product manifold.

Note that canonical angles θ_k are measured between $V_{\mathscr{A}}^{(k)}$ and $V_{\mathscr{B}}^{(k)}$ where each is an orthogonal matrix spanning the row space associated with nonzero singular values from a mode-k unfolded matrix. As such, an N order tensor in $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ would span N row spaces in $I_1, I_2, \ldots, I_N$, respectively, and the dimension of a product manifold is the sum of each order of a data tensor, that is, $(\sum_{i=1}^{N} = I_1 + I_2 + \ldots + I_N)$.

5. The Product Manifold Representation

The tensor representation on a product manifold models the variations in both space and time for action videos. Specifically, the product manifold captures the individual characteristics of spatial and temporal evolution through three factor manifolds. As such, one factor manifold is acquiring the change in time, resulting in the appearance (XY) component, while the other two capture the variations in horizontal and vertical directions, demonstrating the horizontal motion (YT) and vertical motion (XT). Putting all these representations together, geodesic distance on the product manifold measures the changes in both appearance and dynamics.

The aim of this section is to illustrate how the product manifold characterizes appearance and dynamics from action videos. To visualize the product manifold represen-

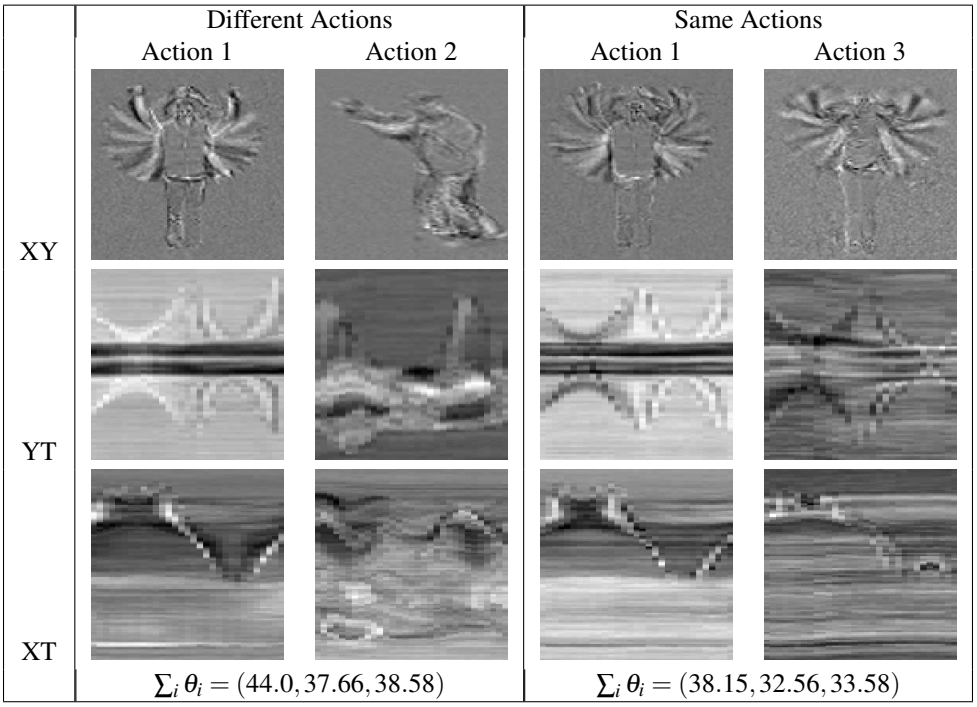


Figure 3: Examples of appearance and motion changes where the first row is the overlay appearances, the second and third rows are the overlay horizontal motion and vertical motion, and the bottom row gives the sum of canonical angles computed from each factorization of the pairs of canonical variates.

tation, let us consider the example given in Figure 3 where the first row expresses the pairs of overlay appearance (XY) canonical variates, the second and third rows reveal the pairs of overlay horizontal motion (YT) and vertical motion (XT) canonical variates, and the bottom row gives the sum of canonical angles computed from the pairs of canonical variates. Note that the canonical variates are elements on Stiefel manifolds. In the first column, two distinct actions are factorized to canonical variates. We can see that all canonical variates exhibit very different characteristics in both appearance and motions. On the contrary, the second column shows the same action performed by different actors and the canonical variates are much more similar than the first column, resulting in smaller canonical angles overall.

One of the advantages of the product manifold representation is that actions do not need to be aligned in temporal space. To demonstrate this merit, we permute the frame order from action 3 denoted as action 4 and match it to action 1. Figure 4 shows the pairs of canonical variates between actions (1, 3) and actions (1, 4). We should first note that the appearance (XY) of action 3 and action 4 span the same space despite the visual differences resulting in the identical sum of canonical angles 38.15. This is because elements on the Grassmann manifold are rotation and reflection invariant

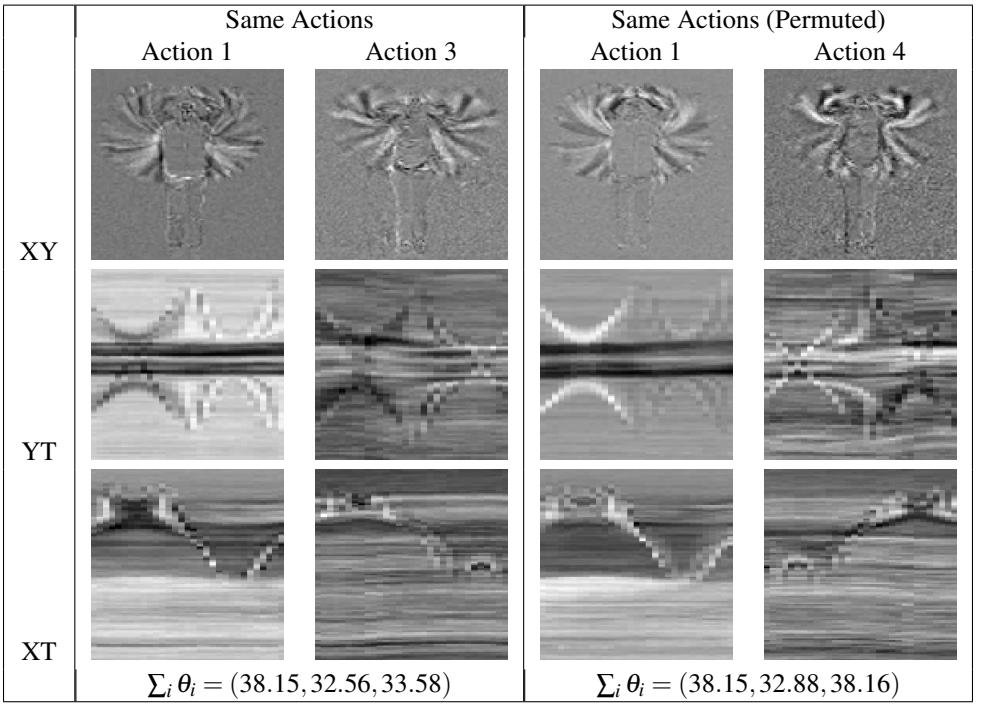


Figure 4: Examples of appearance and motion changes where Action 4 is a permuted version of Action 3. The canonical angles for the appearance indicates that the action is not affected by the frame order.

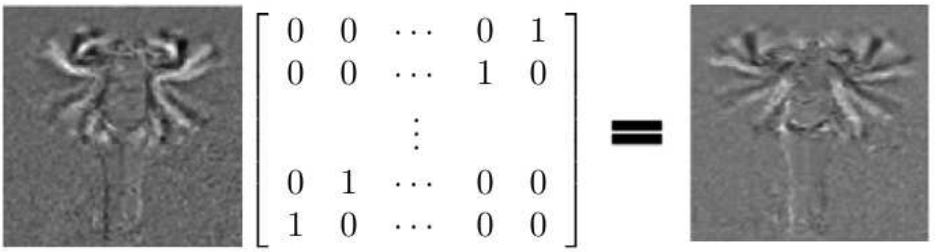


Figure 5: The characterization of the Grassmann manifold where a point is mapped to another point on the Stiefel manifold via an exchanged matrix. The group action is $(X,Q) \longmapsto XQ$ where $X \in \mathscr{V}_{n,p}$ and $Q \in \mathscr{O}(p)$ so that elements on the Grassmann manifold are closed under the orthogonal matrix multiplication.

from elements of the Stiefel manifold. This important concept is illustrated in Figure 5 where the exchange matrix $\mathscr{O}(p)$ maps the appearance of action 4 to the appearance of action 3.

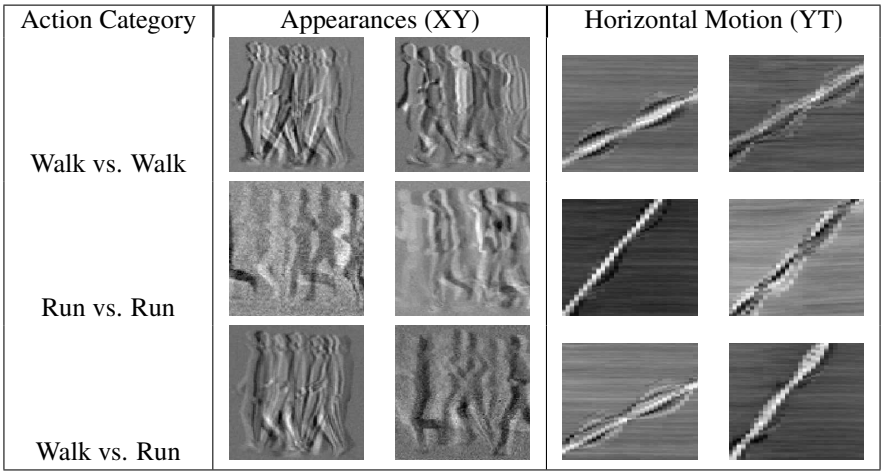


Figure 6: Illustration of capturing the rate of actions. The first column shows the change of appearance while the second column reveals the change of horizontal motion where the slopes exhibit the rate of motion.

In the example given in Figure 4, the most prominent change is related to the motion in vertical directions (XT) between action 3 and action 4. This arises from the fact that the change of motion mostly occurs in the vertical direction when we permute the order of the video frames from action 3. Consequently, the sum of canonical angles in XT varies from 33.58 to 38.16 which is less similar to action 1. When we identify a waving hand moving from top to bottom and from bottom to top, the vertical motion is the key feature. Otherwise, a simple cyclical search can compensate such variation. As a result, the product manifold representation is resilient to misregistration in the temporal space for appearance while keeping the dynamics intact.

Another intriguing attribute of the product manifold representation is its ability to capture the rate of motion, which is useful in identifying some particular actions. Figure 6 shows the pairs of canonical variates of two similar actions - walking and running. First, we note that there is little information from the vertical motion since the movements of walking and running occur horizontally. The appearance differences between walking and running are not substantial, which is shown in the first column of Figure 6. The key information between walking and running is embedded in the horizontal motion (YT). While the structure of horizontal motion between walking and running is similar exhibiting a line-like pattern, they have very distinct slopes shown in the horizontal motion column of Figure 6. These slopes characterize the rate of motion and are the key factors in recognizing these types of actions. In particular, when walking and running are compared depicted in the third row of Figure 6, the idiosyncratic aspect is captured by the rate of horizontal motion. In general, it is possible to see the rate of motion through both motion representations depending on the type of actions.

6. Statistical Modeling

Least squares regression is one of the fundamental techniques in statistical analysis. It is simple and often outperforms complicated models when the number of training samples is small (Hastie et al., 2001). Since video data do not reside in Euclidean space, we pay attention to the manifold structure. Here, we introduce a nonlinear regression framework in non-Euclidean space for gesture recognition. We formulate least squares regression as a composite function; as such, both domain and range values are constrained on a manifold through the regression process. The least squares fitted elements from a training set can then be exploited for gesture recognition.

6.1. Linear Least Squares Regression

Before we discuss the geometric extension, we will first review the standard form of least squares fitting. We consider a regression problem $y = A\beta$ where $y \in \mathbb{R}^n$ is the regression value, $A([a_1|a_2|\cdots|a_k]) \in \mathbb{R}^{n\times k}$ is the training set, and $\beta \in \mathbb{R}^k$ is the fitting parameter. The residual sum-of-squares can be written as:

$$R(\beta) = \| y - A\beta \|^2 \tag{4}$$

and the fitting parameter β can be obtained by minimizing the residual sum-of-squares error from (4). Then, we have

$$\hat{\beta} = (A^T A)^{-1} A^T y.$$

The regressed pattern from the training set has the following form

$$\hat{y} = A\hat{\beta} = A(A^T A)^{-1} A^T y. \tag{5}$$

The key advantage of least squares fitting is its simplicity and it intuitively measures the best fit of the data.

6.2. Least Squares Regression on Manifolds

Non-Euclidean geometry often arises in computer vision applications. We consider the nonlinear nature of space and introduce a geometric framework for least squares regression. First, we extend the linear least squares regression from (5) to a nonlinear form by incorporating a kernel function shown in the following

$$A(A \star A)^{-1}(A \star y)$$

where $\star$ is a nonlinear similarity operator. Obviously, $\star$ is equal to $x^T y$ in the linear case. In this paper, we employ the RBF kernel given as:

$$x \star y \quad = \quad \exp(-\frac{\sum_k \theta_k}{\sigma}) \tag{6}$$

where x and y are the elements on a factor manifold, θ_k is the canonical angle computed from the factor manifold, and σ is set to 2 in all our experiments. While other kernel

functions can be considered, our goal is to demonstrate our geometric framework and choose a commonly used RBF kernel operator.

Considering the similarity measure given in (6), the regression model becomes three sub-regression estimators given by

$$\psi^{(k)}(y) = A^{(k)}(A^{(k)} \star A^{(k)})^{-1}(A^{(k)} \star y^{(k)}) \tag{7}$$

where k denotes the mode of unfolding, $A^{(k)}$ is a set of orthogonal matrices factorized from HOSVD, and $y^{(k)}$ is an orthogonal matrix from the unfolded matrix.

To gain a better insight on the regression model, we explore the geometrical interpretation from (7). Given p training instances, the first element, $A^{(k)}$, is a set of factorized training samples residing on a manifold. Furthermore, $(A^{(k)} \star A^{(k)})^{-1}$ produces a $p \times p$ matrix from the training set and $(A^{(k)} \star y^{(k)})$ would create a $p \times 1$ vector. Therefore, the rest of the regression provides a weighting vector characterizing the training data on a factor manifold as:

$$\mathrm{w} = (A^{(k)} \star A^{(k)})^{-1}(A^{(k)} \star y^{(k)})$$

where the weighting vector is in a vector space, that is, $\mathrm{w} \in \mathscr{V}$.

Now, we have a set of factorized training samples, $A^{(k)}$, on a manifold and a weighting vector, w, in a vector space. To incorporate these two elements with the least squares fitting given in (7), we make a simple modification and reformulate the regression as follows

$$\Psi^{(k)}(y) = A^{(k)} \bullet (A^{(k)} \star A^{(k)})^{-1}(A^{(k)} \star y^{(k)}) \tag{8}$$

where $\bullet$ is an operator mapping points from a vector space back to a factor manifold. By introducing an additional operator, we ensure that both the domain values $y^{(k)}$ and the range values $\Psi^{k}(y)$ reside on a manifold. From a function composition point of view, the proposed regression technique can be viewed as a composition map $\mathscr{G} \circ \mathscr{H}$ where $\mathscr{H} : \mathscr{M} \longrightarrow \mathscr{V}$ and $\mathscr{G} : \mathscr{V} \longrightarrow \mathscr{M}$ where $\mathscr{M}$ is a manifold and $\mathscr{V}$ is a vector space.

One possible way to realize the composition map, $\mathscr{G} \circ \mathscr{H}$, is to employ the tangent space and modify the Karcher mean (Karcher, 1977). The computation of Karcher mean considers the intrinsic geometry and iteratively minimizes the distance between the updated mean and all data samples via the tangent space. Since w is the weighting vector, it naturally produces the weight between training samples. All we need is to apply the weighting vector to weight the training samples on a factor manifold. This is equivalent to computing the weighted Karcher mean, which is an element of a manifold.

So far, our geometric formulation on least squares regression is very general. To make it specific for gesture recognition, we impose rotation and reflection invariance to the factorized element $V^{(k)}$ such that they are elements on a Grassmann manifold and the computation of the weighted Karcher mean can be realized. Here, we sketch the pseudo-code in Algorithm 1. As Algorithm 1 illustrates, the first step is to initialize a base point on a manifold. To do so, we compute the weighted average from the training

Algorithm 1: Weighted Karcher Mean Computation

Initialize a base point μ on a manifold
while *not converged* **do**
    Apply the logarithmic map to the training samples Y_i to the base point μ
    Compute the weighted average on the tangent space at the base point μ
    Update the base point μ by applying the exponential map on the weighted average
end

samples in Euclidean space and project it back to the Grassmann manifold using QR factorization. Then, we iteratively update the base point on the Grassmann manifold. The update procedure involves the standard logarithmic map and the exponential map on Grassmann manifolds (Edelman et al., 1998) described as follows

$$\log_\mu(Y_i) \quad = \quad U_1 \Theta_1 V_1^T$$

where μ is the base point for the tangent space, Y_i is a training instance factorized from the Grassmann manifold, $\mu_\perp \mu_\perp^T Y_i (\mu^T Y_i)^{-1} = U_1 \Sigma_1 V_1^T$, $\Theta_1 = \arctan(\Sigma_1)$, and $\mu_\perp$ is the orthogonal complement to μ.

$$\exp_\mu(\Delta) \quad = \quad \mu V_2 \cos(\Sigma_2) + U_2 \sin(\Sigma_2)$$

where Δ is the weighted tangent vector at μ and $\Delta = U_2 \Sigma_2 V_2^T$. From a geometric point of view, the logarithmic operator maps a point on a manifold to a tangent space whereas the exponential map projects a point in the tangent space back to the manifold. A pictorial illustration is given in Figure 7. In addition, the Karcher mean calculation exhibits fast convergence (Absil et al., 2004). Typically, convergence can be reached within 10 iterations in our experiments. A sample run is depicted in Figure 8 where expeditious reduction of residuals occurs in the first few iterations.

To perform gesture recognition, a set of training videos is collected. All videos are normalized to a standard size. During the test phase, the category of a query video is determined by

$$j^* = \operatorname*{argmin}_j \mathscr{D}(Y, \Psi_j(Y))$$

where Y is a query video, Ψ_j is the regression instance for the class j given in (8), and $\mathscr{D}$ is a geodesic distance measure. Because the query gesture Y and the regression instance are realized as elements on a product manifold, we employ the chordal distance given in (3) for gesture classification.

In summary, the least squares regression model applies HOSVD on a query gesture Y and factorizes it to three sub-regression models $(\Psi_j^{(1)}, \Psi_j^{(2)}, \Psi_j^{(3)})$ on three Grassmann manifolds where regressions are performed. The distance between the regression output and query is then characterized on a product manifold; gesture recognition is achieved using the chordal distance. We note that our least squares framework is applicable

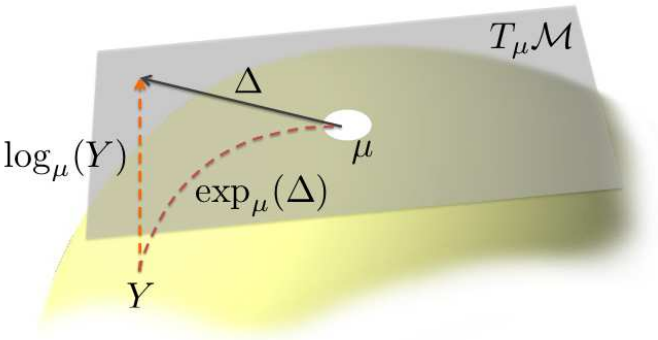


Figure 7: An illustration of logarithmic and exponential maps where Y and μ are points on a manifold, Δ is the tangent vector, and $T_\mu\mathscr{M}$ is the tangent space at μ.

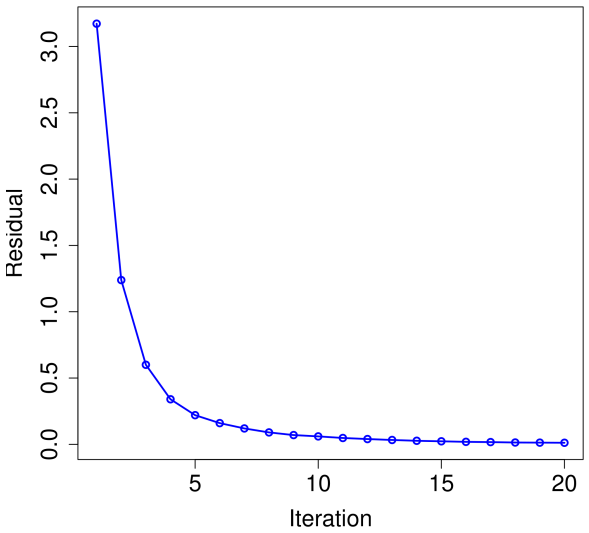


Figure 8: The residual values of tangent vectors.

to many matrix manifolds as long as the logarithmic and exponential maps are well-defined. Furthermore, when the kernel operator is $\star = x^T y$, $\log_x(y) = y$, and $\exp_x(\Delta) = x + \Delta$, the regression model in (8) becomes the canonical least squares regression in Euclidean space.

When statistical models exhibit high variance, shrinkage techniques are often applied (Hastie et al., 2001). We see that a simple regularization parameter turns least

Figure 9: Hand gesture samples. Flat-Leftward, Flat-Rightward, Flat-Contract, Spread-Leftward, Spread-Rightward, Spread-Contract, V-Shape-Leftward, V-Shape-Rightward, and V-Shape-Contract.

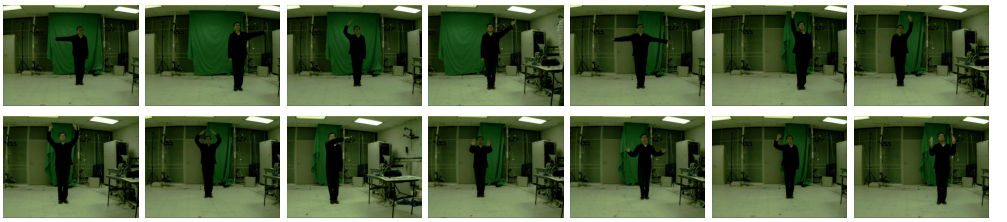


Figure 10: Body gesture samples. First row: Turn Left, Turn Right, Attention Left, Attention Right, Attention Both, Stop Left, and Stop Right. Second row: Stop Both, Flap, Start, Go Back, Close Distance, Speed Up, and Come Near.

squares regression into ridge regression. This observation can also be applied to our non-Euclidean least squares regression framework.

7. Experimental Results

This section summarizes our empirical results and demonstrates the proficiency of our framework on gesture recognition. To facilitate comparison, we first evaluate our method using two publicly available gesture data sets namely Cambridge hand-gesture (Kim and Cipolla, 2009) and UMD Keck body-gesture (Lin et al., 2009). We further extend our method to the one-shot-learning gesture challenge (CHALEARN, 2011). Our experiments reveal that not only does our method perform well on the standard benchmark data sets, but also it generalizes well on the one-shot-learning gesture challenge.

7.1. Cambridge Hand-Gesture Data Set

Our first experiment is conducted using the Cambridge hand-gesture data set which has 900 video sequences with nine different hand gestures (100 video sequences per gesture class). The gesture data are collected from five different illumination sets labeled as Set1, Set2, Set3, Set4, and Set5. Example gestures are shown in Figure 9.

We follow the experimental protocol employed by Kim and Cipolla (2009) where Set5 is the target set, and Set1, Set2, Set3, and Set4 are the test sets. The target Set5 is further partitioned into a training set and validation set (90 video sequences in the

Method	Set1	Set2	Set3	Set4	Total
Graph Embedding (Yuan et al., 2010)	-	-	-	-	82%
TCCA (Kim and Cipolla, 2009)	81%	81%	78%	86%	82±3.5%
DCCA+SIFT (Kim and Cipolla, 2007)	-	-	-	-	85±2.8%
RLPP (Harandi et al., 2012)	86%	86%	85%	88%	86.3±1.3%
TB$\{\mathscr{Y}_{n,p}\}$ (Lui, 2012b)	88%	84%	85%	87%	86±3.0%
PM 1-NN (Lui et al., 2010)	89%	86%	89%	87%	88±2.1%
Our Method	93%	89%	91%	94%	91.7±2.3%

Table 1: Recognition results on the Cambridge Hand-Gesture data set (Five trial runs).

Method	Static Setting	Dynamic Setting
HOG3D (Bilinski and Bremond, 2011)	-	53.6%
Shape Manifold (Abdelkadera et al., 2011)	82%	-
MMI-2+SIFT (Qiu et al., 2011)	95%	-
CC K-Means (Jiang et al., 2012))	-	92.9%
Prototype-Tree (Lin et al., 2009)	95.2%	91.1%
TB$\{\mathscr{Y}_{n,p}\}$ (Lui, 2012b)	92.1%	91.1%
PM 1-NN (Lui et al., 2010)	92.9%	92.3%
Our Method	94.4%	92.3%

Table 2: Recognition results on the UMD Keck Body-Gesture data set.

training set and 90 video sequences in the validation set). We employ five random trials in selecting the training and validation videos in Set5. The recognition results are summarized in Table 1 where the classification rates are the average accuracy obtained from five trial runs followed by the standard deviation. As Table 1 shows, our method performs very well across all illumination sets obtaining 91.7% average classification rate.

7.2. UMD Keck Body-Gesture Data Set

The UMD Keck body-gesture data set consists of 14 naval body gestures acquired from both static and dynamic backgrounds. In the static background, the subjects and the camera remain stationary whereas the subjects and the camera are moving in the dynamic environment during the performance of the gesture. There are 126 videos collected from the static scene and 168 videos taken from the dynamic environment. Example gestures are given in Figure 10.

We follow the experimental protocol proposed by Lin et al. (2009) for both static and dynamic settings. The region of interest is tracked by a simple correlation filter. In the static background, the protocol is leave-one-subject-out (LOSO) cross-validation. As for the dynamic environment, the gestures acquired from the static scene are used for training while the gestures collected from the dynamic environment are the test videos. The recognition results for both static and dynamic backgrounds are reported in Table 2.

We can see that our method is competitive to the current state-of-the-art methods in both protocols. One of the key advantages of our method is its direct use of raw pixels while the prototype-tree (Lin et al., 2009), MMI-2+SIFT (Qiu et al., 2011), and CC K-means (Jiang et al., 2012) methods operate on silhouette images which require image segmentation prior to classification. This makes our representation more generic.

7.3. One-Shot-Learning Gesture Challenge

Microsoft Kinect™ has recently revolutionized gesture recognition by providing both RGB and depth images. To facilitate the adaptation to new gestures, CHALEARN (Guyon et al., 2012) has organized a one-shot-learning challenge for gesture recognition.

The key aspect of one-shot-learning is to perform machine learning on a single training example. As such, intra-class variability needs to be modeled from a single example or learned from different domains. While traditional machine learning techniques require a large amount of training data to model the statistical distribution, least squares regression appears to be more robust when the size of training samples is limited (Hastie et al., 2001). We employ our least squares regression framework and model the intra-class variability by synthesizing training examples from the original training instance. Consequently, we apply the same regression framework on the product manifold to the one-shot-learning gesture challenge.

One of the gesture variations is performing gesture positions. Our initial studies for frame alignment did not yield positive results due in part to the incidental features of the upper body. Since gesture positions are the key source of variations, we synthesize training examples for translational instances on both RGB and depth images. The synthesized examples are generated by shifting the entire action video horizontally and vertically. Specifically, we synthesize two vertically (up/down) and four horizontally (left/right) translated instances along with the original training example. As such, we have seven training instances for RGB and depth images, respectively. We stress that we do not apply any spatial segmentation or intensity normalization to video data; alignment is the only variation that we synthesize for one-shot-learning. Our experiments on the training batches indicate that there is about 2% gain by introducing the translational variations.

We assess the effectiveness of the proposed framework on the development data set for the one-shot-learning gesture challenge. The development data set consists of 20 batches of gestures. Each batch is made of 47 gesture videos and split into a training set and a test set. The training set includes a small set of vocabulary spanning from 8 to 15 gestures. Every test video contains 1 to 5 gestures. Detailed descriptions of the gesture data can be found in Guyon et al. (2012).

Since the number of gestures varies for test videos, we perform temporal segmentation to localize each gesture segment. It is supposed that the actor will return to the resting position before performing a new gesture. Thus, we employ the first frame as a template and compute the correlation coefficient with subsequent frames. We can then localize the gesture segments by identifying the peak locations from the correlations;

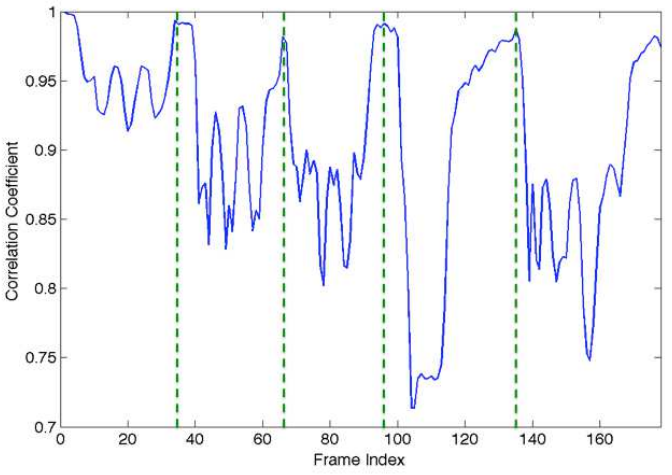


Figure 11: An illustration of temporal segmentation where the dash lines indicate the peak locations and the resting frames from the action sequence.

the number of gestures is the number of peaks + 1. An illustration of temporal segmentation is given in Figure 11 where the peak locations provide a good indication for the resting frames. Furthermore, we fix the spatial dimension to 32×32 and dynamically determine the number of frames by selecting 90% of the PCA energy from each training batch. Linear interpolation is then applied to normalize the video length.

The recognition performance is evaluated using the Levenshtein distance (Levenshtein, 1966), also known as edit distance. Table 3 shows the average errors over 20 batches. As Table 3 reveals, our method significantly outperforms the baseline algorithm (CHALEARN, 2011) and achieves 28.73% average Levenshtein distance per gesture on the development data set. Our method also ranks among the top algorithms in the gesture challenge (Guyon et al., 2012). This illustrates that our method can be effectively adopted for one-shot-learning from the traditional supervised learning paradigm.

While our method performs well on the one-shot-learning gesture challenge, it is not a complete system yet. There are three particular batches that cause difficulties for our algorithm. These batches are devel03, devel10, and devel19 where the example frames are shown in Figure 12. These three batches share a common characteristic that the gesture is only distinguishable by identifying the hand positions. Since we do not have a hand detector, the gross motion dominates the whole action causing it to be confused with other similar gestures.

Another source of errors is made by the temporal segmentation. While the actor is supposed to return to the resting position before performing a new gesture, this rule has not always been observed. As a result, such variation introduces a mismatch between the template and subsequent frames resulting errors in partitioning the video

	Baseline		Our Method	
Batch	TeLev%	TeLen%	TeLev%	TeLen%
devel01	53.33	12.22	13.33	4.44
devel02	68.89	16.67	35.56	14.44
devel03	77.17	5.43	71.74	20.65
devel04	52.22	30.00	10.00	2.22
devel05	43.48	10.87	9.78	7.61
devel06	66.67	17.78	37.78	14.44
devel07	81.32	19.78	18.68	3.30
devel08	58.43	12.36	8.99	5.62
devel09	38.46	9.89	13.19	1.10
devel10	75.82	21.98	50.55	1.10
devel11	67.39	18.48	35.87	2.17
devel12	52.81	5.62	22.47	4.49
devel13	50.00	17.05	9.09	2.27
devel14	73.91	22.83	28.26	3.26
devel15	50.00	8.70	21.74	0.00
devel16	57.47	17.24	31.03	6.90
devel17	66.30	32.61	30.43	4.35
devel18	70.00	28.89	40.00	11.11
devel19	71.43	15.38	49.45	3.30
devel20	70.33	36.26	35.16	12.09
Average	62.32	18.01	28.73	6.24

Table 3: Recognition results on the development data for the one-shot-learning challenge where TeLev is the sum of the Levenshtein distance divided by the true number of gestures and TeLen is the average error made on the number of gestures.

sequence. The large error in devel03 is caused by the need for hand positions and temporal segmentation. Future work will focus on combining both appearance and motion for temporal segmentation.

Nevertheless, the experimental results from the Cambridge hand-gesture and the UMD Keck body-gesture data sets are encouraging. These findings illustrate that our method is effective in both hand gestures and body gestures. Once we have a reliable hand detector, we expect to further improve gesture recognition from a single training example. Currently, the processing time on 20 batches (2,000 gestures) including both training and testing is about 2 hours with a non-optimized MATLAB implementation on a 2.5GHz Intel Core i5 iMac.

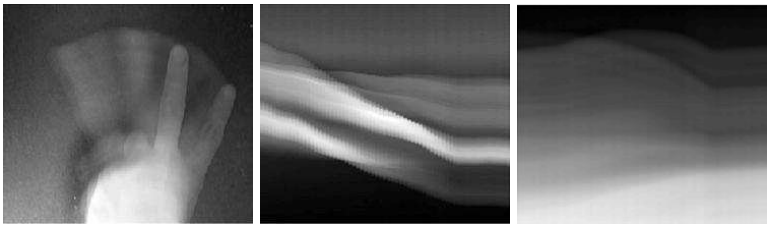
8. Discussion

The proposed method is geometrically motivated. It decomposes a video tensor to three Stiefel manifolds via HOSVD where the orthogonal elements are imposed to Grassman-

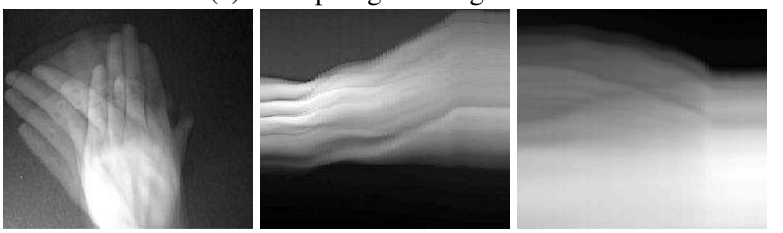
Figure 12: Gesture samples on the one-shot-learning gesture challenge (devel03, devel10, and devel19).

nian spaces. As mentioned before, one of the key advantages of our method is its direct use of raw pixels. This gives rise to a practical and important question. *How robust can the raw pixel representation be against background clutter?*

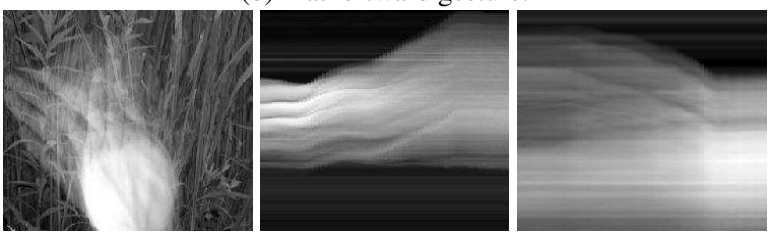
To address this concern, we synthesize an illustrative example given in Figure 13. The first, second, and third columns depict the appearance, horizontal motion, and vertical motion of the gesture, respectively. A V-shape rightward gesture and a flat leftward gesture are shown in the first row and second row. We superpose a cluttered background on every frame of the flat leftward gesture exhibited in the third row. While the appearances between the uniform flat gesture and the cluttered flat gesture emerge differently, the deterioration on the dynamics is quite minimal. As a result, the gesture performed with the background clutter can still be discriminated against other gestures. Numerically, the sum of the canonical angles between the uniform (second row) and the cluttered background (third row) gestures is (56.09, 7.99, 9.17) resulting in a geodesic distance of 5.91 on the product manifold. In contrast, the sum of the canonical angles between the V-shape (first row) and the flat (second row) gestures is (76.35, 23.66, 18.42) yielding a geodesic distance of 8.29. In addition, when the V-shape gesture (first row) matches against the cluttered flat gesture (third row), the sum of the canonical angles is (76.09, 23.75, 18.84) and the geodesic distance is 8.31. This finding reveals that the geodesic distance between the uniform and cluttered background gestures are quite similar against inter-class gestures, while the geodesic distance is significantly smaller for the intra-class gestures. Hence, raw pixels can be directly exploited in our representation.



(a) V-shape rightward gesture.



(b) Flat leftward gesture.



(c) Superposed cluttered background on the flat leftward gesture.

Figure 13: The effect of background clutter. Appearance, horizontal motion, and vertical motion are depicted in the first, second, and third columns, respectively.

As technology advances, we can now separate the foreground and background more easily using a Kinect™ camera. We hypothesize that better recognition results may be obtained when the foreground gestures are extracted. On the other hand, our method can still perform gracefully when a cluttered background is present.

9. Conclusions

This paper promotes the importance of the underlying geometry of data tensors. We have presented a geometric framework for least squares regression and applied it to gesture recognition. We view action videos as third order tensors and impose them on a product manifold where each factor is Grassmannian. The realization of points on these Grassmannians is achieved by applying HOSVD to a tensor representation of the action video. A natural metric is inherited from the factor manifolds since the geodesic on the product manifold is given by the product of the geodesic on the Grassmann manifolds.

The proposed approach provides a useful metric and a regression model based on latent geometry for action recognition. To account for the underlying geometry, we

formulate least squares regression as a composite function. This formulation provides a natural extension from Euclidean space to manifolds. Experimental results demonstrate that our method is effective and generalizes well to the one-shot-learning scheme.

For longer video sequences, micro-action detection is needed which may be modeled effectively using HMM. Future work will focus on developing more sophisticated models for gesture recognition and other regression techniques on matrix manifolds for visual applications.

References

M. F. Abdelkadera, W. Abd-Almageeda, A. Srivastavab, and R. Chellappa. Gesture and action recognition via modeling trajectories on riemannian manifolds. *Computer Vision and Image Understanding*, 115(3):439–455, 2011.

P.-A. Absil, R. Mahony, and R. Sepulchre. Riemannian geometry of grassmann manifolds with a view on algorithmic computation. *Acta Applicandae Mathematicae*, 80 (2):199–220, 2004.

P.-A Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

E. Begelfor and M. Werman. Affine invariance revisited. In *IEEE Conference on Computer Vision and Pattern Recognition, New York*, 2006.

J.G.F. Belinfante and B. Kolman. *A Survey of Lie Groups and Lie Algebras with Applications and Computational Methods*. SIAM, 1972.

P. Bilinski and F. Bremond. Evaluation of local descriptors for action recognition in videos. In *ICVS*, 2011.

A. Bissacco, A. Chiuso, Y. Ma, and S. Soatto. Recognition of human gaits. In *IEEE Conference on Computer Vision and Pattern Recognition, Hawaii*, pages 270–277, 2001.

Å. Björck and G.H. Golub. Numerical methods for computing angles between linear subspaces. *Mathematics of Computation*, 27:579–594, 1973.

CHALEARN. Chalearn gesture dataset (cgd 2011), chalearn, california, 2011.

J.H. Conway, R.H. Hardin, and N.J.A. Sloane. Packing lines, planes, etc.: Packings in grassmannian spaces. *Experimental Mathematics*, 5(2):139–159, 1996.

A. Datta, Y. Sheikh, and T. Kanade. Modeling the product manifold of posture and motion. In *Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (in conjunction with ICCV)*, 2009.

L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000.

P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (in conjunction with ICCV)*, 2005.

A. Edelman, R. Arias, and S. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, 20(2):303–353, 1998.

I. Guyon, V. Athitsos, P. Jangyodsuk, B. Hammer, and H. J. E. Balderas. Chalearn gesture challenge: Design and first results. In *CVPR Workshop on Gesture Recognition*, 2012.

M. T. Harandi, C. Sanderson, A. Wiliem, and B. C. Lovell. Kernel analysis over riemannian manifolds for visual recognition of actions, pedestrians and textures. In *WACV*, 2012.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

Z. Jiang, Z. Lin, and L. Davis. Class consistent k-means: Application to face and action recognition. *Computer Vision and Image Understanding*, 116(6):730–741, 2012.

H. Karcher. Riemannian center of mass and mollifier smoothing. *Comm. Pure Appl. Math.*, 30(5):509–541, 1977.

D. Kendall. Shape manifolds, procrustean metrics and complex projective spaces. *Bull. London Math. Soc.*, 16:81–121, 1984.

T-K. Kim and R. Cipolla. Gesture recognition under small sample size. In *Asian Conference on Computer Vision*, 2007.

T-K. Kim and R. Cipolla. Canonical correlation analysis of video volume tensors for action categorization and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(8):1415–1428, 2009.

T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3), September 2009.

B. Krausz and C. Bauckhage. Action recognition in videos using nonnegative tensor factorization. In *International Conference on Pattern Recognition*, 2010.

J. Lee. *Introduction to Smooth Manifolds*. Springer, 2003.

V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710, 1966.

R. Li and R. Chellappa. Group motion segmentation using a spatio-temporal driving force model. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

X. Li, W. Hu, Z. Zhang, X. Zhang, and G. Luo. Robust visual tracking based on incremental tensor subspace learning. In *IEEE International Conference on Computer Vision*, 2007.

Z. Lin, Z. Jiang, and L. Davis. Recognizing actions by shape-motion prototype trees. In *IEEE International Conference on Computer Vision*, 2009.

Y. M. Lui. Advances in matrix manifolds for computer vision. *Image and Vision Computing*, 30(6-7):380–388, 2012a.

Y. M. Lui. Tangent bundles on special manifolds for action recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(6):930–942, 2012b.

Y. M. Lui and J. R. Beveridge. Grassmann registration manifolds for face recognition. In *European Conference on Computer Vision, Marseille, France*, 2008.

Y. M. Lui, J. R. Beveridge, and M. Kirby. Canonical stiefel quotient and its application to generic face recognition in illumination spaces. In *IEEE International Conference on Biometrics : Theory, Applications and Systems, Washington, D.C.*, 2009.

Y. M. Lui, J. R. Beveridge, and M. Kirby. Action classification on product manifolds. In *IEEE Conference on Computer Vision and Pattern Recognition, San Francisco*, 2010.

Y. Ma, J. Košecká, and S. Sastry. Optimal motion from image sequences: A riemannian viewpoint, 1998. Technical Report No. UCB/ERL M98/37, EECS Department, University of California, Berkeley.

S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, Cybernetics - Part C: Applications and Reviews*, 37:311–324, 2007.

Q. Qiu, Z. Jiang, and R. Chellappa. Sparse dictionary-based representation and recognition of action attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

M. Rodriguez, J. Ahmed, and M. Shah. Action mach: A spatio-temporal maximum average correlation height filter for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

P. Saisan, G. Doretto, Y-N. Wu, and S. Soatto. Dynamic texture recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

P. Turaga and R. Chellappa. Locally time-invariant models of human activities using trajectories on the grassmannian. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

P. Turaga, S. Biswas, and R. Chellappa. The role of geometry for age estimation. In *IEEE International conference Acoustics, Speech and Signal Processing*, 2010.

M. A. O. Vasilescu. Human motion signatures: Analysis, synthesis, recognition. In *International Conference on Pattern Recognition, Quebec City, Canada*, pages 456–460, 2002.

M. A. O. Vasilescu and D. Terzopoulos. Multilinear image analysis for facial recognition. In *International Conference on Pattern Recognition, Quebec City, Canada*, pages 511–514, 2002.

A. Veeraraghavan, A. K. Roy-Chowdhury, and R. Chellappa. Matching shape sequences in video with applications in human movement analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1896–1909, 2005.

H. Wang, M. Ullah, A Klaser, I. Laptev, and C. Schmid. Evaulation of local spatio-temporal features for action recognition. In *British Machine Vision Conference*, 2009.

D. Weinland, R. Ronfard, and E. Boyer. Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104:249–257, 2006.

Y. Yuan, H. Zheng, Z. Li, and D. Zhang. Video action recognition with spatio-temporal graph embedding and spline modeling. In *ICASSP*, 2010.

JMLR: Workshop and Conference Proceedings 13:2205–2231, 2012 Published 7/12

Sign Language Recognition using Sub-Units

Helen Cooper H.M.COOPER@SURREY.AC.UK
Eng-Jon Ong E.ONG@SURREY.AC.UK
Nicolas Pugeault N.PUGEAULT@SURREY.AC.UK
Richard Bowden R.BOWDEN@SURREY.AC.UK
Centre for Vision Speech and Signal Processing
University of Surrey, Guildford. GU2 9PY UK

Editor: Isabelle Guyon and Vassilis Athitsos

Abstract

This paper discusses sign language recognition using linguistic sub-units. It presents three types of sub-units for consideration; those learnt from appearance data as well as those inferred from both 2D or 3D tracking data. These sub-units are then combined using a sign level classifier; here, two options are presented. The first uses Markov Models to encode the temporal changes between sub-units. The second makes use of Sequential Pattern Boosting to apply discriminative feature selection at the same time as encoding temporal information. This approach is more robust to noise and performs well in signer independent tests, improving results from the 54% achieved by the Markov Chains to 76%.

Keywords: sign language recognition, sequential pattern boosting, depth cameras, sub-units, signer independence, data set

1. Introduction

This paper presents several approaches to sub-unit based Sign Language Recognition (SLR) culminating in a real time Kinect^{TM}demonstration system. SLR is a non-trivial task. Sign Languages (SLs) are made up of thousands of different signs; each differing from the other by minor changes in motion, handshape, location or Non-Manual Featuress (NMFs). While Gesture Recognition (GR) solutions often build a classifier per gesture, this approach soon becomes intractable when recognising large lexicons of signs, for even the relatively straightforward task of citation-form, dictionary look-up. Speech recognition was faced with the same problem; the emergent solution was to recognise the subcomponents (phonemes), then combine them into words using Hidden Markov Models (HMMs). Sub-unit based SLR uses a similar two stage recognition system, in the first stage, sign linguistic sub-units are identified. In the second stage, these sub-units are combined together to create a sign level classifier.

Linguists also describe SLs in terms of component sub-units; by using these sub-units, not only can larger sign lexicons be handled efficiently, allowing demonstration on databases of nearly 1000 signs, but they are also more robust to the natural variations of signs, which occur on both an inter and an intra signer basis. This makes

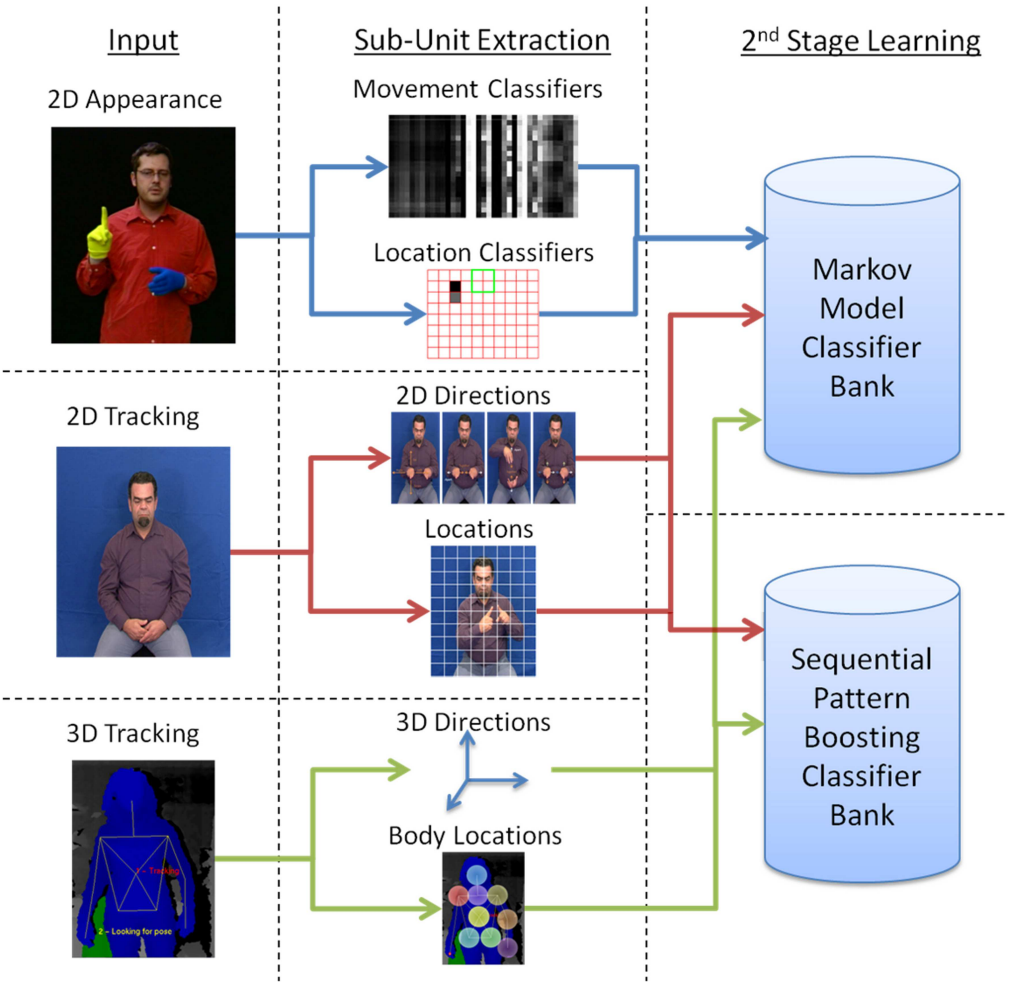


Figure 1: Overview of the 3 types of sub-units extracted and the 2 different sign level classifiers used.

them suited to real-time signer independent recognition as described later. This paper will focus on 4 main sub-unit categories based on *HandShape*, *Location*, *Motion* and *Hand-Arrangement*. There are several methods for labelling these sub-units and this work builds on both the Ha, Tab, Sig, Dez system from the BSL dictionary (British Deaf Association, 1992) and The Hamburg Notation System (HamNoSys), which has continued to develop over recent years to allow more detailed description of signs from numerous SLs (Hanke and Schmaling, 2004).

This paper presents a comparison of sub-unit approaches, focussing on the advantages and disadvantages of each. Also presented is a newly released Kinect data set, containing multiple users performing signs in various environments. There are three different types of sub-units considered; those based on appearance data alone, those

which use 2D tracking data with appearance based handshapes and those which use 3D tracking data produced by a Kinect^{TM}sensor. Each of these three sub-unit types is tested with a Markov model approach to combine sub-units into sign level classifiers. A further experiment is performed to investigate the discriminative learning power of Sequential Pattern (SP) Boosting for signer independent recognition. An overview is shown in Figure 1.

2. Background

The concept of using sub-units for SLR is not novel. Kim and Waldron (1993) were among the first adopters, they worked on a limited vocabulary of 13-16 signs, using data gloves to get accurate input information. Using the work of Stokoe (1960) as a base, and their previous work in telecommunications (Waldron and Simon, 1989), they noted the need to break signs into their component sub-units for efficiency. They continued this throughout the remainder of their work, where they used phonemic recognition modules for hand shape, orientation, position and movement recognition (Waldron and Kim, 1994). They made note of the dependency of position, orientation and motion on one another and removed the motion aspect allowing the other sub-units to compensate (on a small vocabulary, a dynamic representation of position is equivalent to motion) (Waldron and Kim, 1995).

The early work of Vogler and Metaxas (1997) borrowed heavily from the studies of sign language by Liddell and Johnson (1989), splitting signs into motion and pause sections. Their later work (Vogler and Metaxas, 1999), used parallel HMMs on both hand shape and motion sub-units, similar to those proposed by the linguist Stokoe (1960). Kadir et al. (2004) took this further by combining head, hand and torso positions, as well as hand shape, to create a system based on hard coded sub-unit classifiers that could be trained on as little as a single example.

Alternative methods have looked at data driven approaches to defining sub-units. Yin et al. (2009) used an accelerometer glove to gather information about a sign, they then applied discriminative feature extraction and 'similar state tying' algorithms, to decide sub-unit level segmentation of the data. Whereas Kong and Ranganath (2008) and Han et al. (2009) looked at automatic segmentation of sign motion into sub-units, using discontinuities in the trajectory and acceleration to indicate where segments begin and end. These were then clustered into a code book of possible exemplar trajectories using either Dynamic Time Warping (DTW) distance measures Han et al. or Principal Component Analysis (PCA) Kong and Ranganath.

Traditional sign recognition systems use tracking and data driven approaches (Han et al., 2009; Yin et al., 2009). However, there is an increasing body of research that suggests using linguistically derived features can offer superior performance. Cooper and Bowden (2010) learnt linguistic sub-units from hand annotated data which they combined with Markov models to create sign level classifiers, while Pitsikalis et al. (2011) presented a method which incorporated phonetic transcriptions into sub-unit based statistical models. They used HamNoSys annotations combined with the Postures, Detentions, Transitions, Steady Shifts (PDTS) phonetic model to break the signs

and annotations into labelled sub-units. These were used to construct statistical sub-unit models which they combined via HMMs.

The frequent requirement of tracked data means that the Kinect™device has offered the sign recognition community a short-cut to real-time performance. In the relatively short time since its release, several proof of concept demonstrations have emerged. Ershaed et al. (2011) have focussed on Arabic sign language and have created a system which recognises isolated signs. They present a system working for 4 signs and recognise some close up handshape information (Ershaed et al., 2011). At ESIEA they have been using Fast Artificial Neural Networks to train a system which recognises two French signs (Wassner, 2011). This small vocabulary is a proof of concept but it is unlikely to be scalable to larger lexicons. It is for this reason that many sign recognition approaches use variants of HMMs (Starner and Pentland, 1997; Vogler and Metaxas, 1999; Kadir et al., 2004; Cooper and Bowden, 2007). One of the first videos to be uploaded to the web came from Zafrulla et al. (2011) and was an extension of their previous CopyCat game for deaf children (Zafrulla et al., 2010). The original system uses coloured gloves and accelerometers to track the hands. By tracking with a Kinect™, they use solely the upper part of the torso and normalise the skeleton according to arm length (Zafrulla et al., 2011). They have an internal data set containing 6 signs; 2 subject signs, 2 prepositions and 2 object signs. The signs are used in 4 sentences (subject, preposition, object) and they have recorded 20 examples of each. Their data set is currently single signer, making the system signer dependent, while they list under further work that signer independence would be desirable. By using a cross validated system they train HMMs (Via the Georgia Tech Gesture Toolkit Lyons et al., 2007) to recognise the signs. They perform 3 types of tests, those with full grammar constraints achieving 100%, those where the number of signs is known achieving 99.98% and those with no restrictions achieving 98.8%.

2.1. Linguistics

Sign language sub-units can be likened to speech phonemes, but while a spoken language such as English has only 40-50 phonemes (Shoup, 1980), SLs have many more. For example, *The Dictionary of British Sign Language/English* (British Deaf Association, 1992) lists 57 'Dez' (*HandShape*), 36 'Tab' (*Location*), 8 'Ha' (*Hand-Arrangement*), 28 'Sig' (*Motion*) (plus 4 modifiers, for example, short and repeated) and there are two sets of 6 'ori' (*Orientation*), one for the fingers and one for the palm.

HamNoSys uses a more combinatorial approach to sub-units. For instance, it lists 12 basic handshapes which can be augmented using finger bending, thumb position and openeness characteristics to create a single *HandShape* sub-unit. These handshapes are then combined with palm and finger orientations to describe the final hand posture. *Motion* sub-units can be simple linear directions, known as 'Path Movements' these can also be modified by curves, wiggles or zigzags. *Motion* sub-units can also be modified by locations, for example, move from A to B with a curved motion or move down beside the nose.

In addition, whereas spoken phonemes are broadly sequential, sign sub-units are parallel, with some sequential elements added where required. This means that each of the 57 British Sign Language (BSL) *HandShape* options can (theoretically) be in any one of the 36 BSL *Orientation* combinations. In practice, due to the physical constraints of the human body, only a subset of comfortable combinations occur, yet this subset is still considerable.

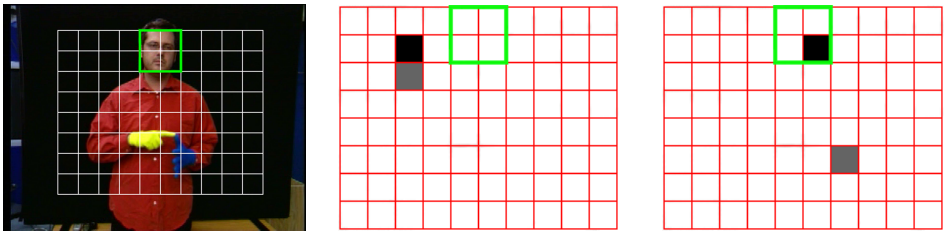
An advantage of the parallel nature of sub-units, is that they can be recognised independently using different classifiers, then combined at the word level. The reason this is advantageous is that *Location* classifiers need to be spatially variant, since they describe where a sign happens. *Hand-Arrangement* should be spatially invariant but not rotationally variant, since they describe positional relationships between the hands. While *Motion* are a mixture of spatially, temporally, rotationally and scale variant sub-units since they describe types of motion which can be as generic as 'hands move apart' or more specific such as 'hand moves left'. Therefore each type of sub-unit can be recognised by classifiers incorporating the correct combination of invariances. This paper presents three methods for extracting sub-units; learnt appearance based (Section 3), hard coded 2D tracking based (Section 4) and hard coded 3D tracking based (Section 5).

3. Learning Appearance Based Sub-units

The work in this section learns a subset of each type of sub-unit using AdaBoost from hand labelled data. As has been previously discussed, not all types of sub-units can be detected using the same type of classifier. For *Location* sub-units, there needs to be correlation between where the motion is happening and where the person is; to this end spatial grid features centred around the face of the signer are employed. For *Motion* sub-units, the salient information is what type of motion is occurring, often regardless of its position, orientation or size. This is approached by extracting moment features and using Binary Patterns (BPs) and additive classifiers based on their changes over time. *Hand-Arrangement* sub-units look at where the hands are in relation to each other, so these are only relevant for bi-manual signs. This is done using the same moment features as for *Motion* but this time over a single frame, as there is no temporal context required. All of these sub-unit level classifiers are learnt using AdaBoost (Freund and Schapire, 1995). The features used in this section require segmentation of the hands and knowledge of where the face is. The Viola Jones face detector (Viola and Jones, 2001) is used to locate the face. Skin segmentation could be used to segment the hands, but since sub-unit labels are required this work uses the data set from the work of Kadir et al. (2004) for which there is an in-house set of sub-unit labels for a portion of the data. This data set was created using a gloved signer and as such a colour segmentation algorithm is used in place of skin segmentation.

3.1. *Location* Features

In order that the sign can be localised in relation to the signer, a grid is applied to the image, dependent upon the position and scale of the face detection. Each cell in the grid



(*a*) The grid applied over the signer (*b*) On Right Shoulder (*c*) Lower Face/Chin

Figure 2: Grid features for two stage classification. (a) shows an example of the grid produced from the face dimensions while (b) and (c) show grid features chosen by boosting for two of the 18 *Location* sub-units. The highlighted box shows the face location and the first and second features chosen, are shown in black and grey respectively.

is a quarter of the face size and the grid is 10 rectangles wide by 8 deep, as shown in Figure 2(*a*). These values are based on the signing space of the signer. However, in this case, the grid does not extend beyond the top of the signers head since the data set does not contain any signs which use that area. The segmented frame is quantised into this grid and a cell fires if over 50% of its pixels are made up of glove/skin. This is shown in Equation 1 where R_{wc} is the weak classifier response and $\Lambda_{skin}(x,y)$ is the likelihood that a pixel contains skin. f is the face height and all the grid values are relative to this dimension.

$$R_{wc} = \begin{cases} 1 \text{ if } \frac{f^2}{8} < \sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_2} (\Lambda_{skin}(i,j) > 0), \\ 0 \text{ otherwise.} \end{cases}$$

Where x_1, y_1, x_2, y_2 are given by

$$\forall G_x, \forall G_y \begin{cases} x_1 = G_x f, \\ x_2 = (G_x + 0.5) f, \\ y_1 = G_y f, \\ y_2 = (G_y + 0.5) f, \end{cases}$$
$$\text{given } G_x = \{-2.5, -2, -1.5 \ldots 2\},$$
$$G_y = \{-4, -3.5, -3 \ldots 0\}. \quad (1)$$

For each of the *Location* sub-units, a classifier was built via AdaBoost to combine cells which fire for each particular sub-unit, examples of these classifiers are shown in Figures 2(*b*) and 2(*c*). Note how the first cell to be picked by the boosting (shown in

black) is the one directly related to the area indicated by the sub-unit label. The second cell chosen by boosting either adds to this location information, as in Figure 2(*b*), or comments on the stationary, non-dominant hand, as in Figure 2(*c*).

Some of the sub-units types contain values which are not mutually exclusive, this needs to be taken into account when labelling and using sub-unit data. The BSL dictionary (British Deaf Association, 1992) lists several *Location* sub-units which overlap with each other, such as face and mouth or nose. Using boosting to train classifiers requires positive and negative examples. For best results, examples should not be contaminated, that is, the positive set should not contain negatives and the negative set should not contain positives. Trying to distinguish between an area and its sub-areas can prove futile, for example, the mouth is also on the face and therefore there are likely to be false negatives in the training set when training face against mouth. The second stage, sign-level classification does not require the sub-unit classifier responses to be mutually exclusive. As such a hierarchy can be created of *Location* areas and their sub-areas. This hierarchy is shown in Figure 3; a classifier is trained for each node of the tree, using examples which belong to it, or its children, as positive data. Examples which do not belong to it, its parent or its child nodes provide negative data.

This eliminates false negatives from the data set and avoids confusion. In Figure 3 the ringed nodes show the sub-units for which there exist examples. Examples are labelled according to this hierarchy, for example, face, face_lower or face_lower_mouth which makes finding children and parents easier by using simple string comparisons.

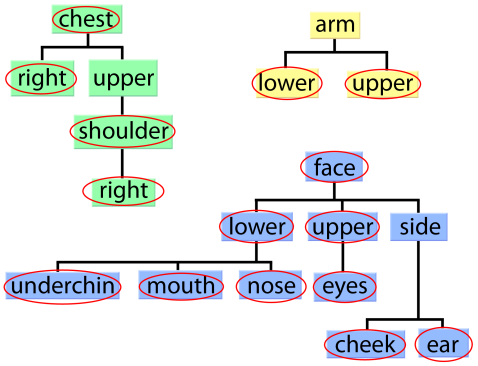


Figure 3: The three *Location* sub-unit trees used for classification. There are three separate trees, based around areas of the body which do not overlap. Areas on the leaves of the tree are sub-areas of their parent nodes. The ringed labels indicate that there are exact examples of that type in the data set.

3.2. *Motion* and *Hand-Arrangement* Moment Feature Vectors

For *Hand-Arrangement* and *Motion*, information regarding the arrangement and motion of the hands is required. Moments offer a way of encoding the shapes in an image; if vectors of moment values per frame are concatenated, then they can encode the change in shape of an image over time.

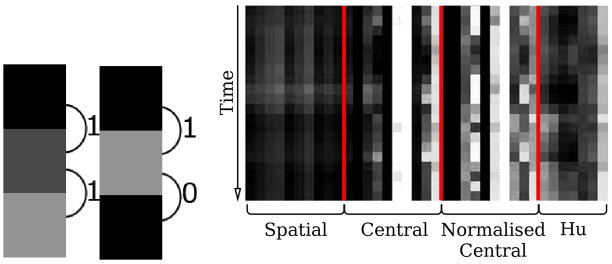
There are several different types of moments which can be calculated, each of them displaying different properties. Four types were chosen to form a feature vector, $\mathbf{m}$: spatial, m_{ab}, central, μ_{ab}, normalised central, $\bar{\mu}_{ab}$ and the Hu set of invariant moments (Hu, 1962) $\mathscr{H}_1$-$\mathscr{H}_7$. The order of a moment is defined as $a+b$. This work uses all moments, central moments and normalised central moments up to the 3rd order, 10 per type, (00, 01, 10, 11, 20, 02, 12, 21, 30, 03). Finally, the Hu set of invariant moments are considered, there are 7 of these moments and they are created by combining the normalised central moments, see Hu (1962) for full details, they offer invariance to scale, translation, rotation and skew. This gives a 37 dimensional feature vector, with a wide range of different properties.

$$R_{wc} = \begin{cases} 1 \text{ if } \mathscr{T}_{wc} < \mathbf{M}_{i,t}, \\ 0 \text{ otherwise.} \end{cases} \tag{2}$$

Since spatial moments are not invariant to translation and scale, there needs to be a common point of origin and similar scale across examples. To this end, the spatial moments are treated in a similar way to the spatial features in Section 3.1, by centring and scaling the image about the face of the signer before computation. For training *Hand-Arrangement*, this vector is used to boost a set of thresholds for individual moments, $\mathbf{m}_i$ on a given frame t, Equation 2. For *Motion*, temporal information needs to be included. Therefore the video clips are described by a stack of these vectors, $\mathbf{M}$, like a series of 2D arrays, as shown in Figure 4(a) where the horizontal vectors of moments are concatenated vertically, the lighter the colour, the higher the value of the moment on that frame.

3.3. *Motion* Binary Patterns and Additive Classifiers

As has been previously discussed, the *Motion* classifiers are looking for changes in the moments over time. By concatenating feature vectors temporally as shown in Figure 4(b), these spatio-temporal changes can be found. Component values can either increase, decrease or remain the same, from one frame to the next. If an increase is described as a 1 and a decrease or no change is described as a 0 then a BP can be used to encode a series of increases/decreases. A temporal vector is said to match the given BP if every '1' accompanies an increase between concurrent frames and every '0' a decrease/'no change'. This is shown in Equation 3 where $\mathbf{M}_{i,t}$ is the value of the component, $\mathbf{M}_i$, at time t and $\mathbf{bp}_t$ is the value of the BP at frame t.



(*a*) BP example (*b*) Concatenated Moment Vector

Figure 4: Moment vectors and Binary Patterns for two stage classification. (a) A pictorial description of moment vectors (normalised along each moment type for a selection of examples), the lighter the colour the larger the moment value. (b) BP, working from top to bottom an increase in gradient is depicted by a 1 and a decrease or no change by a 0.

$$
\begin{aligned}
R_{wc} &= ||\max_{\forall t}(BP(\mathbf{M}_{i,t}))| - 1|, \\
BP(\mathbf{M}_{i,t}) &= \mathbf{bp}_t - d(\mathbf{M}_{i,t}, \mathbf{M}_{i,t+1}), \\
d(\mathbf{M}_{i,t}, \mathbf{M}_{i,t+1}) &= \begin{cases} 0 \text{ if } \mathbf{M}_{i,t} \leq \mathbf{M}_{i,t+1}, \\ 1 \text{ otherwise.} \end{cases}
\end{aligned} \tag{3}
$$

See Figure 5 for an example where feature vector A makes the weak classifier fire, whereas feature vector B fails, due to the ringed gradients being incompatible.

Discarding all magnitude information would possibly remove salient information. To retain this information, boosting is also given the option of using additive classifiers. These look at the average magnitude of a component over time. The weak classifiers are created by applying a threshold, $\mathscr{T}_{wc}$, to the summation of a given component, over several frames. This threshold is optimised across the training data during the boosting phase. For an additive classifier of size T, over component $\mathbf{m}_i$, the response of the classifier, R_{wc}, can be described as in Equation 4.

$$
R_{wc} = \begin{cases} 1 \text{ if } \mathscr{T}_{wc} \leq \sum_{t=0}^{T} \mathbf{M}_{i,t}, \\ 0 \text{ otherwise.} \end{cases} \tag{4}
$$

Boosting is given all possible combinations of BPs, acting on each of the possible components. The BPs are limited in size, being between 2 and 5 changes (3 - 6 frames) long. The additive features are also applied to all the possible components, but the lengths permitted are between 1 and 26 frames, the longest mean length of *Motion* sub-units. Both sets of weak classifiers can be temporally offset from the beginning of an example, by any distance up to the maximum distance of 26 frames.

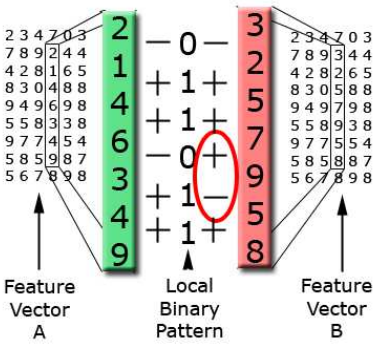
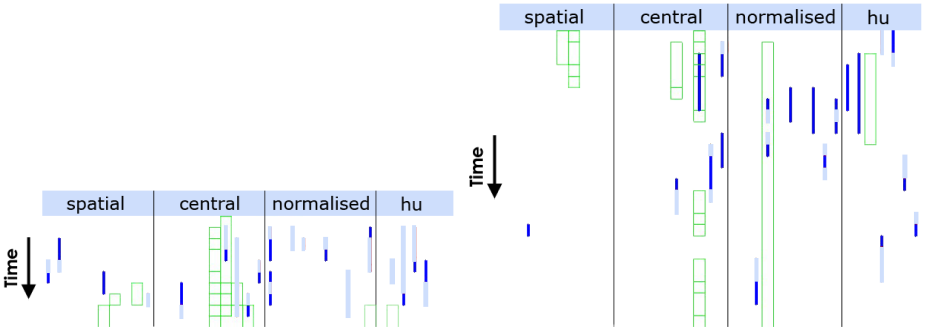


Figure 5: An example of a BP being used to classify two examples. A comparison is made between the elements of the weak classifiers BP and the temporal vector of the component being assessed. If every '1' in the BP aligns with an increase in the component and every '0' aligns with a decrease or 'no change' then the component vector is said to match (e.g., case A). However if there are inconsistencies as ringed in case B then the weak classifier will not fire.



(*a*) hands_move_apart (*b*) Hands_move_together

Figure 6: Boosted temporal moments BP and additive *Motion* classifiers. The moment vectors are stacked one frame ahead of another. The boxes show where an additive classifier has been chosen, a dark line shows a decreasing moment value and a pale line an increasing value.

Examples of the classifiers learnt are shown in Figure 6, additive classifiers are shown by boxes, increasing BPs are shown by pale lines and decreasing ones by dark lines. When looking at a sub-unit such as 'hands move apart' (Figure 6(*a*)), the majority of the BP classifiers show increasing moments, which is what would be expected, as the eccentricity of the moments is likely to increase as the hands move apart. Conversely, for 'hands move together' (Figure 6(*b*)), most of the BPs are decreasing.

Since some *Motion* sub-units occur more quickly than others, the boosted classifiers are not all constrained to being equal in temporal length. Instead, an optimal length is chosen over the training set for each individual sub-unit. Several different length classifiers are boosted starting at 6 frames long, increasing in steps of 2 and finishing at 26 frames long. Training classification results are then found for each sub-unit and the best length chosen to create a final set of classifiers, of various lengths suited to the sub-units being classified.

4. 2D Tracking Based Sub-Units

Unfortunately, since the learnt, appearance based, sub-units require expertly annotated data they are limited to data sets with this annotation. An alternative to appearance based features is given by tracking. While tracking errors can propagate to create sub-unit errors, the hand trajectories offer significant information which can aid recognition. With the advances of tracking systems and the real-time solution introduced by the Kinect^{TM}, tracking is fast becoming an option for real-time, robust recognition of sign language. This section works with hand and head trajectories, extracted from videos by the work outlined by Roussos et al. (2010). The tracking information is used to extract *Motion* and *Location* information. *HandShape* information is extracted via Histograms of Gradients (HOGs) on hand image patches and learnt from labels using random forests. The labels are taken from the linguistic representations of Sign Gesture Mark-up Language (SiGML) (Elliott et al., 2001) or HamNoSys (Hanke and Schmaling, 2004).[1]

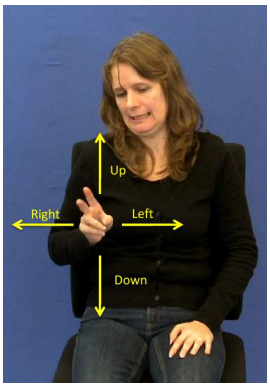
4.1. *Motion* Features

In order to link the x,y co-ordinates obtained from the tracking to the abstract concepts used by sign linguists, rules are employed to extract HamNoSys based information from the trajectories. The approximate size of the head is used as a heuristic to discard ambient motion (that less than 0.25 the head size) and the type of motion occurring is derived directly from deterministic rules on the x and y co-ordinates of the hand position. The types of motions encoded are shown in Figure 7, the single handed motions are available for both hands and the dual handed motions are orientation independent so as to match linguistic concepts.

4.2. *Location* Features

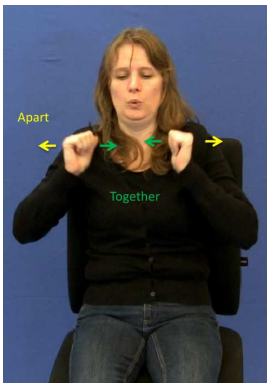
Similarly the x and y co-ordinates of the sign location need to be described relative to the signer rather than in absolute pixel positions. This is achieved via quantisation of the values into a codebook based on the signer's head position and scale in the image. For any given hand position (x_h, y_h) the quantised version (x'_h, y'_h) is achieved using the

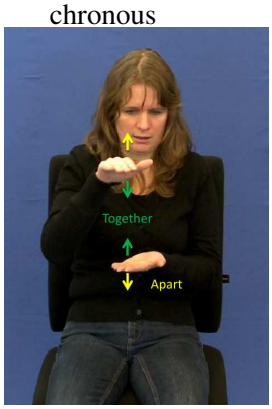
1. Note that conversion between the two forms is possible. However while HamNoSys is usually presented as a font for linguistic use, SiGML is more suited to automatic processing.



(*a*) Single handed

(*b*) Bimanual: Synchronous





(*c*) Bimanual: Together/Apart

Figure 7: Motions detected from tracking

quantisation rules shown in Equation 5, where (x_f, y_f) is the face position and (w_f, h_f) is the face size.

$$x' = (x_h - x_f)/w_f,$$
$$y' = (y_h - y_f)/h_f. \tag{5}$$

Due to the limited size of a natural signing space, this gives values in the range of $y' \in \{0..10\}$ and $x' \in \{0..8\}$ which can be expressed as a binary feature vector of size 36, where the x and y positions of the hands are quantised independently.

4.3. *HandShape* Features

While just the motion and location of the signs can be used for recognition of many examples, it has been shown that adding the handshape can give significant improvement (Kadir et al., 2004). HOG descriptors have proven efficient for sign language hand shape recognition (Buehler et al., 2009) and these are employed as the base feature unit.

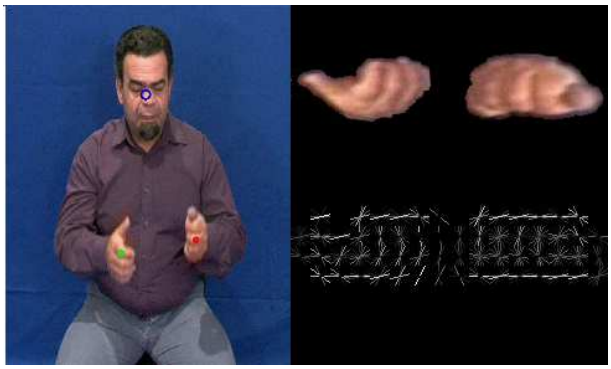


Figure 8: Example HOGs extracted from a frame

In each frame, the signer's dominant hand is segmented using the x,y position and a skin model. These image patches are rotated to their principal axis and scaled to a square, 256 pixels in size. Examples of these image patches are shown in Figure 8 beside the frame from which they have been extracted. HOGs are calculated over these squares at a cell size of 32 pixels square with 9 orientation bins and with 2x2 overlapping blocks, these are also shown in Figure 8. This gives a feature vector of 1764 histogram bins which describes the appearance of a hand.

4.4. *HandShape* Classifiers

This work focusses on just the 12 basic handshapes, building multi-modal classifiers to account for the different orientations. A list of these handshapes is shown in Figure 9.

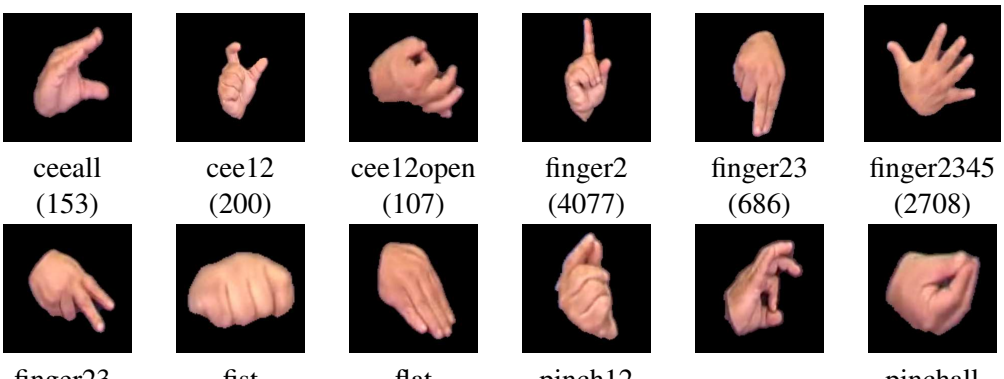
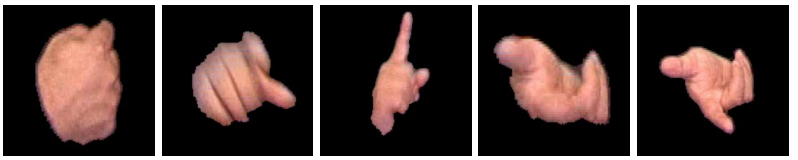


Figure 9: The base handshapes (Number of occurrences in the data set)

Unfortunately, linguists annotating sign do so only at the *sign* level while most sub-units occur for only *part* of a sign. Also, not only do handshapes change throughout the



(*a*) (*b*) (*c*) (*d*) (*e*)

Figure 10: A variety of examples for the HamNoSys/SiGML class 'finger2'.

sign, they are made more difficult to recognise due to motion blur. Using the motion of the hands, the sign can be split into its component parts (as in Pitsikalis et al., 2011), that are then aligned with the sign annotations. These annotations are in HamNoSys and have been prepared by trained experts, they include the sign breakdown but not the temporal alignment. The frames most likely to contain a static handshape (i.e., those with limited or no motion) are extracted for training.

Note that, as shown in Figure 10, a single SiGML class (in this case 'finger2') may contain examples which vary greatly in appearance, making visual classification an extremely difficult task.

The extracted hand shapes are classified using a multi-class random forest. Random forests were proposed by Amit and Geman (1997) and Breiman (2001). They have been shown to yield good performance on a variety of classification and regression problems, and can be trained efficiently in a parallel manner, allowing training on large feature vectors and data sets. In this system, the forest is trained from automatically extracted samples of all 12 handshapes in the data set, shown in Figure 9. Since signs may have multiple handshapes or several instances of the same handshape, the total occurrences are greater than the number of signs, however they are not equally distributed between the handshape classes. The large disparities in the number of examples between classes (see Figure 9) may bias the learning, therefore the training set is rebalanced before learning by selecting 1,000 random samples for each class, forming a new balanced data set. The forest used consists of $N = 100$ multi-class decision trees T_i, each of which is trained on a random subset of the training data. Each tree node splits the feature space in two by applying a threshold on one dimension of the feature vector. This dimension (chosen from a random subset) and the threshold value are chosen to yield the largest reduction in entropy in the class distribution. This recursive partitioning of the data set continues until a node contains a subset of examples that belong to one single class, or if the tree reaches a maximal depth (set to 10). Each leaf is then labelled according to the mode of the contained samples. As a result, the forest yields a probability distribution over all classes, where the likelihood for each class is the proportion of trees that voted for this class. Formally, the confidence that feature vector x describes the handshape c is given by:

$$p[c] = \frac{1}{N} \sum_{i<N} \delta_c(T_i(x)),$$

handshape	predictions											
flat	**0.35**	0.19	0.09	0.03	0.08	0.06	0.03	0.06	0.06	0.01	0.03	0.01
fist	0.03	**0.69**	0.02	0.04	0.11	0.05		0.02	0.03			0.02
finger2345	0.16	0.19	**0.36**	0.02	0.03	0.05		0.06	0.02	0.03	0.06	0.01
finger2	0.02	0.33	0.07	**0.31**	0.11	0.05	0.02	0.03	0.02		0.04	
pinchall	0.03	0.09	0.04	0.01	**0.65**	0.11	0.01	0.01				0.04
pinch12	0.02	0.20	0.01	0.02	0.13	**0.56**	0.01		0.01		0.01	0.02
finger23	0.05	0.17	0.04	0.02	0.05	0.04	**0.54**	0.01			0.07	0.01
pinch12op	0.03	0.12	0.07	0.01	0.15	0.04	0.01	**0.56**				0.01
cee12	0.01	0.05	0.01	0.03	0.04			0.01	**0.82**		0.01	
cee12open					0.01					**0.99**		
finger23sp	0.01	0.15	0.02		0.06	0.01	0.05	0.02			**0.65**	
ceeall	0.01	0.08	0.03		0.08	0.01	0.02	0.01			0.01	**0.77**

Table 1: Confusion matrix of the handshape recognition, for all 12 classes.

where N is the number of trees in the forest, $T_i(x)$ is the leaf of the ith tree T_i into which x falls, and $\delta_c(a)$ is the Kronecker delta function ($\delta_c(a) = 1$ iff. $c = a$, $\delta_c(a) = 0$ otherwise).

The performance of this hand shape classification on the test set is recorded on Table 1, where each row corresponds to a shape, and each column corresponds to a predicted class (empty cells signify zero). Lower performance is achieved for classes that are more frequent in the data set. The more frequently a handshape occurs in the data set the more orientations it is likely to be used in. This in turn makes the appearance of the class highly variable; see, for example, Figure 10 for the case of 'finger2'—the worst performing case. Also noted is the high confusion between 'finger2' and 'fist' most likely due to the similarity of these classes when the signer is pointing to themselves.

The handshape classifiers are evaluated for the right hand only during frames when it is not in motion. The sign recognition system is evaluated using two different encodings for the detected hand shapes. As will be described in Section 6, the next stage classifier requires inputs in the form of binary feature vectors. Two types of 12 bit binary feature vector can be produced from the classifier results. The first method applies a strict Winner Takes All (WTA) on the multi-class forest's response: the class with the highest probability is set to one, and the others to zero. For every non-motion frame, the vector contains a true value in the highest scoring class. The second method applies a fixed threshold ($\tau = 0.25$) on the confidences provided by the classifier for each of the 12 handshapes classes. Handshapes that have a confidence above threshold ($p[c] > \tau$) are set to one, and the others to zero. This soft approach carries the double advantage that a) the feature vector may encode the ambiguity between handshapes, which may itself carry information, and b) may contain only zeros if confidences in all classes are small.

Locations	Motions		
	Right or Left Hand		Bi-manual
head	left	$\Delta x > \lambda$	in sync
neck	right	$\Delta x < -\lambda$	$\lvert\delta(L,R)\rvert < \lambda$
torso	up	$\Delta y > \lambda$	and
L shoulder	down	$\Delta y < -\lambda$	$F^R = F^L$
L elbow	towards	$\Delta z > \lambda$	together
L hand	away	$\Delta z < -\lambda$	$\Delta(\delta(L,R)) < -\lambda$
L hip	none	$\Delta L < \lambda$	apart
R shoulder		$\Delta R < \lambda$	$\Delta(\delta(L,R)) > \lambda$
R hip			

Table 2: Table listing the locations and hand motions included in the feature vectors. The conditions for motion are shown with the label. Where x, y, z is the position of the hand, either left (L) or right (R), Δ indicates a change from one frame to the next and $\delta(L,R)$ is the Euclidean distance between the left and right hands. λ is the threshold value to reduce noise and increase generalisation, this is set to be a quarter the head height. F^R and F^L are the motion feature vectors relating to the right and left hand respectively.

5. 3D Tracking Based Sub-Units

With the availability of the Kinect™, real-time tracking in 3D is now a realistic option. Due to this, this final sub-unit section expands on the previous tracking sub-units to work in 3D. The tracking is obtained using the OpenNI framework (Ope, 2010) with the PrimeSense tracker (Pri, 2010). Two types of features are extracted, those encoding the *Motion* and *Location* of the sign being performed.

5.1. *Motion* Features

Again, the focus is on linear motion directions, as with the sub-units described in Section 4.1, but this time with the z axis included. Specifically, individual hand motions in the x plane (left and right), the y plane (up and down) and the z plane (towards and away from the signer). This is augmented by the bi-manual classifiers for 'hands move together', 'hands move apart' and 'hands move in sync', again, these are all now assessed in 3D. The approximate size of the head is used as a heuristic to discard ambient motion (that less than 0.25 the head size) and the type of motion occurring is derived directly from deterministic rules on the x,y,z co-ordinates of the hand position. The resulting feature vector is a binary representation of the found linguistic values. The list of 17 motion features extracted is shown in Table 2.

5.2. *Location* Features

Whereas previously, with 2D tracking, a coarse grid is applied, in this section the skeleton returned by the PrimeSense tracker can now be leveraged. This allows signer related locations to be described with higher confidence. As such, the location features are calculated using the distance of the dominant hand from skeletal joints. A feature will fire if the dominant hand is closer than $H^{head}/2$ of the joint in question. A list of the 9 joints considered is shown in Table 2 and displayed to scale in Figure 11. While displayed in 2D, the regions surrounding the joints are actually 3D spheres. When the dominant hand (in this image shown by the smaller red dot) moves into the region around a joint then that feature will fire. In the example shown, it would be difficult for two features to fire at once. When in motion, the left hand and elbow regions may overlap with other body regions meaning that more than one feature fires at a time.

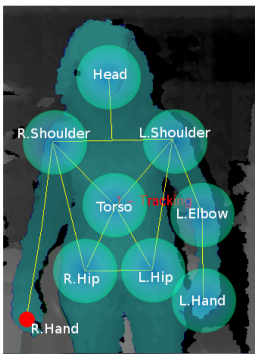


Figure 11: Body joints used to extract sign locations

6. Sign Level classification

Each of the different sub-unit classifier sets is now combined with a sign-level classifier. The groups of binary feature vectors are each concatenated to create a single binary feature vector $F = (f_i)_{i=1}^{D}$ per frame, where $f_i \in \{0,1\}$ and D is the number of dimensions in the feature vector. This feature vector is then used as the input to a sign level classifier for recognition. By using a binary approach, better generalisation is obtained. This requires far less training data than approaches which must generalise over both a continuous input space as well as the variability between signs (e.g., HMMs). Two sign level classification methods are investigated. Firstly, Markov models which use the feature vector as a whole and secondly Sequential Patten Boosting which performs discriminative feature selection.

6.1. Markov Models

HMMs are a proven technology for time series analysis and recognition. While they have been employed for sign recognition, they have issues due to the large training

requirements. Kadir et al. (2004) overcame these issues by instead using a simpler Markov model when the feature space is discrete. The symbolic nature of linguistic sub-units means that the discrete time series of events can be modelled without a hidden layer. To this end a Markov chain is constructed for each sign in a lexicon. An ergodic model is used and a Look Up Table (LUT) employed to maintain as little of the chain as is required. Code entries not contained within the LUT are assigned a nominal probability. This is done to avoid otherwise correct chains being assigned zero probabilities if noise corrupts the input signal. The result is a sparse state transition matrix, $P_\omega(F_t|F_{t-1})$, for each word ω giving a classification bank of Markov chains. During creation of this transition matrix, secondary transitions can be included, where $P_\omega(F_t|F_{t-2})$. This is similar to adding skip transitions to the left-right hidden layer of a HMM which allows deletion errors in the incoming signal. While it could be argued that the linguistic features constitute discrete emission probabilities; the lack of a doubly stochastic process and the fact that the hidden states are determined directly from the observation sequence, separates this from traditional HMMs which cannot be used due to their high training requirements. During classification, the model bank is applied to incoming data in a similar fashion to HMMs. The objective is to calculate the chain which best describes the incoming data, that is, has the highest probability that it produced the observation F. Feature vectors are found in the LUT using an L1 distance on the binary vectors. The probability of a model matching the observation sequence is calculated as

$$P(\omega|s) = \upsilon_w \prod_{t=1}^{l} P_\omega(F_t|F_{t-1}),$$

where l is the length of the word in the test sequence and υ_ω is the prior probability of a chain starting in any one of its states. In this work, without grammar, $\forall\omega, \upsilon_\omega = 1$.

6.2. SP Boosting

One limitation of Markov models is that they encode exact series of transitions over all features rather than relying only on discriminative features. This leads to reliance on user dependant feature combinations which if not replicated in test data, will result in poor recognition performance. Sequential Patterns (SPs), on the other hand, compare the input data for relevant features and ignore the irrelevant features. A SP is a sequence of discriminative *itemsets* (i.e., feature subsets) that occur in positive examples and not negative examples (see Figure 12). We define an itemset T as the dimensions of the feature vector $F = (f_i)_{i=1}^{D}$ that have the value of 1: $T \subset \{1,...,D\}$ is a set of integers where $\forall t \in T, f_t = 1$. Following this, we define a SP $\mathbf{T}$ of length $|\mathbf{T}|$ as: $\mathbf{T} = (T_i)_{i=1}^{|\mathbf{T}|}$, where T_i is an itemset.

In order to use SPs for classification, we first define a method for detecting SPs in an input sequence of feature vectors. To this end, firstly let $\mathbf{T}$ be a SP we wish to detect. Suppose the given feature vector input sequence of $|\mathbf{F}|$ frames is $\mathbf{F} = (F_t)_{t=1}^{|F|}$,

where F_t is the binary feature vector defined in Section 6. We firstly convert $\mathbf{F}$ into the SP $\mathbf{I} = (I_t)_{t=1}^{|\mathbf{F}|}$, where I_t is the itemset of feature vector F_t. We say that the SP $\mathbf{T}$ is present in $\mathbf{I}$ if there exists a sequence $(\beta_i)_{i=1}^{|\mathbf{T}|}$, where $\beta_i < \beta_j$ when $i < j$ and $\forall i = \{1,...,|\mathbf{T}|\}, T_i \subset I_{\beta_i}$. This relationship is denoted with the $\subset_S$ operator, that is, $\mathbf{T} \subset_S \mathbf{I}$. Conversely, if the sequence $(\beta_i)_{i=1}^{|\mathbf{T}|}$ does not exist, we denote it as $\mathbf{T} \not\subset_S \mathbf{I}$.

From this, we can then define a SP weak classifier as follows: Let $\mathbf{T}$ be a given SP and $\mathbf{I}$ be an itemset sequence derived from some input binary vector sequence F. A *SP weak classifier*, $h^{\mathbf{T}}(\mathbf{I})$, can be constructed as follows:

$$h^{\mathbf{T}}(\mathbf{I}) = \begin{cases} 1, & \text{if } \mathbf{T} \subset_S \mathbf{I}, \\ -1, & \text{if } \mathbf{T} \not\subset_S \mathbf{I}. \end{cases}$$

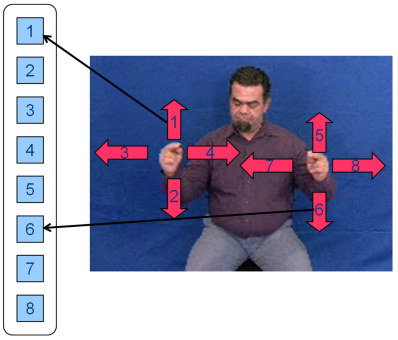
A strong classifier can be constructed by linearly combining a number (S) of selected SP weak classifiers in the form of:

$$H(I) = \sum_{i=1}^{S} \alpha_i h_i^{\mathbf{T}_i}(I).$$

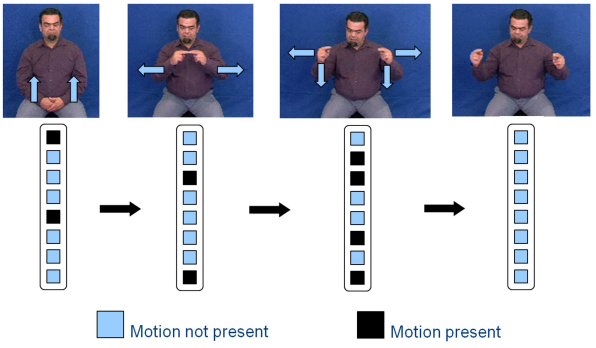
The weak classifiers h_i are selected iteratively based on example weights formed during training. In order to determine the optimal weak classifier at each Boosting iteration, the common approach is to exhaustively consider the entire set of candidate weak classifiers and finally select the best weak classifier (i.e., that with the lowest weighted error). However, finding SP weak classifiers corresponding to optimal SPs this way is not possible due to the immense size of the SP search space. To this end, the method of SP Boosting is employed (Ong and Bowden, 2011). This method poses the learning of discriminative SPs as a tree based search problem. The search is made efficient by employing a set of pruning criteria to find the SPs that provide optimal discrimination between the positive and negative examples. The resulting tree-search method is integrated into a boosting framework; resulting in the SP-Boosting algorithm that combines a set of unique and optimal SPs for a given classification problem. For this work, classifiers are built in a one-vs-one manner and the results aggregated for each sign class.

7. Appearance Based Results

This section of work uses the same 164 sign data set as Kadir et al. (2004) and therefore a direct comparison can be made between their hard coded tracking based system and the learnt sub-unit approach using detection based sub-units. For this work, extra annotation was required as Kadir et al. (2004) used only sign boundaries. 7410 *Location* examples, 322 *Hand-Arrangement* examples and 578 *Motion* were hand labelled for training sub-unit classifiers. The data set consists of 1640 examples (ten of each sign). Signs were chosen randomly rather than picking specific examples which are known to be easy to separate. The sub-unit classifiers are built using only data from four of the ten examples of each sign and the word level classifier is then trained on five examples



(*a*) Feature vector



(*b*) SP

Figure 12: Pictorial description of SPs. (a) shows an example feature vector made up of 2D motions of the hands. In this case the first element shows 'right hand moves up', the second 'right hand moves down' etc. (b) shows a plausible pattern that might be found for the sign 'bridge'. In this sign the hands move up to meet each other, they move apart and then curve down as if drawing a hump-back bridge.

(including the four previously seen by the sub-unit classifiers) leaving five completely unseen examples for testing purposes. The second stage classifier is trained on the previously used four training examples plus one other, giving five training examples per sign. The results are acquired from the five unseen examples of each of the 164 signs. This is done for all six possible combinations of training/test data. Results are shown in Table 3 alongside the results from Kadir et al. (2004). The first three columns show the results of combining each type of appearance sub-unit with the second stage sign classifier. Unsurprisingly, none of the individual types contains sufficient information to be able to accurately separate the data. However, when combined, the appearance based classifiers learnt from the data are comparable to the hard coded classifiers used on perfectly tracked data. The performance drops by only 6.6 Percentage Points (pp), from

79.2% to 72.6% whilst giving the advantage of not needing the high quality tracking system.

Figure 13, visually demonstrates the sub-unit level classifiers being used with the second stage classifier. The output from the sub-unit classifiers are shown on the right hand side in a vector format on a frame by frame basis. It shows the repetition of features for the sign 'Box'. As can be seen there is a pattern in the vector which repeats each time the sign is made. It is this repetition which the second stage classifier is using to detect signs.

	Hand-Arrangement	***Location***	***Motion***	**Combined**	**(Kadir et al., 2004)**
Minimum (%)	31.6	30.7	28.2	68.7	76.1
Maximum (%)	35.0	32.2	30.5	74.3	82.4
Std Dev	0.9	0.4	0.6	1.5	2.1
Mean (%)	33.2	31.7	29.4	72.6	79.2

Table 3: Classification performance of the appearance based two-stage detector. Using the appearance based sub-unit classifiers. Kadir et al. (2004) results are included for comparison purposes.

8. 2D Tracking Results

The data set used for these experiments contains 984 Greek Sign Language (GSL) signs with 5 examples of each performed by a single signer (for a total of 4920 samples). The handshape classifiers are learnt on data from the first 4 examples of each sign. The sign level classifiers are trained on the same 4 examples, the remaining sign of each type is reserved for testing.

Table 4 shows sign level classification results. It is apparent from these results, that out of the independent vectors, the location information is the strongest. This is due to the strong combination of a detailed location feature vector and the temporal information encoded by the Markov chain.

Shown also is the improvement afforded by using the handshape classifiers with a threshold vs a WTA implementation. By allowing the classifiers to return multiple possibilities more of the data about the handshape is captured. Conversely, when none of the classifiers is confident, a 'null' response is permitted which reduces the amount of noise. Using the non-mutually exclusive version of the handshapes in combination with the motion and location, the percentage of signs correctly returned is 68.4%. By

Figure 13: Repetition of the appearance based sub-unit classifier vector. The band down the right hand side of the frame shows the sub-unit level classifier firing patterns for the last 288 frames, the vector for the most recent frame is at the bottom. The previous video during the 288 frames shows four repetitions of the sign 'Box'.

Motion	25.1%
Location	60.5%
HandShape	3.4%
All: WTA	52.7%
All: Thresh	68.4%
All + Skips ($P(F_t\|F_{t-2})$)	**71.4%**

Table 4: Sign level classification results using 2D tracked features and the Markov Models. The first three rows show the results when using the features independently with the Markov chain (The handshapes used are non-mutually exclusive). The next three rows give the results of using all the different feature vectors. Including the improvement gained by allowing the handshapes to be non-mutually exclusive (thresh) versus the WTA option. The final method is the combination of the superior handshapes with the location, motion and the second order skips.

including the 2nd order transitions whilst building the Markov chain there is a 3 pp boost to 71.4%.

This work was developed for use as a sign dictionary, within this context, when queried by a video search, the classification would not return a single response. Instead, like a search engine, it should return a ranked list of possible signs. Ideally the target sign would be close to the top of this list. To this end we show results for 2 possibilities; The percentage of signs which are correctly ranked as the first possible sign (Top 1) and the percentage which are ranked in the top 4 possible signs.

	Markov Chains		SPs	
	Top 1	Top 4	Top 1	Top 4
recall	**71.4%**	**82.3%**	**74.1%**	**89.2%**

Table 5: Comparison of recall results on the 2D tracking data using both Markov chains and SPs

This approach is applied to the best sub-unit features above combined with either the Markov Chains or the SP trees. The results of these tests are shown in Table 5. When using the the same combination of sub-unit features as found to be optimal with the Markov Chains, the SP trees are able to improve on the results by nearly 3 pp, increasing the recognition rate from 71.4% to 74.1%. A further improvement is also found when expanding the search results list, within the top 4 signs the recall rate increases from 82.3% to 89.2%.

9. 3D Tracking Results

While the Kinect™work is intended for use as a live system, quantitative results can be obtained by the standard method of splitting pre-recorded data into training and test sets. The split between test and training data can be done in several ways. This work uses two versions, the first to show results on signer dependent data, as is often used, the second shows performance on unseen signers, a signer independent test.

9.1. Data Sets

Two data sets were captured for training; The first is a data set of 20 GSL signs, randomly chosen and containing both similar and dissimilar signs. This data includes six people performing each sign an average of seven times. The signs were all captured in the same environment with the Kinect™and the signer in approximately the same place for each subject. The second data set is larger and more complex. It contains 40 Deutsche Gebärdensprache - German Sign Language (DGS) signs, chosen to provide a phonetically balanced subset of HamNoSys phonemes. There are 15 participants each performing all the signs 5 times. The data was captured using a mobile system giving varying view points.

9.2. GSL Results

Two variations of tests were performed; firstly the signer dependent version, where one example from each signer was reserved for testing and the remaining examples were used for training. This variation was cross-validated multiple times by selecting different combinations of train and test data. Of more interest for this application however, is signer independent performance. For this reason the second experiment involves reserving data from a subject for testing, then training on the remaining signers. This

	Test	Markov Models Top 1	Markov Models Top 4	SP-Boosting Top 1	SP-Boosting Top 4
Independent	1	56%	80%	72%	91%
	2	61%	79%	80%	98%
	3	30%	45%	67%	89%
	4	55%	86%	77%	95%
	5	58%	75%	78%	98%
	6	63%	83%	80%	98%
	Mean	**54%**	**75%**	**76%**	**95%**
	StdDev	12%	15%	5%	4%
Dependent Mean		**79%**	**92%**	**92%**	**99.90%**

Table 6: Results across the 20 sign GSL data set.

process is repeated across all signers in the data set. The results of both the Markov models and the Sequential Patten Boosting applied to the basic 3D features are shown in Table 6.

As is noted in Section 6.2, while the the Markov models perform well when they have training data which is close to the test data, they are less able to generalise. This is shown by the dependent results being high, average 92% within the top 4, compared to the average independent result which is 17 pp lower at 75%. It is even more noticeable when comparing the highest ranked sign only, which suffers from a drop of 25 pp, going from 79% to 54%. When looking at the individual results of the independent test it can be seen that there are obvious outliers in the data, specifically signer 3 (the only female in the data set), where the recognition rates are markedly lower. This is reflected in statistical analysis which gives high standard deviation across the signers in both the top 1 and top 4 rankings when using the Markov Chains.

When the SP-Boosting is used, again the dependant case produces higher results, gaining nearly 100% when considering the top 4 ranked signs. However, due to the discriminative feature selection process employed; the user independent case does not show such marked degradation, dropping just 4.9 pp within the top 4 signs, going from 99.9% to 95%. When considering the top ranked sign the reduction is more significant at 16 pp, from 92% to 76%, but this is still a significant improvement on the more traditional Markov model. It can also be seen that the variability in results across signers is greatly reduced using SP-Boosting, whilst signer 3 is still the signer with the lowest percentage of signs recognised, the standard deviation across all signs has dropped to 5% for the first ranked signs and is again lower for the top 4 ranked signs.

9.3. DGS Results

The DGS data set offers a more challenging task as there is a wider range of signers and environments. Experiments were run in the same format using the same features as

	Subject Dependent		Subject Independent	
	Top 1	Top 4	Top 1	Top 4
Min	56.7%	90.5%	39.9%	74.9%
Max	64.5%	94.6%	67.9%	92.4%
StdDev	1.9%	1.0%	8.5%	5.2%
Mean	**59.8%**	**91.9%**	**49.4%**	**85.1%**

Table 7: Subject Independent (SI) and Subject Dependent (SD) test results across 40 signs in the DGS data set.

for the GSL data set. Table 7 shows the results of both the dependent and independent tests. As can be seen with the increased number of signs the percentage accuracy for the first returned result is lower than that of the GSL tests at 59.8% for dependent and 49.4% for independent. However the recall rates within the top 4 ranked signs (now only 10% of the data set) are still high at 91.9% for the dependent tests and 85.1% for the independent ones. Again the relatively low standard deviation of 5.2% shows that the SP-Boosting is picking the discriminative features which are able to generalise well to unseen signers.

As can be seen in the confusion matrix (see Figure 14), while most signs are well distinguished, there are some signs which routinely get confused with each other. A good example of this is the three signs 'already', 'Athens' and 'Greece' which share very similar hand motion and location but are distinguishable by handshape which is not currently modelled on this data set.

10. Discussion

Three different approaches to sub-unit feature extraction have been compared in this paper. The first based on appearance only, the latter two on tracking. The advantage of the first approach is that it doesn't depend on high quality tracking for good results. However, it would be easily confused via cluttered backgrounds or short sleeves (often a problem with sign language data sets). The other advantage of the appearance based classification is that it includes information not available by trajectories alone, thus encoding information about handshape within the moment based classifiers. While this may aid classification on small data sets it makes it more difficult to de-couple the handshape from the motion and location sub-units. This affects the generalisation ability of the classifiers due to the differences between signers.

Where 2D tracking is available, the results are superior in general to the appearance based results. This is shown in the work by Kadir et al. (2004), who achieve equivalent results on the same data using tracking trajectories when compared to the appearance based ones presented here. Unfortunately, it is not always possible to accurately track video data and this is why it is still valid to examine appearance based approaches. The

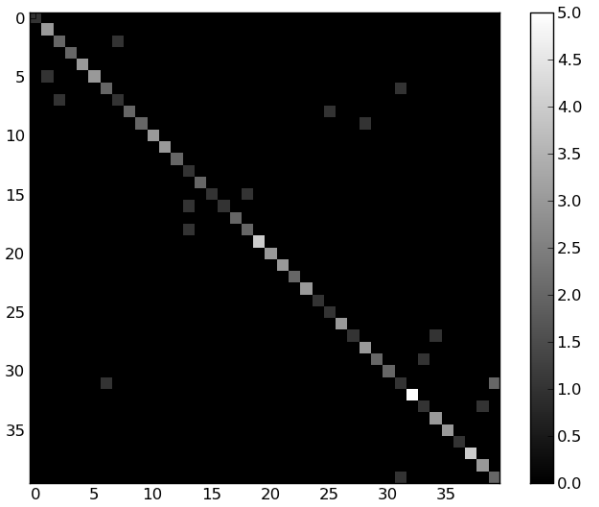


Figure 14: Aggregated confusion matrix of the first returned result for each subject independent test on the DGS data set.

2D tracking *Location* sub-features presented here are based around a grid, while this is effective in localising the motion it is not as desirable as the HamNoSys derived features used in the improved 3D tracking features. The grid suffers from boundary noise as the hands move between cells. This noise causes problems when the features are used in the second stage of classification. With the 3D features this is less obvious due to them being relative to the signer in 3D and therefore the locations are not arbitrarily used by the signer in the same way as the grid is. For example if a signer puts their hands to their shoulders, this will cause multiple cells of the grid to fire and it may not be the same one each time. When using 3D, if the signer puts their hands to their shoulders then the shoulder feature fires. This move from an arbitrary grid to consciously decided body locations reduces boundary effect around significant areas in the signing space.

This in turn leads to the sign level classifiers. The Markov chains are very good at recognising signer dependent, repetitive motion, in these cases they are almost on a par with the SPs. However, they are much less capable of managing signer independent classification as they are unable to distinguish between the signer accents and the signs themselves and therefore over-fit the data. Instead the SPs look for the discriminative features between the examples, ignoring any signer specific features which might confuse the Markov Chains.

11. Conclusions

This work has presented three approaches to sub-unit based sign recognition. Tests were conducted using boosting to learn three types of sub-units based on appearance

features, which are then combined with a second stage classifier to learn word level signs. These appearance based features offer an alternative to costly tracking.

The second approach uses a 2D tracking based set of sub-units combined with some appearance based handshape classifiers. The results show that a combination of these robust, generalising features from tracking and learnt handshape classifiers overcomes the high ambiguity and variability in the data set to achieve excellent recognition performance: achieving a recognition rate of 73% on a large data set of 984 signs.

The third and final approach translates these tracking based sub-units into 3D, this offers user independent, real-time recognition of isolated signs. Using this data a new learning method is introduced, combining the sub-units with SP-Boosting as a discriminative approach. Results are shown on two data sets with the recognition rate reaching 99.9% on a 20 sign multi-user data set and 85.1% on a more challenging and realistic subject independent, 40 sign test set. This demonstrates that true signer independence is possible when more discriminative learning methods are employed. In order to strengthen comparisons within the SLR field the data sets created within this work have been released for use within the community.

12. Future Work

The learnt sub-units show promise and, as shown by the work of Pitsikalis et al. (2011), there are several avenues which can be explored. However, for all of these directions, more linguistically annotated data is required across multiple signers to allow the classifiers to discriminate between the features which are signer specific and those which are independent. In addition, handshapes are a large part of sign, while the work on the multi-signer depth data set has given good results, handshapes should be included in future work using depth cameras. Finally, the recent creation of a larger, multi-signer data set has set the ground work in place for better quantitative analysis. Using this data in the same manner as the DGS40 data set should allow bench-marking of Kinect sign recognition approaches, both for signer dependent and independent recognition. Appearance only techniques can also be verified using the Kinect data set where appropriate as the RGB images are also available though they are not used in this paper. Though it should be noted that this is an especially challenging data set for appearance techniques due to the many varying backgrounds and subjects.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 231135 Dicta-Sign. The Dicta-Sign data sets used and additional SL resources are available via http://www.dictasign.eu/

References

Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588, 1997.

L. Breiman. Random forests. *Machine Learning*, pages 5–32, 2001.

British Deaf Association. *Dictionary of British Sign Language/English*. Faber and Faber, 1992.

P. Buehler, M. Everingham, and A. Zisserman. Learning sign language by watching TV (using weakly aligned subtitles). In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2961 – 2968, Miami, FL, USA, June 20 – 26 2009.

H. Cooper and R. Bowden. Large lexicon detection of sign language. In *Proceedings of the IEEE International Conference on Computer Vision: Workshop Human Computer Interaction*, pages 88 – 97, Rio de Janario, Brazil, October 16 – 19 2007. doi: 10.1007/978-3-540-75773-3_10.

H. Cooper and R. Bowden. Sign language recognition using linguistically derived subunits. In *Proceedings of the Language Resources and Evaluation ConferenceWorkshop on the Representation and Processing of Sign Languages : Corpora and Sign Languages Technologies*, Valetta, Malta, May17 – 23 2010.

R. Elliott, J. Glauert, J. Kennaway, and K. Parsons. D5-2: SiGML Definition. *ViSiCAST Project working document*, 2001.

H. Ershaed, I. Al-Alali, N. Khasawneh, and M. Fraiwan. An arabic sign language computer interface using the xbox kinect. In *Annual Undergraduate Research Conf. on Applied Computing*, May 2011.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the European Conference on Computational Learning Theory*, pages 23 – 37, Barcelona, Spain, March 13 – 15 1995. Springer-Verlag. ISBN 3-540-59119-2.

J.W. Han, G. Awad, and A. Sutherland. Modelling and segmenting subunits for sign language recognition based on hand motion analysis. *Pattern Recognition Letters*, 30(6):623 – 633, April 2009.

T Hanke and C Schmaling. *Sign Language Notation System*. Institute of German Sign Language and Communication of the Deaf, Hamburg, Germany, January 2004. URL http://www.sign-lang.uni-hamburg.de/projects/hamnosys.html.

M. K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, IT-8:179–187, February 1962.

T. Kadir, R. Bowden, E.J. Ong, and A Zisserman. Minimal training, large lexicon, unconstrained sign language recognition. In *Proceedings of the BMVA British Machine Vision Conference*, volume 2, pages 939 – 948, Kingston, UK, September 7 – 9 2004.

S Kim and M.B Waldron. Adaptation of self organizing network for ASL recognition. In *Proceedings of the Annual International Conference of the IEEE Engineering in Engineering in Medicine and Biology Society*, pages 254 – 254, San Diego, California, USA, October 28 – 31 1993.

W.W. Kong and S. Ranganath. Automatic hand trajectory segmentation and phoneme transcription for sign language. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, pages 1 – 6, Amsterdam, The Netherlands, September 17 – 19 2008. doi: 10.1109/AFGR.2008.4813462.

S.K. Liddell and R.E Johnson. American sign language: The phonological base. *Sign Language Studies*, 64:195 – 278, 1989.

K. Lyons, H. Brashear, T. L. Westeyn, J. S. Kim, and T. Starner. Gart: The gesture and activity recognition toolkit. In *Proceedings of the International Conference HCI*, pages 718–727, July 2007.

E. J. Ong and R. Bowden. Learning sequential patterns for lipreading. In *Proceedings of the BMVA British Machine Vision Conference*, Dundee, UK, August 29 – September 10 2011.

OpenNI User Guide. OpenNI organization, November 2010. Last viewed 20-04-2011 18:15.

V. Pitsikalis, S. Theodorakis, C. Vogler, and P. Maragos. Advances in phonetics-based sub-unit modeling for transcription alignment and sign language recognition. In *Proceedings of the International Conference IEEE Computer Society Conference on Computer Vision and Pattern RecognitionWorkshop : Gesture Recognition*, Colorado Springs, CO, USA, June 21 – 23 2011.

Prime Sensor™NITE 1.3 Algorithms notes. PrimeSense Inc., 2010. Last viewed 20-04-2011 18:15.

A. Roussos, S. Theodorakis, V. Pitsikalis, and P. Maragos. Hand tracking and affine shape-appearance handshape sub-units in continuous sign language recognition. In *Proceedings of the International Conference European Conference on Computer VisionWorkshop : SGA*, Heraklion, Crete, September 5 – 11 2010.

J. E. Shoup. Phonological aspects of speech recognition. In Wayne A. Lea, editor, *Trends in Speech Recognition*, pages 125 – 138. Prentice-Hall, Englewood Cliffs, NJ, 1980.

T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden markov models. *Computational Imaging and Vision*, 9:227 – 244, 1997.

W.C Stokoe. Sign language structure: An outline of the visual communication systems of the american deaf. *Studies in Linguistics: Occasional Papers*, 8:3 – 37, 1960.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511 – 518, Kauai, HI, USA, December 2001.

C. Vogler and D Metaxas. Adapting hidden markov models for ASL recognition by using three-dimensional computer vision methods. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, volume 1, pages 156 – 161, Orlando, FL, USA, October 12 – 15 1997.

C. Vogler and D Metaxas. Parallel hidden markov models for american sign language recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, pages 116 – 122, Corfu, Greece, September 21 – 24 1999.

M. B. Waldron and S Kim. Increasing manual sign recognition vocabulary through relabelling. In *Proceedings of the IEEE International Conference on Neural Networks IEEE World Congress on Computational Intelligence*, volume 5, pages 2885 – 2889, Orlando, Florida, USA, June 27 – July 2 1994. doi: 10.1109/ICNN.1994.374689.

M. B. Waldron and S Kim. Isolated ASL sign recognition system for deaf persons. *IEEE Transactions on Rehabilitation Engineering*, 3(3):261 – 271, September 1995. doi: 10.1109/86.413199.

M.B. Waldron and D Simon. Parsing method for signed telecommunication. In *Proceedings of the Annual International Conference of the IEEE Engineering in Engineering in Medicine and Biology Society: Images of the Twenty-First Century*, volume 6, pages 1798 – 1799, Seattle, Washington, USA, November 1989. doi: 10.1109/IEMBS.1989.96461.

H. Wassner. kinect + reseau de neurone = reconnaissance de gestes. http://tinyurl.com/5wbteug, May 2011.

P. Yin, T. Starner, H. Hamilton, I. Essa, and J.M. Rehg. Learning the basic units in american sign language using discriminative segmental feature selection. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4757 – 4760, Taipei, Taiwan, April 19 – 24 2009. doi: 10.1109/ICASSP.2009.4960694.

Z. Zafrulla, H. Brashear, P. Presti, H. Hamilton, and T. Starner. Copycat - center for accessible technology in sign. http://tinyurl.com/3tksn6s, December 2010. URL http://www.youtube.com/watch?v=qFH5rSzmgFE&feature=related.

Z. Zafrulla, H. Brashear, T. Starner, H. Hamilton, and P. Presti. American sign language recognition with the kinect. In *Proceedings of the 13th International Conference*

on Multimodal Interfaces, ICMI '11, pages 279–286, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0641-6. doi: 10.1145/2070481.2070532. URL http://doi.acm.org/10.1145/2070481.2070532.

JMLR: Workshop and Conference Proceedings 14:209–242, 2013 Published 1/13

MAGIC Summoning: Towards Automatic Suggesting and Testing of Gestures With Low Probability of False Positives During Use

Daniel Kyu Hwa Kohlsdorf DKOHL@TZI.DE
Thad E. Starner THAD@CC.GATECH.EDU
GVU & School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30332

Editor: Isabelle Guyon and Vassilis Athitsos

Abstract

Gestures for interfaces should be short, pleasing, intuitive, and easily recognized by a computer. However, it is a challenge for interface designers to create gestures easily distinguishable from users' normal movements. Our tool MAGIC Summoning addresses this problem. Given a specific platform and task, we gather a large database of unlabeled sensor data captured in the environments in which the system will be used (an "Everyday Gesture Library" or EGL). The EGL is quantized and indexed via multi-dimensional Symbolic Aggregate approXimation (SAX) to enable quick searching. MAGIC exploits the SAX representation of the EGL to suggest gestures with a low likelihood of false triggering. Suggested gestures are ordered according to brevity and simplicity, freeing the interface designer to focus on the user experience. Once a gesture is selected, MAGIC can output synthetic examples of the gesture to train a chosen classifier (for example, with a hidden Markov model). If the interface designer suggests his own gesture and provides several examples, MAGIC estimates how accurately that gesture can be recognized and estimates its false positive rate by comparing it against the natural movements in the EGL. We demonstrate MAGIC's effectiveness in gesture selection and helpfulness in creating accurate gesture recognizers.

Keywords: gesture recognition, gesture spotting, false positives, continuous recognition

1. Introduction

The success of the Nintendo Wii, Microsoft Kinect, and Google's and Apple's mobile devices demonstrates the popularity of gesture-based interfaces. Gestural interfaces can be expressive, quick to access, and intuitive (Guimbretière and Winograd, 2000; Pirhonen et al., 2002; Starner et al., 1998; Witt, 2007). Yet gesture-based interfaces may trigger functionality incorrectly, confusing normal movement with a command. For example, the Apple iPod's "shake-to-shuffle" gesture, which is intended to signal when the user wants to skip a song and randomly select another, tends to trigger falsely while

the user is walking (see Figure 1a). Part of the difficulty is that the recognizer must constantly monitor an accelerometer to determine if the gesture is being performed. Some accelerometer or gyro-based interfaces constrain the problem by requiring the user to segment the gesture by pressing a button. For example, in Nintendo's Wii Bowling the player presses the "B" trigger when beginning to swing his arm and releases the trigger at the end of the swing to release the virtual bowling ball. Such a push-to-gesture approach is similar to the push-to-talk method that speech recognition researchers use to improve performance. Yet such mechanisms can slow interactions, confuse users, and limit the utility of gesture interaction. For example, the fast, easy-to-access nature of the shake-to-shuffle gesture would be impeded if the user needed to hold a button to perform the gesture. Ideally, such free-space "motion gestures" (Ashbrook, 2009) should be short, pleasing to perform, intuitive, and easily recognized by a computer against a background of the user's normal movements.

Touchpad gesture shortcuts, which upon execution can start an affiliated application on a laptop or mobile phone (Ouyang and Li, 2012), are another example of command gestures that must be differentiated from everyday motions. Fortunately, these gestures are naturally isolated in time from each other since most touchpad hardware does not even provide data to the operating system when no touches are being sensed. However, an interface designer must still create gesture commands that are not easily confused with normal click or drag and drop actions (see Figure 1b).

Many "direct manipulation" (Hutchins et al., 1985) gestures such as pointing gestures and pinch-to-zoom gestures are used in modern interfaces. These gestures provide the user continuous feedback while the gesture is occurring, which allows the user to adjust to sensing errors or cancel the interaction quickly. However, representational gestures that are intended to trigger a discrete action are less common. We posit that their relative scarcity relates to the difficulty of discovering appropriate gestures for the task. Our previous studies have shown that designing command gestures that do not trigger accidentally during normal, everyday use is difficult for both human computer interaction (HCI) and pattern recognition experts (Ashbrook and Starner, 2010). In addition, the current process to determine the viability of a gesture is challenging and expensive. Gestures are often found to be inappropriate only after the system has entered user testing. If a gesture is found to trigger accidentally during testing, the gesture set has to be changed appropriately, and the testing has to be repeated. Such an iterative design cycle can waste a month or more with each test. Thus, we posit the need for a tool to help designers quickly judge the suitability of a gesture from a pattern recognition perspective while they focus on the user experience aspects of the gestural interface.

Several gesture design tools have been described in the HCI literature (Dannenberg and Amon, 1989; Long, 2001; Fails and Olsen, 2003; Maynes-Aminzade et al., 2007; Dey et al., 2004), yet none address the issue of false positives. Similarly, most gesture recognition toolkits in the pattern recognition and related literature focus on isolated gestures (Wobbrock et al., 2007; Lyons et al., 2007) or the recognition of strings of gestures, such as for sign language (Westeyn et al., 2003). Rarely do such tools focus

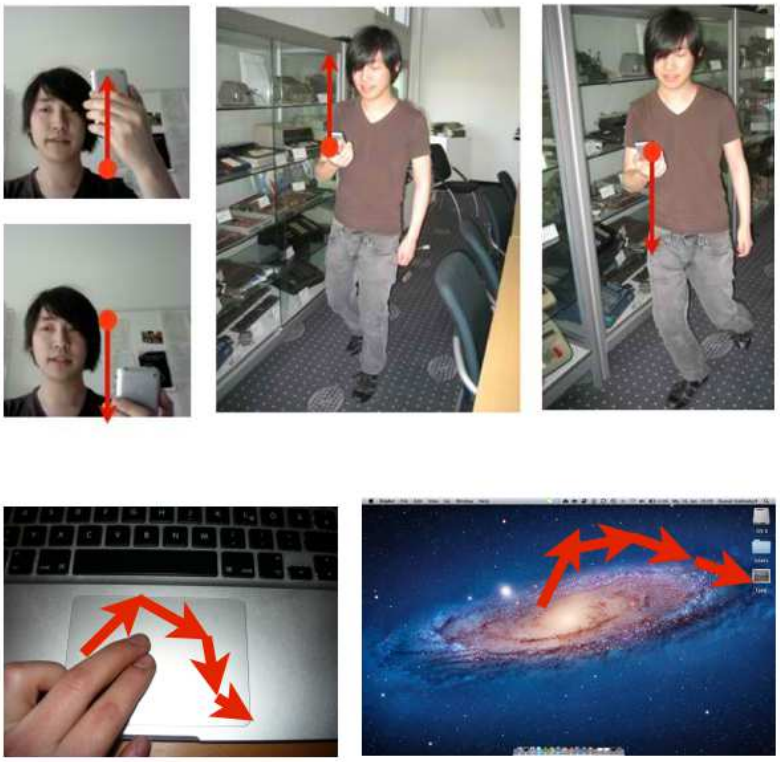


Figure 1: *Top:* A "shake-to-shuffle" gesture (*left*) can be confused with normal up-and-down movement while walking (*right*). *Bottom:* A touchpad shortcut gesture (*left*) can be confused with normal cursor movement (*right*).

on gesture spotting (Yang et al., 2009) for which the critical metric is false positives per hour.

Ashbrook and Starner (2010) introduced the "Multiple Action Gesture Interface Creation" (MAGIC) Toolkit. A MAGIC user could specify gesture classes by providing examples of each gesture. MAGIC provided feedback on each example and each gesture class by visualizing intra- and inter-class distances and estimating the prototype recognizer's accuracy by classifying all provided gesture examples in isolation. Unlike the above tools, MAGIC could predict whether a query gesture would tend to trigger falsely by comparing the gesture to a database of movements recorded in the everyday lives of users. Primarily designed as an HCI Tool, the system used a nearest neighbor method with a dynamic time warping (DTW) distance measure (Fu et al., 2008).

One shortcoming of this work was that the relative false positive rates predicted in user studies were not compared to the actual false positive rates of a gesture recognizer running in the field. Another shortcoming was the long time (up to 20 minutes) needed to search for potential hits in a database of everyday user movements (an "Everyday

Gesture Library" or EGL) even while using approximations like scaling with matching (Fu et al., 2008). MAGIC was designed as an interactive tool, yet due to the delay in feedback, gesture interaction designers waited until all gestures were designed before testing them against the EGL. Often, when doing an EGL test in batch, the interface designers discovered that many of their gestures were poor choices. Designers "learned to fear the EGL." Faster feedback would allow designers to compare candidate gestures to the EGL as they perform each example, speeding the process and allowing more exploration of the space of acceptable gestures. Another result from previous studies is that users were frustrated by encountering too many false positives in the Everyday Gesture Library (Ashbrook and Starner, 2010). In other words, many designed gestures are rejected since the number of predicted false positives is too high.

Here, we focus on the pattern recognition tasks needed to create MAGIC Summoning, a completely new, web-based MAGIC implementation designed to address the needs discovered from using the original. Section 2 introduces the basic operation of the tool. Section 3 describes an indexing method for the EGL using a multi-dimensional implementation of indexable Symbolic Aggregate approXimation (iSAX) that speeds EGL comparisons by an order of magnitude over the DTW implementation. While not as accurate as DTW or other methods such as HMMs, our system's speed allows interface designers to receive feedback after every gesture example input instead of waiting to test the gesture set in batch. We compare the iSAX approach to linear searches of the EGL with HMMs and DTW to show that our approach, while returning fewer matches, does predict the relative suitability of different gestures. Section 4.4 continues this comparison to show that the predictions made by MAGIC match observations made when the resulting gesture recognizers are tested in a real continuous gesture recognition setting. Sections 5 and 6 provide additional details. The first describes a method of using the EGL to create a null (garbage) class that improves the performance of a HMM classifier and a DTW classifier when compared to a typical thresholding method. The second demonstrates the stability of our method by examining its sensitivity to its parameters and provides a method capable of learning reasonable defaults for those parameters in an unsupervised manner. These sections expand significantly upon previous work published in Face and Gesture (Kohlsdorf et al., 2011), while the remaining sections represent unpublished concepts.

Section 7 may be of the most interest to many readers. This section describes how MAGIC Summoning suggests novel gestures that are predicted to have a low probability of false positives. While the capability may be surprising at first, the technique follows directly from the iSAX indexing scheme. In Section 7.2 we show that the suggested gestures have low false positive rates during a user study in a real life setting. In our tests, the space of gestures that are not represented in EGLs tends to be large. Thus, there are many potential gestures from which to choose. Section 7.3 describes our attempts at finding metrics that enable ordering of the suggested gestures with regard to brevity, simplicity, and "quality."

2. MAGIC Summoning Web-based Toolkit

MAGIC Summoning is a web-based toolkit that helps users design motion-based gestural commands (as opposed to static poses) that are expected not to trigger falsely in everyday usage (Kohlsdorf, 2011; Kohlsdorf et al., 2011). All MAGIC experiments described in this paper focus on creating **user-independent** recognizers. This choice reflects our interest in creating useful gesture interfaces and is also due to practicality; collecting large data sets for the EGL from a single user is time consuming and onerous. To ground the discussion with a practical problem, we focus on the challenge of designing gestures performed by moving an Android phone in one's hand. We assume a three-axis accelerometer, which is always included in modern Android phones. The goal is to create gestures (and an appropriate classifier) that, when recognized, trigger functions like "open mailbox" or "next song." Without a push-to-gesture trigger, such gestures are highly susceptible to false positives (Ashbrook, 2009), which emphasizes the need for the MAGIC tool.

2.1. Creating Gesture Classes And Testing For Confusion Between Classes

MAGIC Summoning has two software components: a gesture recorder running on the Android device and the MAGIC web application. The first step in gesture creation is to start a new project in the web service. The interface designer specifies the set of gestures through collecting training data for each of the gestures using the recorder. In order to record a training example, the interaction designer opens the recorder on his smart phone and performs the gesture. The recorder automatically estimates when the gesture starts and when it ends using the method described by Ashbrook (2009). Specifically, the recorder tracks the variance of the accelerometer data in a sliding window. If the variance is above a user-defined threshold, recording starts. If it falls below the threshold, then recording ends.

After the example is recorded, the designer is asked to associate the example with an appropriate gesture label, and the recorder uploads the example to the web. The designer evaluates the gesture in the web application to determine how well it can be distinguished from other gestures. All gestures and their examples are listed in MAGIC Summoning's sidebar (see Figure 2). Examples marked with a red cross are misclassified given the current model, and instances marked with a green circle indicate correct classification. By default, MAGIC Summoning uses a one nearest neighbor classifier with dynamic time warping (NN-DTW) to classify gestures, although other classifiers such as a hidden Markov model (HMM) could be substituted. By clicking on an instance, the designer can see the raw sensor data plotted for that example as well as the predicted number of false positives in the EGL (the method used to calculated this number is explained in Section 3).

Clicking on a gesture in the sidebar opens a view with statistics about it. One statistic is the goodness of the gesture. The goodness is defined as the harmonic mean of precision and recall (Ashbrook, 2009):

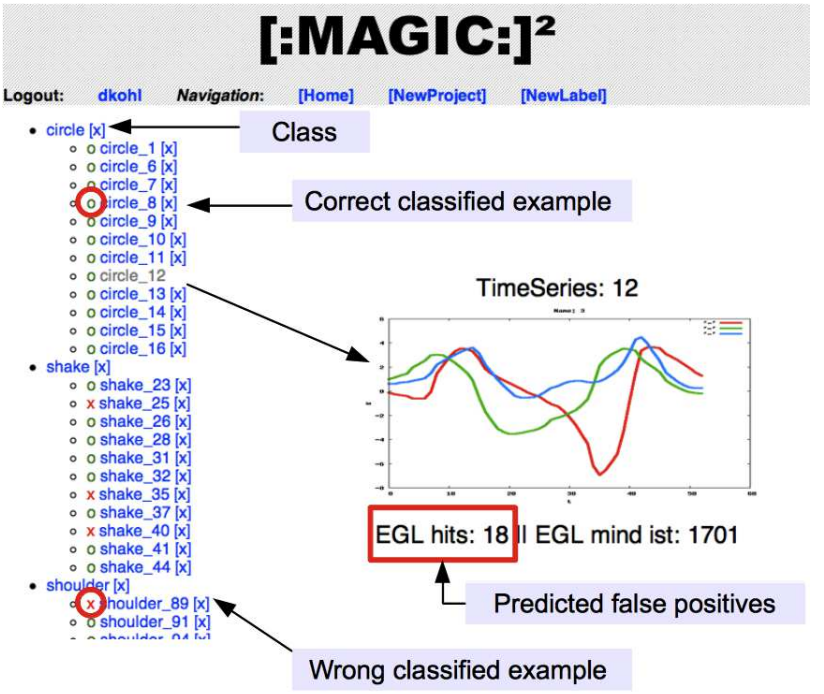


Figure 2: Magic Summoning showing the gesture classes, their examples, and the number of EGL hits (lower numbers are better).

$$goodness = 2 * \frac{precision * recall}{precision + recall}.$$

Similar to the original work by Ashbrook (2009), MAGIC Summoning provides users with information about the inter-class distance and the intra-class distance of the gesture. Both are visualized using a mean and standard deviation plot. In an intra-class distance plot we calculate the means and standard deviations of the distances from all examples in a class to all other examples in that class and visualize the result as a box plot (see Figure 3). In an inter-class distance plot we calculate the means and standard deviations from one class to all the others in the training set. The distance between two classes is the mean distance of all examples of one class to all examples of another. These statistics and visualizations help designers find inconsistencies in the examples of a given gesture class as well as unintentional similarities between classes.

2.2. Android Phone Accelerometer Everyday Gesture Library

We collected a large EGL ($>$ 1.5 million seconds or 19 days total) using six participants' Android phones in Bremen, Germany. The age of the participants ranged from

20 to 30 years. We implemented a background process that wrote the three-axis accelerometer data to the phone's flash memory. Unfortunately, the sampling frequency varied as the models of Android phones we used return samples only when the change in the accelerometer reading exceeds a factory-defined threshold. The phones used are the Motorola Droid, the Samsung Galaxy, HTC Nexus One, the HTC Legend, and the HTC Desire. Other EGLs loadable in MAGIC include movements sensed with a Microsoft Kinect and gestures made on trackpads. We focus mostly on our EGL created with Android phones, but readers interested in experiments with other sensors can refer to Kohlsdorf (2011) for more information.

2.3. Testing For False Positives With The EGL

The original Macintosh-based MAGIC tool displayed a timeline that showed which candidate gesture matched the EGL and at which time. However, gesture designers did not care when or why a given gesture showed a given false positive in the EGL; they just wished to know how many "hits" occurred in the EGL so that they could accept or reject the gesture (Ashbrook, 2009). Thus, we omitted the timeline for simplicity in the web-based application. In the following section we will describe our accelerated method for testing a gesture for potential false positives against the EGL. This method enables rapid iteration on different gesture sets by the interaction designer.

If a user is displeased by the results after testing, he can delete gestures suspected of high false positive rates or misclassification errors and design new gestures. When the user is satisfied with the gesture set, MAGIC Summoning can train a classifier based on hidden Markov models (HMMs) or the default NN-DTW method. The user can then download the trained recognizer. Note that we do not suggest using the iSAX method used to search the EGL as a gesture recognizer as we have tuned the method for speed, not accuracy.

3. False Positive Prediction

When testing a gesture set against the EGL, the original MAGIC calculates the DTW distance for every example of each candidate gesture, sliding a window through time across the EGL and allowing the window to grow or shrink to better match the example when a potential close match is discovered. If the resulting distance is above a certain user-defined threshold it counts as a false positive "hit." Ashbrook and Starner (2010) assert that the sum of the hits predicts how well the gesture will perform in everyday life (an assertion supported by our experiments described later).

In optimizing the speed of the EGL comparison, Ashbrook (2009) observed that not all regions of the EGL need checking. Since we are interested in motion-based gestures instead of static poses, parts of the EGL with low variance in their signal need not be examined. Thus, we pre-process the EGL to find "interesting" regions where the average variance over all dimensions in the sensor data in a region defined by a sliding

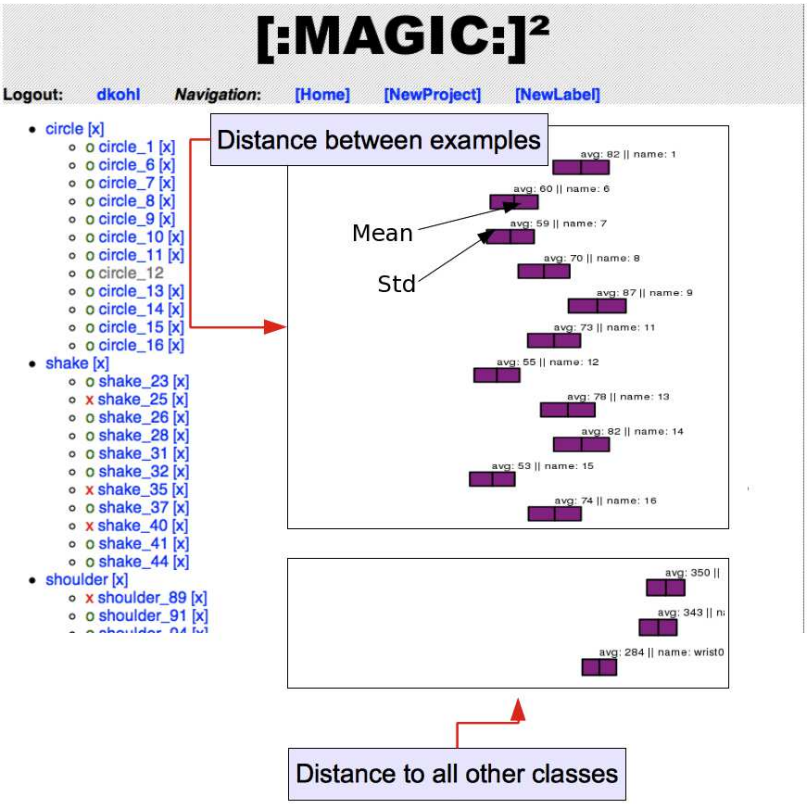


Figure 3: Mean and standard deviation of the distance between each example in a class and of the class as a whole in relation to other classes.

window over 10 samples exceeds a given threshold (see Figure 4).[1] Eliminating regions from the EGL that can not possibly match candidate gestures significantly speeds EGL search. Note that a similar technique was described earlier to segment gestures when the interface designer is creating examples of candidate gestures. All experiments in this paper will use these techniques.

Searching the EGL parallelizes well, as each processor can be devoted to different regions of the EGL. However, even on a high-end, eight-core Macintosh workstation, searches were too slow for an interactive system. For a small, five-hour EGL with three-axis accelerometer data sampled at 40Hz, each example required between 5-25 seconds to check. Thus, one gesture with 10 examples could require minutes to search in the EGL. This slowness causes interface designers to create gestures in batch and then check them against the EGL. Testing a set of eight gestures with all their examples could take up to 20 minutes, leading to a relatively long and frustrating development cycle for the designer (Ashbrook and Starner, 2010). In the following sections, we

1. Word spotting algorithms in speech recognition perform similar checks, rejecting regions of "silence" before employing more computationally intensive comparisons.

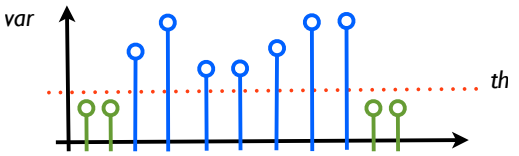


Figure 4: When finding start and stop points of a gesture or finding interesting regions in the EGL, we run a sliding window over the raw recorded time series and calculate the sample variance in that window when a new sample is inserted. If the variance is above a certain threshold, the gesture or interesting region starts. It stops when the variance falls below that threshold.

describe a method to speed the EGL search using iSAX. We start with an overview of our method and our assumptions. We then provide the specific methods we used to adapt iSAX to our problem.

3.1. Overview Of EGL Search Method And Assumptions

In MAGIC Summoning, we first segment the EGL into interesting regions as defined previously. Each region is divided into four even subregions to form a "word" of length four. The region is then encoded into a string of symbols using the standard SAX quantization method. The string is entered into an iSAX tree representing the EGL. The iSAX tree is initialized with cardinality two but quickly grows as many regions hash to the same leaf on the suffix tree and the leaf needs to be split (Shieh and Keogh, 2008). As each region is encoded into the iSAX tree, its location in the EGL is recorded in the leaf. Once the EGL is completely encoded into an iSAX tree, we can perform "approximate search" using a gesture example as a query (Shieh and Keogh, 2008). The query is split into four regions and SAX-encoded in much the same way as the interesting regions of the EGL. An approximate search to determine the number of matches between the query and the EGL becomes a simple matter of matching the query string to the appropriate branch of the iSAX suffix tree and returning the number of strings contained in that branch.

One failing of this approach is that the interesting regions may be significantly larger or smaller than the candidate gestures. Regions significantly smaller than the command gestures are not of concern as they will never falsely match a command gesture in practice. We can eliminate such regions out-of-hand from the comparison. However, regions of movement that might match the query gesture may be hidden within longer regions in the EGL.

A key insight, which will be used repeatedly, is that we need not recover every region of the EGL that might cause a false match with the query. We are not intending iSAX to be used as a gesture recognizer. Instead, our goal is to allow the designer to compare the suitability of a gesture relative to other candidates quickly. As long as the movement occurs repeatedly in the EGL at isolated times as well as in longer regions,

the iSAX method will report a number of "hits," which will be sufficient to warn the interaction designer of a problem.

A second insight is that users of gesture interfaces often pause before and after they perform a command gesture. Gesture recognizers exploit this behavior and use these pauses to help identify the command gesture. Movements that look like command gestures embedded in long regions of user motion are unlikely to be matched in practice by these recognizers. However, short everyday user motions that are similar to a command gesture are a particular worry for false positives. Thus, the iSAX encoding scheme of the EGL above seems suitable for our needs. However, if the goal of the interaction designer is to create gestures that can be chained together to issue a series of commands quickly, these longer regions in the EGL will need to be encoded more formally using constraints on how long a section can be encoded in each symbol. Such constraints can be derived from the length of expected command gestures (usually between 1-4 seconds in our experience), and the length of SAX word defined by the system.

A final insight is that a more precise comparison against the EGL can be made at the end of the gesture design process with the gesture recognizer that is output by MAGIC. During gesture design, all we require of the EGL search method is that it is fast enough to be interactive and that it provides an early warning when a given gesture may be susceptible to false triggering. Given the above operational scenario, we tune our iSAX implementation to provide fast feedback to the user. Details on the implementation follow below.

3.2. SAX Encoding

SAX quantizes time series in both time and value and encodes them into a string of symbols (Lin et al., 2007). For example, the time series in Figure 5 is divided into four equal portions (for a "word" length of four) and converted into a string using a four symbol vocabulary (a "cardinality" of four).

To be more precise, we first normalize the time series to have a zero mean and standard deviation of one. Assuming the original time series $T = t_1, ..., t_j, ...t_n$ has n samples, we want to first quantize the time series into a shorter time series $\bar{T} = \bar{t}_1, ..., \bar{t}_i, ...\bar{t}_w$ of word length w. The i^{th} element of $\bar{T}$ can be calculated by

$$\bar{t}_i = \frac{w}{n} \sum_{k=(\frac{n}{w}(i-1)+1)}^{\frac{n}{w}i} t_k.$$

Given the values in the compressed time series, we next convert them into symbols using a small alphabet of size (cardinality) a. Imagine the y-axis divided into an arbitrary number of regions bounded by $a-1$ breakpoints. Each of these regions is assigned to a symbol from the alphabet. Since we wish each symbol in the vocabulary to be used approximately the same amount, we place a normal Gaussian curve centered at 0 on the y-axis and place the breakpoints such that the area under the Gaussian for each section is equal. By performing the SAX process on the EGL and each gesture

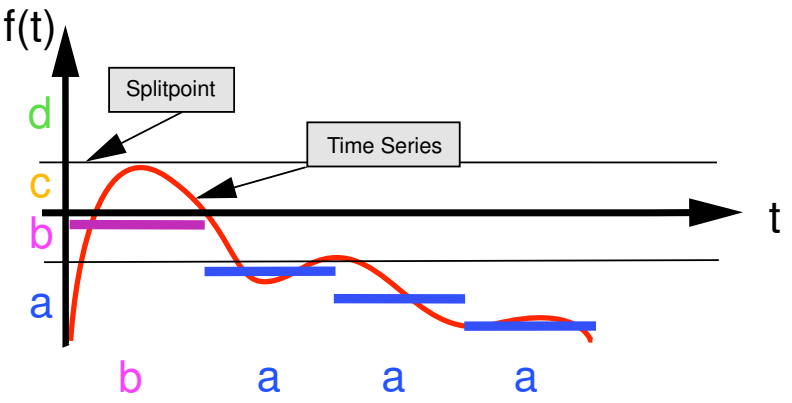


Figure 5: SAX process used to convert a time series to a string. The raw data is segmented into a user-specified word length, in this case four. Then each segment is replaced by a symbol associated with that region on the y-axis, based on the average value. The resulting string is represented by the string of symbols with superscripts indicating the number of symbols used to quantize each region: $b^4a^4a^4a^4$.

example separately, we are able to compare the changes in the signals through time without concern regarding their offsets from zero or relative amplitudes.

One convenience of the SAX representation is that there exists a distance calculation between two strings, defined as MINDIST by Lin et al. (2007), that is a lower bound on the Euclidean distance between the original two time series. Thus, we can search the EGL for possible false positives with some measure of confidence.

Another convenience of the representation is that the cardinality of each separate region can be increased whenever more precision is needed. For example, suppose we increase the cardinality of the first region in Figure 5 to eight (thus, the vocabulary would include letters a-h). The string might then be $d^8a^4a^4a^4$, as the region of the y-axis formerly covered by symbols a and b would now be covered by symbols a, b, c, and d. We can compare strings with regions of different cardinality by observing that we know that each time series is normalized before SAX encoding and that the regions are defined by a normal Gaussian centered at zero with all regions having an equal area under the Gaussian's curve. Thus, we still know the minimal distance possible between each region, and we can still use MINDIST to determine a lower bound on the Euclidean distance between the original two time series. This capability will be useful in our upcoming discussion on iSAX and its application to the EGL.

3.3. Multi-Dimensional iSAX Indexing And EGL Search

iSAX is a tree-based method for time series indexing introduced in Shieh and Keogh (2008). For encoding the EGL, our goal is create an iSAX tree that can be traversed quickly when searching for a match to a SAX-encoded example of a gesture. Each leaf of the tree contains the number of occurrences of that string in the EGL as well as the position of each occurrence. To begin, assume we are searching an EGL represented by the simple iSAX tree in Figure 6 with a query represented by $a^2a^2b^2b^2$ (for the sake of argument, assume we decided to represent the example gesture crudely, with regions of cardinality two). Immediately, we see that there is no branch of the tree with an a^2 in the first position, and we return no matches in the EGL. Now assume that we are searching the EGL for a query of $b^2b^2b^2b^2$. We find that there is a node of the EGL that contains that string, and that node has children (that is, the node is an "internal node"). Looking at the children in that branch, we see that we need to re-code the query gesture to have cardinality three in the first region. Re-coding reveals that the query gesture is better represented by the sequence $c^3b^2b^2b^2$, which matches one of the terminal leaves in the tree. The number of sequences from the EGL stored in that leaf is returned as the number of "hits" in the EGL.

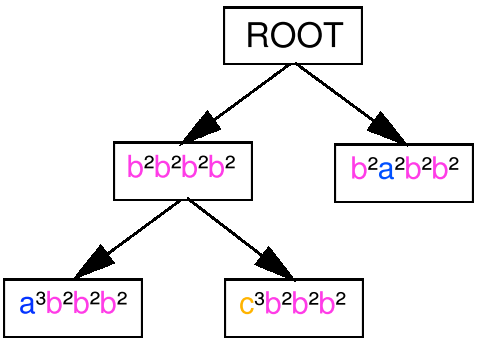


Figure 6: iSAX tree with three leaves. On the first level all symbols' cardinalities are equal. The node $b^2b^2b^2b^2$ is an internal node. For the children under this node, the cardinality of the first region is increased by one.

Next we describe how to encode a one-dimensional EGL into an iSAX tree. First, we find all the "interesting" regions in the EGL using the variance method discussed earlier. We divide the regions evenly into four sections and encode them using SAX with cardinality two, allowing for sixteen possible strings. Note that each node in an iSAX tree holds a hash table mapping child nodes to an iSAX word. Thus, when inserting a region into the iSAX tree, we compare the region's SAX string to the hash table in the root node. If there is no match, we create a child node and enter it into the hash table using its SAX string. If the SAX string is found, we examine the node to

see if it is a terminal leaf. Each leaf points to a file (called a "bucket") stored on disk holding all of the regions that have mapped to it. The leaf also contains the position of each of the regions in the EGL and a count of the number of regions contained in the leaf. If the number of regions in the bucket exceeds a user specified size (called the "bucket size"), it is deleted, and the cardinality of the iSAX word is increased at one position (picked by round robin). At the deleted node's position we insert a new internal node. All the time series of the deleted node are inserted into the new node but with a higher cardinality. Children of the internal node are created as needed, effectively splitting the previous leaf into several new leaves. When we encounter a internal node during the insertion of a region, we search the node's hash table for children that match and proceed normally, creating a new leaf node if no matching child exists.

Note that this method of creating the iSAX tree dynamically adjusts the size of the vocabulary to better distinguish similar regions in the EGL. Given a bigger vocabulary, the SAX word will fit more exactly to the region. In other words, this method of encoding devotes more bits to describing similar movements that are repeated often in the EGL. Thus, when a query gesture is compared to the EGL iSAX tree, MAGIC will quickly return with no or few hits (depending on the specified bucket size) if the query is very distinct from the EGL. If the query is similar to motions in the EGL, the search process will traverse deeper in the tree, examining finer and finer distinctions between the query and the regions contained in the EGL.

The above discussion assumed that the data was one-dimensional. For multi-dimensional data, such as is used in the experiments described below, we create n iSAX trees, one for each dimension of the recorded data. We index all dimensions separately and join those n trees under one new root node (see Figure 7).

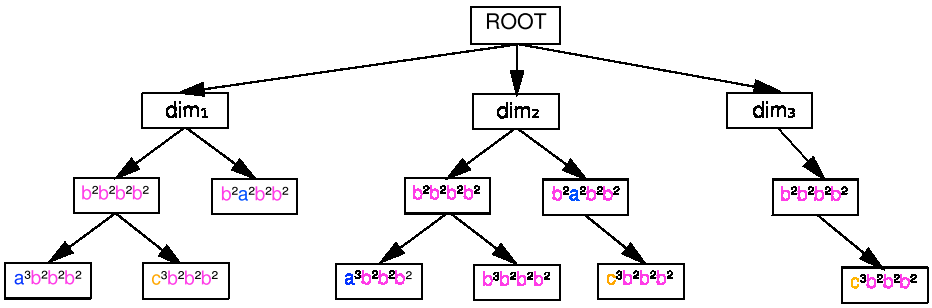


Figure 7: A multi-dimensional iSAX tree. Under the root node there is a dimension layer. Each node in this layer is the root node for a one-dimensional iSAX tree. During search, we search all iSAX trees, one for each dimension.

We query the EGL iSAX tree (constructed from the EGL) in all n dimensions. The result of that search is n files, one for each dimension. The number of hits can then be calculated by counting the number of places where each hit from each dimension overlap for all dimensions. Comparing the timestamps can be costly, so we introduced

```
Test preparation:
0) Collect a large data base of user movements in advance.
1) Find interesting regions by applying variance thresholding.
2) Build an n dimensional iSAX tree.

Gesture testing:
0) Find start and end point of gesture.
1) Search the iSAX tree in all n dimensions.
2) Return the number of time series in the minimum file.
```

Table 1: Testing gestures for potential false positives against a database of pre-recorded device usage.

an approximation based on the observation that there can never be more overlapping time series than the number in the dimension with the lowest number of matches. For example, consider the result of a search in three dimensions (x, y, z) where the number of hits in the EGL are $x = 4$, $y = 20$ and $z = 6$. There can never be more then four hits total if we require that hits must overlap in all dimensions. The overall EGL testing method is summarized in Table 1.

Upon reflection, the EGL search procedure described above raises several questions and possibilities. What are reasonable values for the bucket size, word size, and cardinalities used in encoding the EGL, and how sensitive is MAGIC to these parameters? This question will be examined in detail in Section 6. A nice side effect of EGL search is that we can use the matches found to train a class of gestures that a recognizer should ignore (a "garbage" or NULL class). Section 5 will explore this option. Searching for which SAX strings are not contained in the EGL tree can suggest which gestures are not made during everyday movement. In Section 7, we exploit this attribute to recommend gestures to the interaction designer. However, first we will provide evidence that searching the EGL does indeed predict the number of false positives during the usage of a gesture interface.

4. Experimental Verification

In the following section we describe two experiments that suggest that an iSAX search of the EGL is a viable means to predict false positives. Our first goal is to show that false positive prediction using iSAX is correlated with the previous method of searching the EGL linearly using dynamic time warping (Ashbrook, 2009). We will also conduct an experiment in which we will show that the EGL is able to predict the relative number of false positives when using a gesture interface in everyday life. We describe the data used for the experiments and our experimental method before presenting our findings.

4.1. EGLs And Gestures Used In Evaluations

We use three different data sets to serve as EGL databases. The first is our Android accelerometer data set as described earlier. Before indexing the recorded data, we extracted the interesting regions, applying a threshold of $th = 0.001$ (triggering at almost any movement) and a window size of $N = 10$ (0.25 sec at 40Hz). The average duration of the interesting regions is $11,696ms$. The second EGL is based on the Alkan database[2] of everyday movements collected with an iPhone (Hattori et al., 2011). The third data set is another collection of everyday movements collected on Android phones for a different project at Georgia Tech. These two latter EGLs were processed in the same manner as the first.

We collected a reference data set of gestures for evaluation purposes. We acted as interaction designers and designed four gestures by performing them while holding a smart phone. For each gesture we collected 10 examples, resulting in 40 examples total. The four gestures are: drawing a circle in the air, touching your shoulder, shaking the phone up and down, and hacking (a motion similar to swinging an ax). The average duration of the gestures is between one and two seconds.

4.2. Comparison Conditions: NN-DTW And HMMs

When comparing the dynamic time warping EGL search method to a search in iSAX index space we will use the following procedure. The DTW method compares each interesting region from the EGL to each gesture example (Ashbrook, 2009). We calculate the dynamic time warping distance of a new gesture to all examples in the EGL and apply a threshold chosen empirically. All regions for which the distance is below this threshold for any example count as a false positive (in keeping with MAGIC's ability to output a one nearest neighbor classifier for live gesture recognition).

For yet another comparison, we use hidden Markov models to search the EGL for false positives. For the experiments in this paper, we use a six-state HMM (ignoring initial and end states) with one skip transition and one Gaussian output probability per state per dimension (see Figure 8). We collect all the examples for our gesture set first and then train a HMM for each of the gestures. We classify each region in the EGL and apply a threshold based on maximum likelihood to determine if a region in the EGL is close enough to the gesture to count as a false positive. We chose both the maximum likelihood threshold as well as the distance threshold so that classifier accuracy stayed high (93% for NN-DTW and 100% for HMM).

4.3. Comparison Of iSAX To NN-DTW And HMM In Searching EGLs

We wish to compare our iSAX EGL search method to the more conventional NN-DTW and HMM techniques described above. When selecting between two candidate gestures, the interaction designer wishes to choose the one with a lower number of predicted false positives. Thus, if a first gesture has few hits when NN-DTW or HMMs are

2. Alkan web site can be found at: http://alkan.jp/.

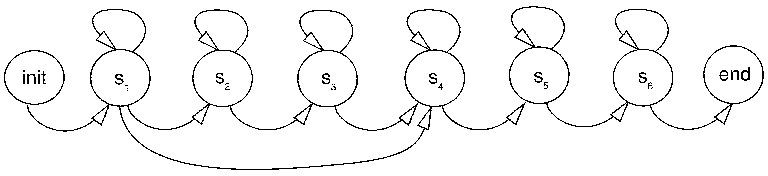


Figure 8: The topology of the left-right, six-state HMM used in our experiments. The first state is the start state, and the eighth state is the end state. Each internal state transitions to itself and its successor. We include a skip transition to help recognize shorter gestures.

used and a second gesture has many hits, that same trend should be shown with iSAX. The absolute number of EGL hits does not matter, but there should be a strong correlation between the relative number of hits returned by iSAX and the other two techniques when run on the same set of gestures. We use the Pearson correlation coefficient as a metric to compare the techniques.

Regardless of the search method used, we store the number of hits in a vector. Each entry of that vector corresponds to the overall number of false positives for a given gesture. For iSAX and NN-DTW, the overall number of false positives for a gesture is calculated by searching the EGL for each example of that gesture and summing the resulting numbers of hits. For HMM models, thresholding on the log likelihood probability is used. For our set of four test gestures, testing returns three vectors (one for each method) of four elements (one for each gesture). We calculate the Pearson correlation coefficient between the iSAX vector and the NN-DTW vector and between the iSAX vector and the HMM vector.

To reassure ourselves that this technique produces a meaningful metric, we performed Monte Carlo simulation experiments. Indeed, the correlation of random vectors with four elements show low r values.

First, we compare the search methods on the EGL from Bremen. We chose the iSAX parameters empirically:

word length: 4
base cardinality: 2
bucket: 6000.

Figure 9 compares the number of hits per hour returned by each method. The hits per hour metric reflects the number of matches found in the EGL divided by the original time required to record the EGL. One can see that our iSAX search approximation returns many fewer hits than NN-DTW or HMMs. However, the magnitude of the iSAX values correlate strongly with the NN-DTW ($r = 0.96$) and HMM ($r = 0.97$) results. Thus, a high number of hits returned by iSAX on the EGL (high compared to other gestures tested with iSAX) is a good indicator for when a gesture should be discarded. The remaining gestures are suitable candidates for user testing.

We also measured the time needed to complete the search for each method on a 2.0GHz Intel Core Duo T2500 Macbook with 2GB of RAM. The NN-DTW and HMM methods require more then 10 minutes to complete the search on all 40 gesture examples whereas iSAX search required 22 seconds, a 27X increase in speed. With such speed, each of the gesture examples could have been checked as it was entered by the interaction designer. In fact, the EGL search would require less than a second for each gesture example, which is less than the amount of time required to check a new example for confusion against all the other gesture examples with NN-DTW when creating a eight gesture interface (Ashbrook, 2009). Thus, we have obtained our goal of maintaining interactivity during gesture design.

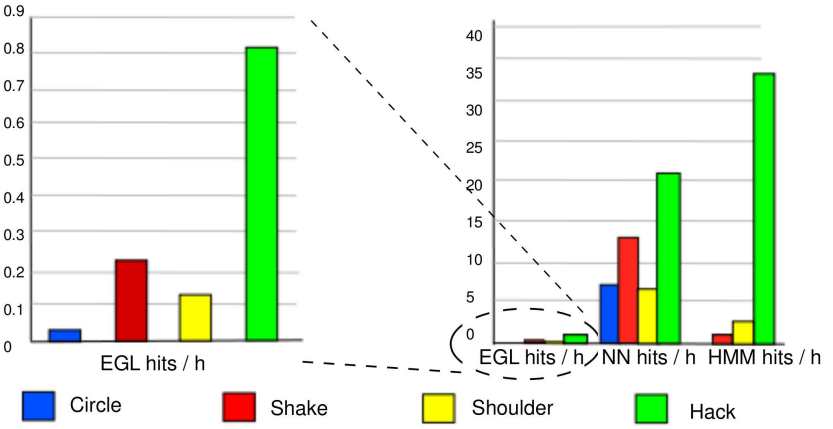


Figure 9: *Left:* The hits per hour in the EGL based on iSAX search. *Right:* A comparison of the number of hits per hour returned by iSAX, NN-DTW, and HMMs from the EGL.

We were curious as to how much EGL data is needed to predict poor command gestures. We generated three random subsets of the EGL by picking 100, 200 and 500 interesting regions at random from the data set and comparing the correlation coefficient between iSAX and NN-DTW. The correlation between the results remained surprisingly high, even with an EGL containing only 100 regions:

- **n = 100:** $r = 0.89$
- **n = 200:** $r = 0.93$
- **n = 500:** $r = 0.93$.

As later experiments show, more data is better, but even a relatively small EGL can help the interaction designer avoid choosing troublesome gestures. We also compared iSAX versus NN-DTW in the Alkan and Georgia Tech EGLs, with similar results to the original Bremen EGL:

- **Alkan:** $r = 0.94$
- **Georgia Tech:** $r = 0.99$.

Our results suggest that the results of an iSAX search on the EGL correlate highly with those of the slower EGL search methods. Even though the absolute number of hits found by the iSAX method are significantly fewer than the other methods, the relative number of hits can be used to compare the desirability of one candidate gesture versus another.

4.4. Comparison Of iSAX Predictions To HMM And NN-DTW Gesture Recognizer Use In Practice

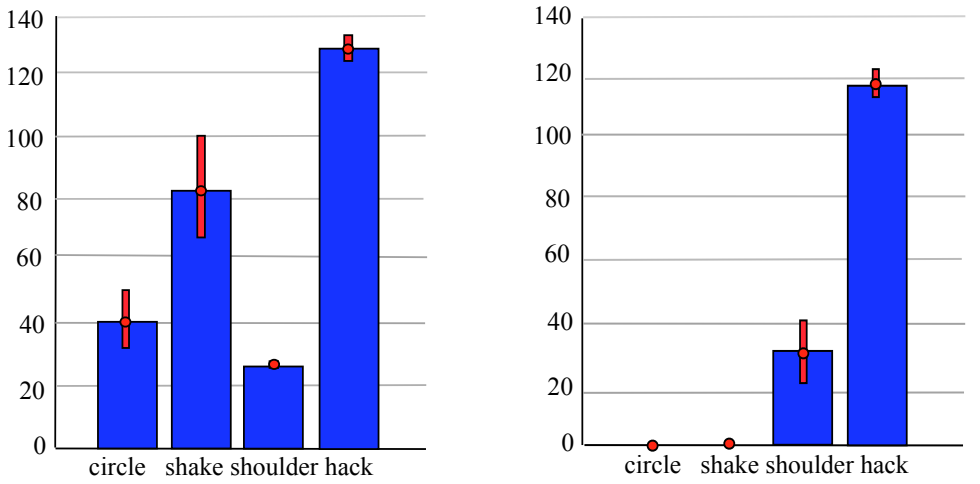


Figure 10: The EGL hits per hour found during deployment. *Left:* The EGL hits for NN-DTW search per gesture. *Right:* The EGL hits for HMM search per gesture. The EGL hits for a gesture are the average hits over all four users. The bars correspond to one standard deviation.

Next, we examine whether our iSAX EGL search method is able to predict false positives in everyday life. In fact, this experiment is the first to verify that any EGL search is able to predict false positive rates of a gesture recognizer in practice.

We exported NN-DTW and HMM recognizers from MAGIC Summoning for the four gestures trained during the process described in the previous experiment. We integrated the HMM classifier into an interactive system. Next, we recruited four Android phone users who had not contributed to the EGLs nor the training of the gestures.

In order to understand how difficult it was to perform the gestures correctly, we asked the users to perform each gesture 10 times without feedback. The HMM classifier performed at 60% accuracy, which is not surprising given the gestures and testing procedure. Next we allowed the users to train with the HMM recognizer to become more familiar with how to perform the gestures so that they could be more easily recognized. This way of learning can be found in commercial systems like the Nintendo Wii, which uses avatars to help users learn control gestures. Not surprisingly, the four users' average accuracy with the HMM recognizer improved to 95% after training.

After the users completed their training, we installed a software application on their phones that notified the users when to perform one randomly selected gesture, once every hour. Otherwise, the users performed their normal activities, and the application records all the users' movements. We searched the recorded data for the intended gestures. The HMM classifier found $50\% - 70\%$ of the intentional gestures whereas NN-DTW search found all of them. However, the NN-DTW classifier had lower precision than the HMMs. Given that we specifically allowed gestures that were known to be poor (from EGL testing) and that the system did not provide feedback to the users, such poor performance is to be expected (and desired from the point of the experiment).

Figure 10 shows the false positive rates for each gesture and recognizer. We observed a high correlation ($r = 0.84$) between the relative false positive rates predicted by the iSAX search on the original EGL and the actual, tested NN-DTW performance on the users' data. The correlation was even higher ($r = 0.97$) for the HMM classifier. These results support our hypothesis that MAGIC Summoning can be used to predict gestures at risk of having many false positives when deployed in gesture recognizers in practice.

5. Improving Recognition Through A NULL Class Created From EGL Search

In the experiments in the previous section, we needed to specify a threshold to avoid false positives when distinguishing the four gestures from our four users' everyday motions. For NN-DTW, the threshold was a distance, while with HMMs it was a probability. Setting this threshold requires more pattern recognition experience than an interaction designer may possess, and often gestures are not separable from everyday movements with a simple threshold. Another option is to create a NULL (garbage) class, which attempts to capture all the motion not matching the gestures of interest. With this technique, the recognizer runs continually but does not return a result when the sensor data matches the NULL class.

Here, we use EGL data to train a NULL class automatically so that a user-defined threshold is not needed. Multi-dimensional iSAX search of the EGL returns time series similar to a query gesture. Thus, it is a simple matter to collect the EGL hits from all examples of all gestures in the gesture interface to train a NULL gesture (using either technique).

The following experiment is based on the data collected while our four users performed the four requested gestures during their daily activities. We adjusted the thresholds upward for the HMM and NN-DTW recognizers to avoid misclassifications in the EGL while still detecting the gestures from the training set. We also trained NULL classes for both recognizers. Figure 11 shows the results of all four recognizers running on the user study data. Using the EGL NULL class method resulted in a statistically significant improvement of both the NN-DTW ($p << 0.0001$) and HMM ($p < 0.05$) recognizers. Both avoided more false positives using the NULL class instead of a threshold. Gesture recognition accuracy and correlation to the iSAX EGL hits remained consistent with the experiment in the previous section. The results suggest that training a NULL class based on EGL hits can be a successful way to improve performance and reduce complexity for the interaction designer. Note that many variations of this technique are possible and might further improve results. For example, a different NULL class could be trained for each gesture.

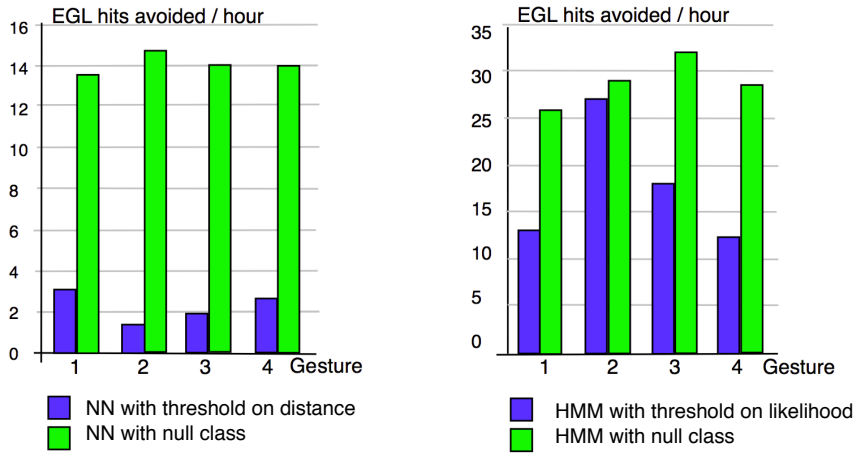


Figure 11: *Left:* The false positives per hour avoided using a NULL class for each gesture based on EGL hits versus the simple threshold. *Right:* The false positives per hour avoided for HMMs using the NULL class versus the simple threshold.

6. iSAX Parameter Sensitivity Experiments

In Section 4.4, iSAX was able to predict the relative performance of a gesture during continuous recognition. However, the process required setting several parameters: word length, bucket size, and initial cardinality. In addition, we compared the false positive predictions to that of the NN-DTW method, which itself required a distance threshold

(when a NULL class is not used). How sensitive is our method to these parameters? We use the same four test gestures (circle, shake, shoulder, hack) and EGL as in our user study to explore this issue.

Observe that the cardinality of the sequences is automatically adjusted during the creation of the EGL iSAX tree, quickly changing from its initial minimal setting of two. Effectively, this parameter is not set by the user, and we can remove it from the following experiments on parameter sensitivity by holding it at a reasonable value. We choose a base cardinality of four ($card = 4$), given that this level of complexity was judged sufficient from observations in the original iSAX experiments (Shieh, 2010; Shieh and Keogh, 2008) and in our own work (Kohlsdorf et al., 2011).

In the experiments below, we compare the iSAX results, while varying the bucket size and word length, to the NN-DTW method using the correlation method described above. We also tried comparing the iSAX method to NN-DTW with different reasonable distance thresholds (3.3, 5, 10), but we found little change in the results. For example, the bottom of Figure 12 shows graphs comparing iSAX word length to correlation with NN-DTW at each of the distance thresholds. The graphs are very similar, indicating that the comparison with NN-DTW is relatively stable with respect to the distance threshold used. Thus, we turn our attention to word length and bucket size.

In the first part of the experiment we test the correlation of the EGL searches using NN-DTW and iSAX trees constructed with different iSAX word lengths (4,5,6, ... ,13) and bucket sizes (1000, 2000, ... , 10000). Figure 12 plots the results. Changes in bucket size cause minor variations in correlation; however, word length has significant effects.

Since the performance of our method seems mostly dependent on one parameter, we propose an automatic parameter tuning method that does not require any data except a pre-recorded EGL. The central concept is to choose random regions from the EGL to serve as a gesture training set and to tune the iSAX parameters to that set using hill climbing.

We require the user to specify the number of gestures in the data set (N), how many examples we want to collect for each gesture (M), and a threshold on the dynamic time warping distance over which two time series are distinct. We pick N regions of motion ("interesting" regions) at random from the EGL to serve as "reference gestures." For those N reference gestures we extract M examples from the EGL where the DTW distance to the reference gesture is smaller than a threshold. Then we compute the false positives for this gesture set using the NN-DTW method. In order to find the appropriate word length we use hill climbing in the iSAX parameter space. At each step, we perform false positive prediction using iSAX and compare the results to the NN-DTW results using the Pearson correlation coefficient as an objective function.

We ran an experiment to test this hill-climbing technique, allowing the procedure to set the word length automatically and comparing the results to NN-DTW. We started the word length at 4 and increased it to 13. If the observed correlation at a given word length is followed by a smaller one when the next word length is tried, the algorithm stops and returns the last word length. As one can see in Figure 12, after 3 iterations iSAX finds

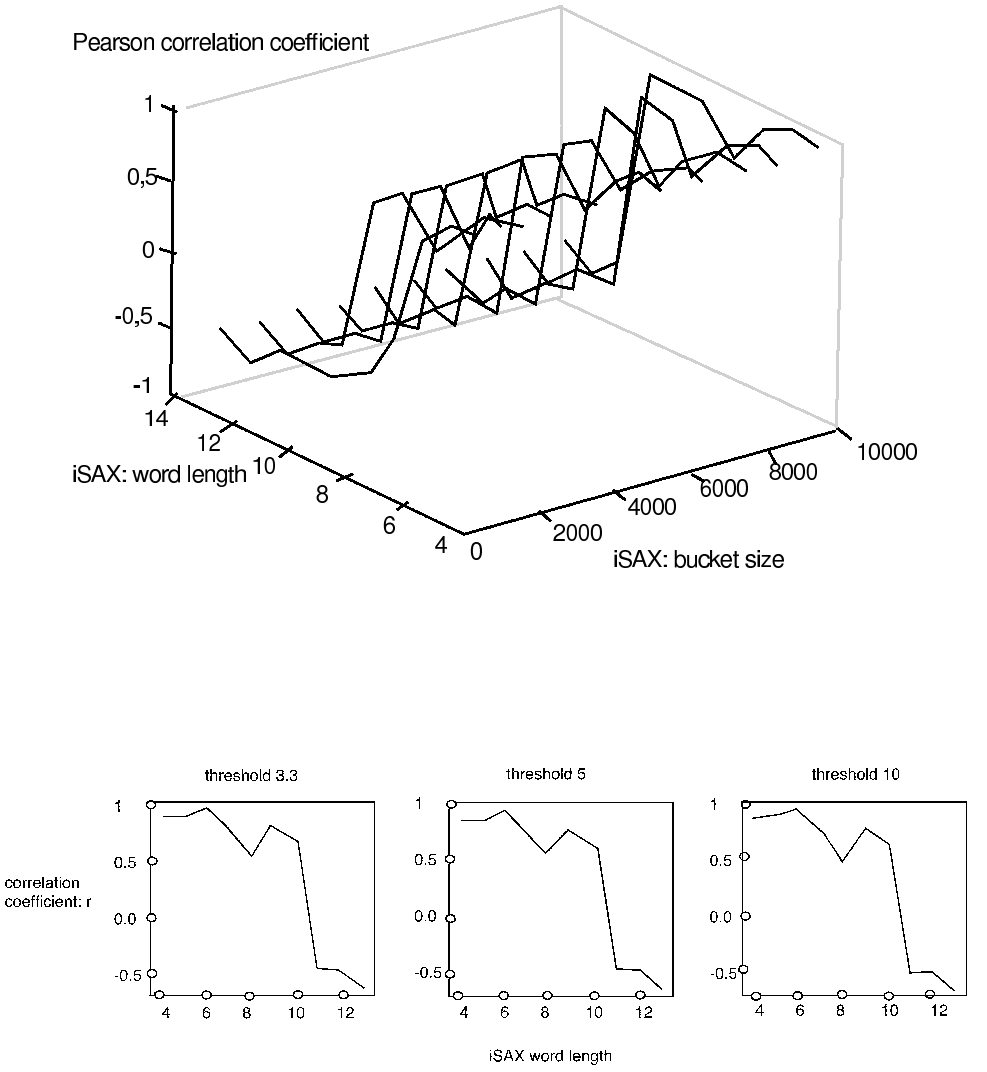


Figure 12: Top: correlation to NN-DTW vs. iSAX word length vs iSAX bucket size. Bottom: iSAX word length vs. correlation to NN-DTW for distance thresholds of 3.3, 5, and 10, respectively.

a local maximum. However, this sequential method is not optimal. For example, if the word length which maximizes the correlation is 9 and the local maximum at the word length 6 is smaller, we would stop too early. However, this problem can be solved by including simulated annealing or stochastic gradient descent in the future.

In this chapter, we showed that the iSAX EGL search relies on several parameters but that the parameters can be tuned automatically. Word length seems the primary parameter that needs to be tuned.

7. MAGIC Summoning: Suggesting Gestures With Low Probability Of False Positives During Use

To this point, we have focused on efficient gesture testing. However, when using MAGIC to design gestures in previous studies, our participants wished to have MAGIC suggest potential gestures instead of creating their own. Often the gestures designed by the participants showed high false positive rates when tested against the EGL, leading to frustration. MAGIC users said they would rather select from a set of gestures that were "known good" than experiment blindly with constraints they did not understand (Ashbrook, 2009).

In the next section, we describe a method for suggesting gestures based on a pre-recorded EGL. We then perform an experiment where we test suggested gestures for false positives during normal device usage by naive subjects. Finally, we examine different possible metrics to order the suggestions for easier selection by the designer. While we have mostly used accelerometers in our experiments to date, here we concentrate on capacitive trackpads, specifically those used on Apple's laptops. Data from inertial sensors are hard to visualize for an interaction designer without a inverse kinematic system to map the sensor readings into limb movement. While such systems are now feasible with adequate accuracy, we wished to avoid the additional complexity for these first experiments. Trackpads provide two dimensional data that are easy to visualize for an interaction designer, and trackpads are commonly used in everyday office work. In addition, industry has begun to include more complex command gestures in their trackpad-based products (Li, 2010).

7.1. Synthesizing And Visualizing Gestures

We introduce a method for proposing gestures that do not collide with every day movements using four steps, briefly outlined here. First, we collect an EGL that is representative of the usage of the device or sensor. Next, we build an iSAX tree based on the EGL. We systematically enumerate the possible SAX strings and check for those which are NOT contained in the tree. Finally, we visualize these gestures and present them to the interaction designer. Once the designer selects a set of gestures for his interface, MAGIC Summoning can train a recognizer for the gestures using synthesized data.

7.1.1. COLLECTING AN EGL

Collecting a representative EGL is often time-consuming and is best done by someone familiar both with the specific sensor involved and pattern recognition in general. Fortunately, the process is only necessary once for the device of interest and then can be used for different interface designers and tasks. Mostly, the EGL will be collected across multiple people to ensure that the resulting gestures can be user independent. Ideally, the EGL should be collected across every situation and physical context where the device might be used (for example, sitting at a desk or driving) to make sure that incidental motions are well represented. If the resulting gesture recognizer is intended to

work across different devices (for example, across multiple version of Android phones), the EGL should be collected from a representative sample of those devices.

7.1.2. REPRESENTING THE EGL AND GENERATING GESTURES

Next, we convert the EGL into a simplified iSAX tree structure. Unlike the work above, here we only care that a given string occurred in the EGL instead of how many times it occurred. Thus, we can use a simpler indexing method that will allow easier gesture building later. We convert interesting regions from the EGL to SAX words and build the set of all strings observed in the EGL. Since the sensor input is multivariate, we build the SAX word in each dimension and concatenate the words. Thus, for n dimensions and a word length of w, the indexing key grows to $n * w$. Given the cardinalities in the word, discovering gestures that are not represented in the EGL is a simple matter of combinatorics. We generate all possible gestures and store the gesture as a viable candidate if it is not contained in the EGL.

7.1.3. VISUALIZING CANDIDATE GESTURES AND TRAINING GESTURE RECOGNIZERS

In order for the interface designer to select between the different candidate gestures, we must visualize them. Specifically, we need to convert the candidate gesture from a SAX string into a real valued time series. For each SAX symbol, we know that valid values are somewhere between the upper and lower breakpoint of the area assigned to the symbol. We choose a random point between these breakpoints for each symbol. We then use spline interpolation or re-sampling to fit a curve through the resulting values from each SAX symbol. We used an exponential moving average to smooth the resulting curve. The overall process is shown in Figure 13. Note that by repeating this process we can generate a synthetic set of time series that could have generated the SAX word. This synthetic data is used to visualize acceptable versions of the trackpad gesture to the interaction designer. We will also use this synthetic data to train a recognizer for the gesture if it is selected (see below).

Figure 14 shows MAGIC Summoning's user interface for gesture suggestion. In the center of the window we display a synthesized gesture. The color of the lines indicates the time course of the gesture as it is performed on the trackpad (from dark to light). Many synthetic examples of a given SAX word are drawn to give the interaction designer a sense of the possible shapes of the gesture. New suggestions are displayed periodically, effectively creating a movie of potential gestures. In our first implementation, gesture suggestions were selected randomly, keeping a list of previously viewed gestures so as to avoid repetition. If the interaction designer sees a desirable gesture, he stops the presentation with a key press.

If other gestures have already been selected by the user, the similarity of the currently displayed gesture to the already selected gestures is shown in a bar plot in a window at the bottom left. Based on these similarity scores, the user can retain the gesture or continue searching other suggestions. In this case, we decided to use the $1

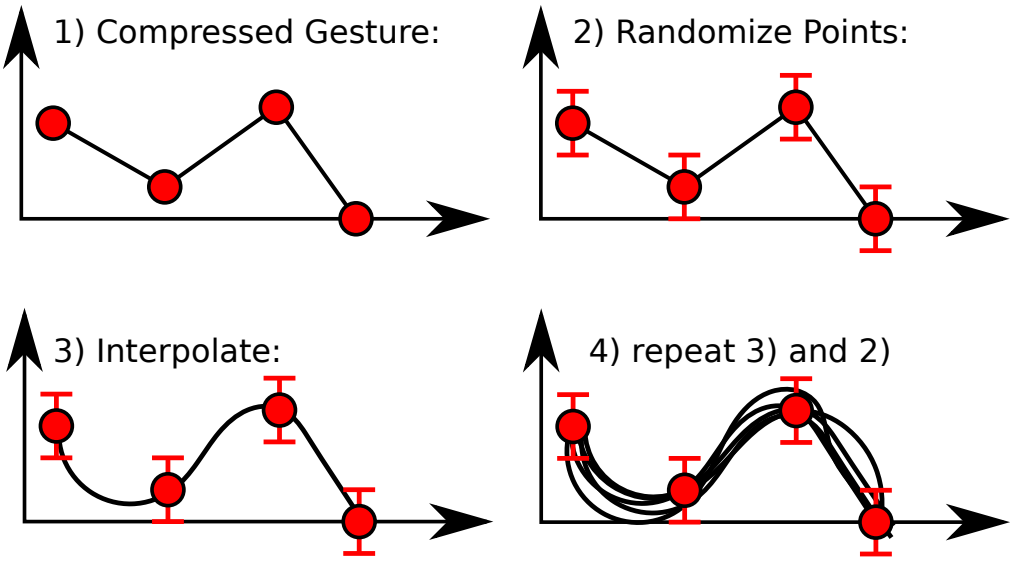


Figure 13: Converting a SAX word to example gestures.

Figure 14: The MAGIC Summoning gesture suggestion interface.

Recognizer (Wobbrock et al., 2007) both for generating similarity scores and for gesture recognition. To train the gesture recognizer, we simply used the synthetic examples generated during the visualization process.

7.1.4. $1 RECOGNIZER

Since the $1 Recognizer is widely used in HCI research (Belatar and Coldefy, 2010; Dang and André, 2010) but is not necessarily known to machine learning researchers,

we give a quick overview here. The recognizer is optimized for single stroke gestures and can be considered instance-based learning. Each instance or template is re-sampled to be of equal length with all others and then rotated, scaled, and translated to a canonical form before being used. During recognition the query gesture is compared to all the stored templates using an angular distance metric. In continuous recognition we can apply a threshold on that distance, and the rest of the recognition process is similar to the dynamic time warping approach described earlier. The authors report recognition accuracies of 99%, which is comparable to DTW implementations on the same data sets. The method is simple, fast to compute, and understandable by pattern recognition novices. Thus, the algorithm is well-suited for experimentation by interface designers. With MAGIC Summoning, interaction designers do not need to collect any training data for the recognizer. The training data is produced synthetically from the EGL as described above. Note that we can use the $1 Recognizer as a distance measure for EGL search (albeit slowly compared to iSAX), which will be useful for comparison experiments below.

7.2. Testing Suggested Gestures And Recognizers In Practice

We collected an EGL consisting of ten participants using their personal Mac laptops for one week. Figure 15 visualizes the EGL. While indexing the EGL, we set the SAX word length to four. For a two dimensional touchpad, the length doubles to eight. Setting the cardinality to four leads to a total number of 65536 (4^8) possible strings.

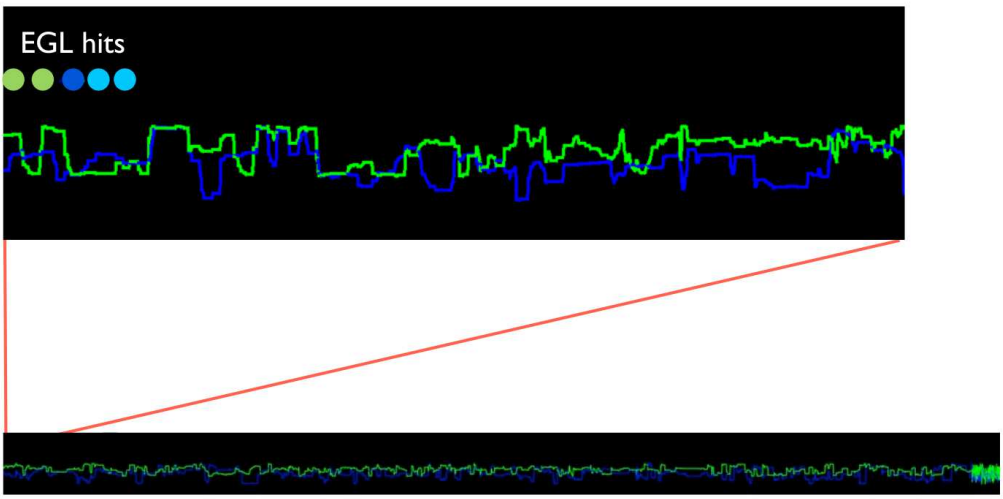


Figure 15: Bottom: The touch pad EGL. Top: An excerpt from the EGL showing five false positives during testing of a gesture, indicated as colored bubbles.

We observed 1222 unique strings in the collected EGL. The space is surprisingly sparse; there are 64314 strings not found in the EGL, suggesting that there are a large number of gestures that could be made with a low probability of false positives.

We performed an experiment to evaluate if the proposed suggestion and selection process described in the previous section can produce gestures that show a low false positive rate in everyday life. In addition, we were concerned as to whether synthetic data would be sufficient to train a high accuracy recognizer for this domain. We acted as an interaction designer and selected six gestures using the visualization tool above (see Figure 16). We preferred gestures that were simple and memorable. Figure 17 demonstrates 70 other gestures suggested by the system that were not used. We trained a $1 Recognizer for each of the six gestures selected using synthetic data generated by MAGIC.

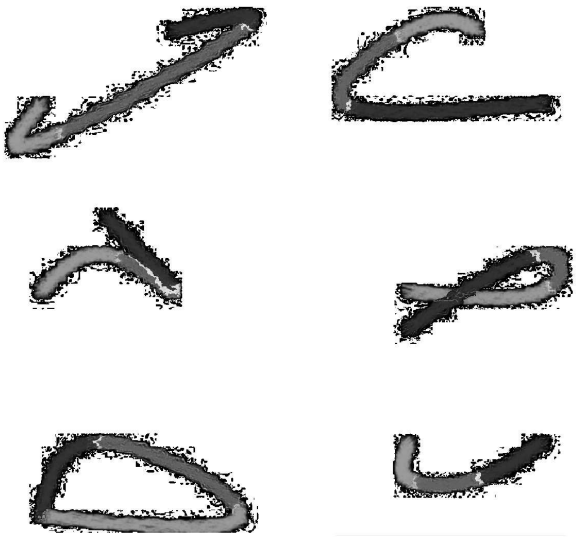


Figure 16: The six gestures used in the study. Gestures are drawn from dark to light.

We designed a six user study with users who did not contribute to the EGL. As in the false positive prediction experiments from the previous section, we asked users to practice with the recognition system so that they could perform the gestures with confidence. Users were able to improve their performance from $\approx 46\%$ to $\approx 90\%$ quickly. Afterward, the users worked on their computers for four hours while all touchpad movements were recorded. Every 10 minutes we sent a notification to the users asking them to perform one of the six gestures, resulting in four examples of each gesture for each participant. Thus, we collected 24 hours of data and 144 gesture examples.

The gesture recognizer was able to recognize 98% of the performed gestures. Even though synthetic data was use to train the recognizer, these findings are similar to those of Wobbrock et al. (2007), who reported a 99% accuracy in their experiments. The false positive rates of the gestures are low except for one gesture (see Figure 18). Thus,

Figure 17: 70 generated gestures with potential low false positive rates. Gestures ordered from left-to-right and from top-to-bottom with increasing entropy.

the experiment supports the hypothesis that MAGIC Summoning can suggest gestures and aid the interaction designer in creating a gesture system that results in low false positives. However, several questions remain. Can we order the suggestions so as to present the "best" gestures first? Also, the experiment as described has no control condition. What would have been the result if we had tried suggesting random gestures from the 64,314 available?

7.3. Ordering Gesture Suggestions

In this section we will explore possible ways of ordering gestures such that users can quickly find desirable gestures from the large number of possibilities. Our intuition is that users prefer simple gestures since they can be accessed quickly and are easy to memorize.

Our first approach is defining the complexity of a gesture as the entropy of its SAX word (Mitchell, 1997):

$$H(word) = -\sum_{i=0}^{card} p(symbol_i) * log(symbol_i).$$

However, if we want to prefer simpler gestures, we should check to determine if false positive rates in real usage are correlated with simplicity. Otherwise, proposing simpler gestures first could be counterproductive. Intuitively, one would think that

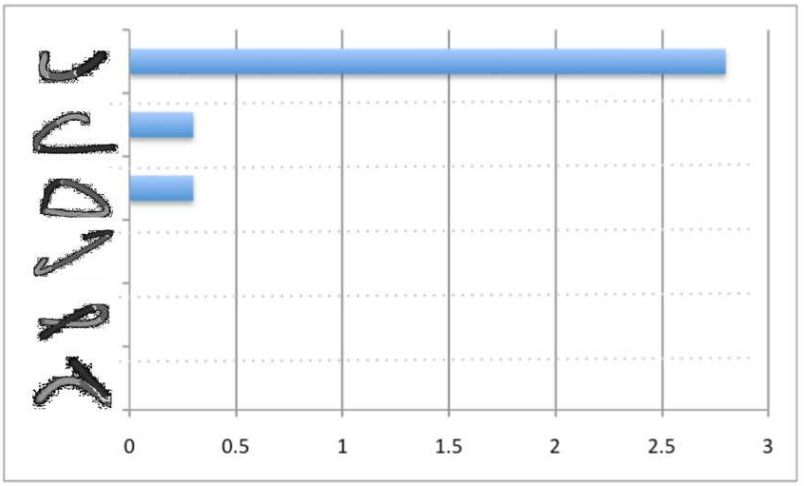


Figure 18: Results of the trackpad gesture user study in false positives per hour. All but one of the gestures suggested by MAGIC Summoning show a low false positive rate.

simpler gestures would trigger more often in everyday life. To investigate this question we trained the $1 Recognizer with 100 randomly chosen gestures and searched the EGL with it. For each gesture we calculated the entropy and compared the false positive rate to the entropy and found no correlation ($r^2 \approx 0.04$). Thus, there seems to be little additional risk to suggesting lower entropy gestures first.

The above heuristic seems logical for ordering suggestions. Low entropy gestures would seem to be simpler and easier to perform. To confirm this intuition we ran a small user study. We generated 100 gestures and sorted them using the above score. We examined the 20 best-ranked gestures and rejected ones that required significant overlap of the strokes (see Figure 19) as the static visualization of the strokes could confuse subjects. For each of the 10 remaining gestures we asked six users to perform the gesture in the air, on the table or on their touchpad and asked them to assign a score of performability between 1 and 10. All participants received the same gestures. Interestingly, we were not able to find a correlation between the entropy of a gesture's SAX word and the users' ratings ($r^2 = 0.09$).

Given the above result, we desire gestures not in the EGL but that are known to be performable. With a trackpad, all suggested gestures should be physically possible, but in future work with inertial sensors the suggestions could become impossible without constraining the system in some manner.

We decided to prefer gesture suggestions where the substrings of the SAX word representing the candidate gesture are represented in the EGL, but the gesture string itself was not present. We will assume one dimension for ease of illustration. If a gesture ACBD is not in the EGL, but the subcomponents AC, CB, and BD or ACB

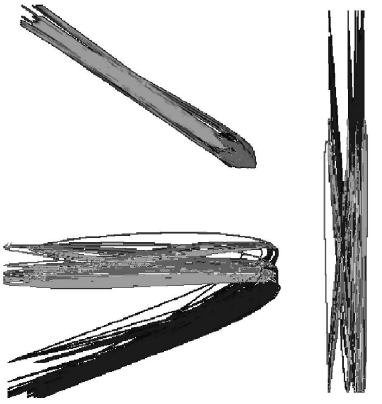


Figure 19: MAGIC Summoning gestures with significant overlap of the strokes were rejected to avoid user confusion.

and CBD were well represented in the EGL, we might conclude that ACBD is possible for the user to perform. In other words, we will prefer gestures where the most n-grams from the EGL are included in the suggested gesture's string. Intuitively, though, such a heuristic causes concern that such gestures might have a higher chance of false triggering.

To investigate this possibility, we extracted bi-grams and tri-grams from the EGL, created candidate gestures from them, and tried to find a correlation between the false positives in the EGL and the number of n-grams in the gesture's string. Note that this method of composition creates gestures with a variety of properties: ones common in the EGL, rare in the EGL, and not present in the EGL. A correlation would indicate an increased risk with this method of ordering the suggestions, but we did not find one, giving a modicum of assurance in the method:

Bi-grams $r^2 = 0.000676$

Tri-grams $r^2 = 0.000256$.

Beside low false positives, another criteria for a good gesture system is that there should be a low chance of confusion between gestures. If the user is creating a control system with six gestures and has already selected five of them, we should prefer suggestions that are distinct from the five gestures already chosen. We measure the distinguishably of a gesture using the Hamming distance (Hamming, 1950) of the gesture's SAX word. Thus, when ordering gestures, we sort using a score defined as

$$score(word) = \frac{dist(word)}{(1 + entropy(word))}$$

where the distance of the word is the average Hamming distance to all other gestures in the gesture set. This metric provides a high distance to the other gestures and a low

entropy. Note that we use $(1 + entropy(word))$ to avoid unreasonably high or infinite scores when the entropy value is near 0.

Given the results of the above experiments, we are now tuning MAGIC Summoning to generate gestures composed from parts of the EGL and to suggest gestures that are most dissimilar to each other. We intend to test this ordering system in our future work with suggesting gestures for use with inertial sensors.

7.4. How Selective Are MAGIC Summoning's Suggestions?

In the above user study, we selected six gestures by hand from MAGIC Summoning's suggestions and tested the $1 Recognizer that MAGIC output for both accuracy and false triggering. However, there were many possible gestures that the system could have output instead. In this last section we will investigate if suggesting gestures based on our method is better generated ones by chance.

As we have seen previously, using iSAX results in fewer hits being identified in an EGL than those found by typical gesture recognizers (HMM, NN-DTW, $1 Recognizer, etc.). The sole reason to use iSAX is that it quickly returns whether or not a candidate gesture is worthwhile to investigate further. However, we do not need to generate gesture suggestions in real time. In fact, as soon as an EGL is collected, the same "overnight" process that generates the EGL's iSAX tree representation for prediction could track the gestures not represented in the EGL. Once these gestures are known, the recognizer of choice could be trained with synthetic data of the gesture, and the recognizer could be run on the EGL for a more precise estimate of the expected hits. The number of false positives returned should allow a finer discrimination between candidate gestures. In the following experiment, we use this new procedure to generate suggested gestures and test ones with the lowest number of false positives on the test data collected from subjects not represented in the EGL.

In this experiment, we generated 2000 random gestures from SAX strings not in the EGL. For each of the gestures we synthesized 40 examples and trained a $1 recognizer with them. We used this recognizer to test search the EGL in the classic way, that is testing each interesting region using the trained recognizer. We used a typical threshold ($th = .85$) for the $1 score. All results above that threshold count as a hit with the EGL. Figure 20 orders the gestures by least to most number of hits per hour in the EGL. Clearly the $1 Recognizer identifies many potential false positives, yet most of the gestures still have low rates.

Figure 21, top, shows another view of this data. Note that over 35% of the 2000 gestures have 0 - 0.0001 false positives/hour. Compare this rate to that of Figure 21, bottom. This graph was generated using all the SAX strings represented in the EGL. Less than 12% of these gestures have such low false positive rates. Clearly, the SAX representation does have considerable predictive power on which suggested gestures are least likely to trigger falsely using the $1 Recognizer in the EGL. In fact, better than one in three of the gestures suggested by choosing SAX strings not in the EGL will be candidates for very low false positive rates with the synthetically trained $1 Recognizer.

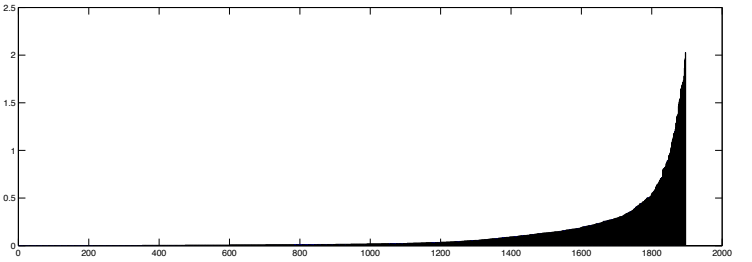


Figure 20: Number of false positives identified in the EGL using the $1 Recognizer for each of 2000 gestures synthesized from SAX strings not represented in the EGL. Gestures with more than 2 hits per hour are not graphed to preserve scale.

The above observation suggests a relatively efficient method for creating gesture candidates for the interaction designer. First, randomly choose a unrepresented SAX string in the EGL. Train the desired recognizer using synthetic data. Run the recognizer on the EGL. If the rate of false positives per hour is less than 0.0001, keep the gesture. Otherwise, discard it. Generate as many gesture suggestions as is possible given time constraints. (Approximately 25 minutes is required to generate 100 gesture suggestions using a modern laptop, but such a process is highly parallelizable and can be run in batch before the interaction designer approaches the system.) Order the suggestions as described above and present them to the interaction designer for selection.

We conducted an experiment evaluating this algorithm. We split the collected EGL for touchpad gestures into two subsets. Each subset contains randomly chosen, distinct time series from the original EGL. The intersection between the subsets is empty. We used the first subset to generate 100 randomly chosen, distinct gestures candidates that show less then 0.0001 false positives per hour using the $1 Recognizer. We used these recognizers to then search the data in the second subset. On average we found the gestures to trigger 0.0022 times per hour, with a standard deviation of 0.003. These rates correspond to an average time between false triggerings of 455 hours, or approximately one month assuming usage 16 hours/day. Thus, this method of choosing gestures to suggest to an interaction designer seems desirable as well as practical.

8. Future Work

To date, the task for most gesture recognition systems has been to optimize accuracy given a set of gestures to be recognized. In this paper, we have reversed the problem, seeking to discover which gestures might be most suitable for recognition.

However, improved suggestion ordering is an area for improvement. Performability might be improved by modeling how gestures are produced (Cao and Zhai, 2007) and prioritizing those gestures with least perceived effort. For domains where the coupling

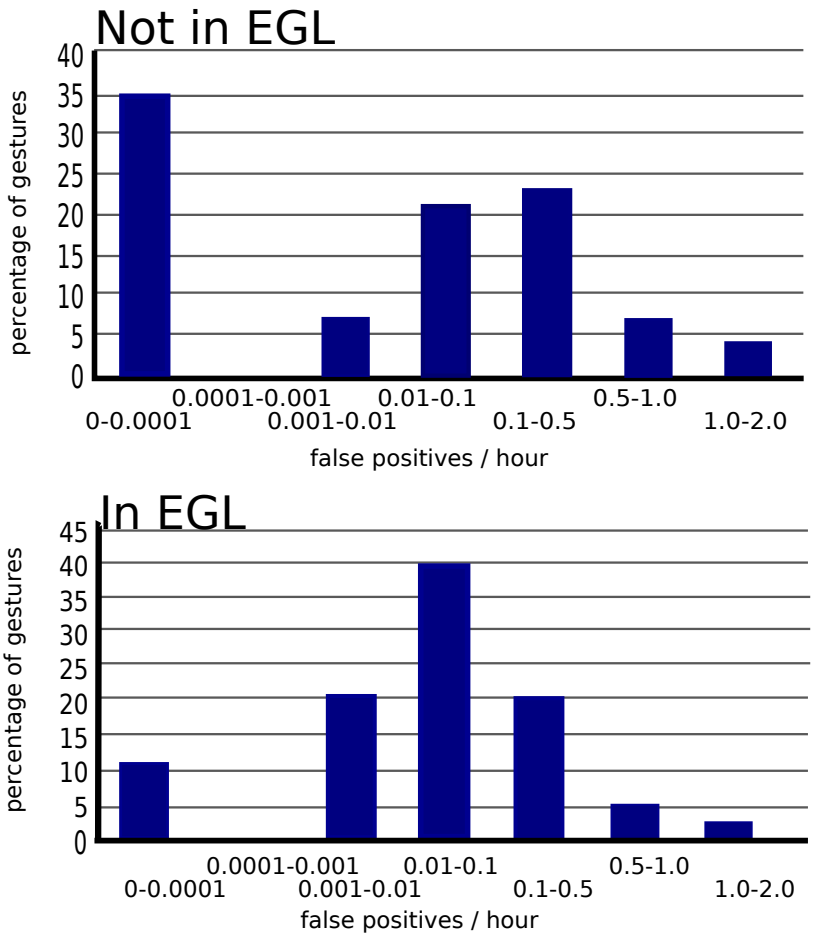


Figure 21: Histogram demonstrating the percentages of the number of false positives per hour for gestures with SAX representations not in the EGL (top) and all gestures with SAX representations in the EGL (bottom).

between sensor data and limb movement are not as apparent, such as accelerometer-based motion gestures, inverse kinematic models and 3D avatars seem appropriate both for prioritizing suggestions and for visualizing the gesture for the interaction designer. For situations with many degrees of freedom, such as whole body movement as tracked by the Microsoft Kinect©, the space of potential gestures may be extremely large. Physical and behavioral constraints might be applied to reduce the search space for the interaction designer. While MAGIC and MAGIC Summoning have been applied to multiple domains, we have only applied the gesture suggestion functions to trackpads. We are eager to investigate MAGIC Summoning's usefulness and usability in other domains.

9. Conclusion

We have described two pattern recognition tasks that can be used to help interaction designers create gesture interfaces: testing a user-defined gesture (and its classifier) against a previously captured database of typical usage sensor data to determine its tendency to trigger falsely and suggesting gestures automatically to the designer. We have shown that iSAX can be used to provide near immediate feedback to the user as to whether a gesture is inappropriate. While this method is approximate and recovers only a fraction of the total false positives in the EGL, MAGIC Summoning's results correlate strongly with those of HMMs, DTW, and the $1 Recognizer and can thus be used to provide guidance during training. We showed that MAGIC Summoning and the EGL could be used to create a null class of close false matches that increase the performance of the chosen classifier.

To suggest gestures to the interaction designer that may have low chance of triggering falsely, we exploited the SAX representation used to index the EGL. MAGIC Summoning generates all the strings not in the EGL, converts the SAX strings back into a gesture visualization, and suggests appropriate gestures to the designer. MAGIC Summoning also outputs classifiers for the gesture, trained on synthetic data generated from the SAX string. Using the task of finding command gestures for Mac trackpads, we showed that the gestures generated by MAGIC Summoning have generally low false positive rates when deployed and that the classifiers output by the system were adequate to the task of spotting the gesture.

Even if iSAX search of an EGL is not a perfect predictor for the false positives of a gesture in every day usage, we find that the approximations are sufficient to speed interface design significantly. MAGIC's methods are not intended to replace user testing with the final device. However, we believe that the tool will decrease the number of iterations needed to build a fast and stable gesture recognition interface.

Acknowledgments

This material is based upon work supported, in part, by the National Science Foundation under Grant No. 0812281. We would also like to thank Google for their support of the most recent advances in this project. Thanks also to David Quigley for sharing his Android EGL data set and Daniel Ashbrook for his original MAGICal efforts and collaborations.

References

Dan Ashbrook. *Enabling Mobile Microinteractions*. PhD thesis, Georgia Institute of Technology, Atlanta, Georgia, 2009.

Daniel Ashbrook and Thad Starner. MAGIC: a motion gesture design tool. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 2159–2168, New York, New York, 2010.

Mohammed Belatar and François Coldefy. Sketched menus and iconic gestures, techniques designed in the context of shareable interfaces. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 143–146, New York, New York, 2010.

Xiang Cao and Shumin Zhai. Modeling human performance of pen stroke gestures. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 1495–1504, New York, New York, 2007.

Chi Tai Dang and Elisabeth André. Surface-poker: multimodality in tabletop games. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 251–252, New York, New York, 2010.

Roger Dannenberg and Dale Amon. A gesture based user interface prototyping system. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 127–132, New York, New York, 1989.

Anind K. Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. a CAPpella: programming by demonstration of context-aware applications. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 33–40, New York, New York, 2004.

Jerry Fails and Dan Olsen. A design tool for camera-based interaction. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 449–456, New York, New York, 2003.

Ada Wai-Chee Fu, Eamonn Keogh, Leo Yung Lau, Chotirat Ann Ratanamahatana, and Raymond Chi-Wing Wong. Scaling and time warping in time series querying. *The International Journal on Very Large Data Bases*, 17(4):899–921, 2008.

François Guimbretière and Terry Winograd. Flowmenu: combining command, text, and data entry. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 213–216, 2000.

Richard Hamming. Error detecting and error correcting codes. *Bell Systems Technical Journal*, 29:147–160, 1950.

Yuichi Hattori, Sozo Inoue, and Go Hirakawa. A large scale gathering system for activity data with mobile sensors. In *Proceedings of the IEEE International Symposium on Wearable Computers*, pages 97–100, Washington, District of Columbia, 2011.

Edwin L. Hutchins, James D. Hollan, and Donald A. Norman. Direct manipulation interfaces. *Human-Computer Interaction*, 1(4):311–338, December 1985. ISSN 0737-0024.

Daniel Kohlsdorf, Thad Starner, and Daniel Ashbrook. MAGIC 2.0: A web tool for false positive prediction and prevention for gesture recognition systems. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pages 1–6, Washington, District of Columbia, 2011.

Danile Kohlsdorf. Motion gesture: False positive prediction and prevention. Master's thesis, University of Bremen, Bremen, Germany, 2011.

Yang Li. Gesture search: a tool for fast mobile data access. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 87–96, New York, New York, 2010.

Jessica Lin, Li Wei, and Eamonn Keogh. Experiencing sax: A novel symbolic representation of time series. *Journal of Data Mining and Knowledge Discovery*, 15(2): 107–144, 2007.

Chris Long. *Quill: A Gesture Design Tool for Pen-based User Interfaces*. PhD thesis, University of California, Berkeley, California, 2001.

Kent Lyons, Helene Brashear, Tracy Westeyn, Jung Soo Kim, and Thad Starner. GART: the gesture and activity recognition toolkit. In *Proceedings of the International Conference on Human-Computer Interaction: Intelligent Multimodal Interaction Environments*, pages 718–727, Berlin, Germany, 2007.

Dan Maynes-Aminzade, Terry Winograd, and Takeo Igarashi. Eyepatch: prototyping camera-based interaction through examples. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 33–42, New York, New York, 2007.

Tom M. Mitchell. *Machine Learning*. McGraw Hill, New York, New York, 1997.

Tom Ouyang and Yang Li. Bootstrapping personal gesture shortcuts with the wisdom of the crowd and handwriting recognition. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 2895–2904, 2012.

Antti Pirhonen, Stephen Brewster, and Christopher Holguin. Gestural and audio metaphors as a means of control for mobile devices. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 291–298, New York, New York, 2002.

Jin Shieh and Eamonn Keogh. iSAX: indexing and mining terabyte sized time series. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 623–631, New York, New York, 2008.

Jin-Wien Shieh. *Time Series Retrievel: Indexing and Mapping Large Datasets*. PhD thesis, University California, Riverside, California, 2010.

Thad Starner, Joshua Weaver, and Alex Pentland. Real-time American Sign Language recognition using desk and wearable computer-based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371 – 1375, December 1998.

Tracy Westeyn, Helene Brashear, Amin Atrash, and Thad Starner. Georgia Tech Gesture Toolkit: supporting experiments in gesture recognition. In *Proceedings of the International Conference on Multimodal Interfaces*, pages 85–92, New York, New York, 2003.

Hendrik Witt. *Human-Computer Interfaces for Wearable Computers*. PhD thesis, University Bremen, Bremen, Germany, 2007.

Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 159–168, New York, New York, 2007.

Hee-Deok Yang, Stan Sclaroff, and Seong-Whan Lee. Sign language spotting with a threshold model based on conditional random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(7):1264–1277, 2009.

JMLR: Workshop and Conference Proceedings 14:2189–2212, 2013 Published 7/13

Language-Motivated Approaches to Action Recognition

Manavender R. Malgireddy MRM42@BUFFALO.EDU
Ifeoma Nwogu INWOGU@BUFFALO.EDU
Venu Govindaraju GOVIND@BUFFALO.EDU
Department of Computer Science and Engineering
University at Buffalo, SUNY
Buffalo, NY 14260, USA

Editor: Isabelle Guyon and Vassilis Athitsos

Abstract

We present language-motivated approaches to detecting, localizing and classifying activities and gestures in videos. In order to obtain statistical insight into the underlying patterns of motions in activities, we develop a dynamic, hierarchical Bayesian model which connects low-level visual features in videos with poses, motion patterns and classes of activities. This process is somewhat analogous to the method of detecting topics or categories from documents based on the word content of the documents, except that our documents are dynamic. The proposed generative model harnesses both the temporal ordering power of dynamic Bayesian networks such as hidden Markov models (HMMs) and the automatic clustering power of hierarchical Bayesian models such as the latent Dirichlet allocation (LDA) model. We also introduce a probabilistic framework for detecting and localizing pre-specified activities (or gestures) in a video sequence, analogous to the use of filler models for keyword detection in speech processing. We demonstrate the robustness of our classification model and our spotting framework by recognizing activities in unconstrained real-life video sequences and by spotting gestures via a one-shot-learning approach.

Keywords: dynamic hierarchical Bayesian networks, topic models, activity recognition, gesture spotting, generative models

1. Introduction

Vision-based activity recognition is currently a very active area of computer vision research, where the goal is to automatically recognize different activities from a video. In a simple case where a video contains only one activity, the goal is to classify that activity, whereas, in a more general case, the objective is to detect the start and end locations of different specific activities occurring in a video. The former, simpler case is known as *activity classification* and latter as *activity spotting*. The ability to recognize activities in videos, can be helpful in several applications, such as monitoring elderly persons; surveillance systems in airports and other important public areas to detect abnormal and suspicious activities; and content based video retrieval, amongst other uses.

There are several challenges in recognizing human activities from videos and these include videos taken with moving background such as trees and other objects; different lighting conditions (day time, indoor, outdoor, night time); different view points; occlusions; variations within each activity (different persons will have their own style of performing an activity); large number of activities; and limited quantities of labeled data amongst others.

Recent advances in applied machine learning, especially in natural language and text processing, have led to a new modeling paradigm where high-level problems can be modeled using combinations of lower-level segmental units. Such units can be learned from large data sets and represent the universal set of alphabets to fully describe a vocabulary. For example, in a high-level problem such as speech recognition, a phoneme is defined as the smallest segmental unit employed to form an utterance (speech vector). Similarly, in language based documents processing, words in the document often represent the smallest segmental unit while in image-based object identification, the bag-of-words (or bag-of-features) technique learns the set of small units required to segment and label the object parts in the image. These features can then be input to generative models based on hierarchical clustering paradigms, such as topic modeling methods, to represent different levels of abstractions.

Motivated by the successes of this modeling technique in solving general high-level problems, we define an activity as a sequence of contiguous sub-actions, where the sub-action is a discrete unit that can be identified in a action stream. For example, in a natural setting, when a person waves goodbye, the sub-actions involved could be (i) raising a hand from rest position to a vertical upright position; (ii) moving the arm from right to left; and (iii) moving the arm from left to right. The entire activity or gesture[1] therefore consists of the first sub-action occurring once and the second and third sub-actions occurring multiple times. Extracting the complete vocabulary of sub-actions in activities is a challenging problem since the exhaustive list of sub-actions involved in a set of given activities is not necessarily known beforehand. We therefore propose machine learning models and algorithms to (i) compose a compact, near-complete vocabulary of sub-actions in a given set of activities; (ii) recognize the specific actions given a set of known activities; and (iii) efficiently learn a generative model to be used in recognizing or spotting a pre-specified action, given a set of activities.

We therefore hypothesize that the use of sub-actions in combination with the use of a generative model for representing activities will improve recognition accuracy and can also aid in activity spotting. We will perform experiments using various available publicly available benchmark data sets to evaluate our hypothesis.

2. Background and Related Work

Although extensive research has gone into the study of the classification of human activities in video, fewer attempts have been made to spot actions from an activity

1. When referring to activity spotting purposes, we use the term gestures instead of activities, only to be consistent with the terminology of the *ChaLearn Gesture Challenge*.

stream. A recent, more complete survey on activity recognition research is presented by Aggarwal and Ryoo (2011). We divide the related work in activity recognition into two main categories: activity classification and activity spotting.

2.1. Activity Classification

Approaches for activity classification can be grouped into three categories: (i) space-time approaches: a video is represented as a collection of space-time feature points and algorithms are designed to learn a model for each activity using these features; (ii) sequential approaches: features are extracted from video frames sequentially and a state-space model such as a hidden Markov model (HMM) is learned over the features; (iii) hierarchical approaches: an activity is modeled hierarchically, as combination of simpler low level activities. We will briefly describe each of these approaches along with the relevant literature, in sections below.

2.1.1. SPACE-TIME APPROACHES

Space-time approaches represent a video as a collection of feature points and use these points for classification. A typical space-time approach for activity recognition involves the detection of interest points and the computation of various descriptors for each interest point. The collection of these descriptors (bag-of-words) is therefore the representation of a video. The descriptors of labeled training data are presented to a classifier during training. Hence, when an unlabeled, unseen video is presented, similar descriptors are extracted as mentioned above and presented to a classifier for labeling. Commonly used classifiers in the space-time approach to activity classification include support vector machines (SVM), K-nearest neighbor (KNN), etc.

Spatio-temporal interest points were initially introduced by Laptev and Lindeberg (2003) and since then, other interest-point-based detectors such as those based on spatio-temporal Hessian matrix (Willems et al., 2008) and Gabor filters (Bregonzio et al., 2009; Dollár et al., 2005) have been proposed. Various other descriptors such as those based on histogram-of-gradients (HoG) (Dalal and Triggs, 2005) or histogram-of-flow (HoF) (Laptev et al., 2008), three-dimensional histogram-of-gradients (HoG3D) (Kläser et al., 2008), three-dimensional scale-invariant feature transform (3D-SIFT) (Scovanner et al., 2007) and local trinary patterns (Yeffet and Wolf, 2009), have also been proposed to describe interest points. More recently, descriptors based on tracking interest points have been explored (Messing et al., 2009; Matikainen et al., 2009). These use standard Kanade-Lucas-Tomasi (KLT) feature trackers to track interest points over time.

In a recent paper by Wang et al. (2009), the authors performed an evaluation of local spatio-temporal features for action recognition and showed that dense sampling of feature points significantly improved classification results when compared to sparse interest points. Similar results were also shown for image classification (Nowak et al., 2006).

2.1.2. SEQUENTIAL APPROACHES

Sequential approaches represent an activity as an ordered sequence of features, here the goal is to learn the order of specific activity using state-space models. HMMs and other dynamic Bayesian networks (DBNs) are popular state-space models used in activity recognition. If an activity is represented as a set of hidden states, each hidden state can produce a feature at each time frame, known as the observation. HMMs were first applied to activity recognition in 1992 by Yamato et al. (1992). They extracted features at each frame of a video by first binarizing the frame and dividing it into $(M \times N)$ meshes. The feature for each mesh was defined as the ratio of black pixels to the total number of pixels in the mesh and all the mesh features were concatenated to form a feature vector for the frame. An HMM was then learned for each activity using the standard Expectation-Maximization (EM) algorithm. The system was able to detect various tennis strokes such as forehand stroke, smash, and serve from one camera viewpoint. The major drawback of the conventional HMM was its inability to handle activities with multiple persons. A variant of HMM called coupled HMM (CHMM) was introduced by Oliver et al. (2000), which overcame this drawback by coupling HMMs, where each HMM in the CHMM modeled one person's activity. In their experiments they coupled two HMMs to model human-human interactions, but again this was somewhat limited in its applications. An approach to extend both HMM and CHMMs by explicitly modeling the duration of an activity using states was also proposed by Natarajan and Nevatia (2007). Each state in a coupled hidden semi-Markov model (CHSMMs) had its own duration and the sequence of these states defined the activity. Their experiments showed that CHSMM modeled an activity better than the CHMM.

2.1.3. HIERARCHICAL APPROACHES

The main idea of hierarchical approaches is to perform recognition of higher-level activities by modeling them as a combination of other simpler activities. The major advantage of these approaches over sequential approaches is their ability to recognize activities with complex structures. In hierarchical approaches, multiple layers of state-based models such as HMMs and other DBNs are used to recognize higher level activities. In most cases, there are usually two layers. The bottom layer takes features as inputs and learns atomic actions called *sub-actions*. The results from this layer are fed into the second layer and used for the actual activity recognition. A layered hidden Markov model (LHMM) (Oliver et al., 2002) was used in an application for office awareness. The lower layer HMMs classified the video and audio data with a time granularity of less than 1 second while the higher layer learned typical office activities such as phone conversation, face-to-face conversation, presentation, etc. Each layer of the HMM was designed and trained separately with fully labeled data. Hierarchical HMMs (Nguyen et al., 2005) were used to recognize human activities such as person having "short-meal", "snacks" and "normal meal". They also used a 2-layer architecture where lower layer HMM modeled simpler behaviors such as moving from one location in a room to another and the higher layer HMM used the information from layer one as its features. The higher layer was then used to recognize activities. A method based on modeling

temporal relationships among a set of different temporal events (Gong and Xiang, 2003) was developed and used for a scene-level interpretation to recognize cargo loading and unloading events.

The main difference between the above mentioned methods and our proposed method, is that these approaches assume that the higher-level activities and atomic activities (sub-actions) are known *a priori*, hence, the parameters of the model can be learned directly based on this notion. While this approach might be suitable for a small number of activities, it does not hold true for real-word scenarios where there is often a large number of sub-actions along with many activities (such as is found in the HMDB data set which is described in more detail in Section 6.2). *For activity classification, we propose to first compute sub-actions by clustering dynamic features obtained from videos, and then learn a hierarchical generative model over these features, thus probabilistically learning the relations between sub-actions, that are necessary to recognize different activities including those in real-world scenarios*.

2.2. Activity Spotting

Only a few methods have been proposed for activity spotting. Among them is the work of Yuan et al. (2009), which represented a video as a 3D volume and activities-of-interest as sub-volumes. The task of activity spotting was therefore reduced to one of performing an optimal search for activities in the video. Another work in spotting by Derpanis et al. (2010) introduced a local descriptor of video dynamics based on visual spacetime oriented energy measures. Similar to the previous work, their input was also a video which was searched for a specific action. The limitation of these techniques is their inability to adapt to changes in view points, scale, appearance etc. Rather than being defined on the motion patterns involved in an activity, these methods performed template matching type techniques, which do not readily generalize to new environments exhibiting a known activity. Both methods reported their results on the KTH and CMU data sets (described in more detail in Section 6), where the environment in which the activities were being performed did not readily change.

3. A Language-Motivated Hierarchical Model for Classification

Our proposed language-motivated hierarchical approach aims to perform recognition of higher-level activities by modeling them as a combination of other simpler activities. The major advantage of this approach over the typical sequential approaches and other hierarchical approaches is its ability to recognize activities with complex structures. By employing a hierarchical approach, multiple layers of state-based dynamic models can be used to recognize higher level activities. The bottom layers take observed features as inputs in order to recognize atomic actions (sub-actions). The results from these lower layers are then fed to the upper layers and used to recognize the modular activity.

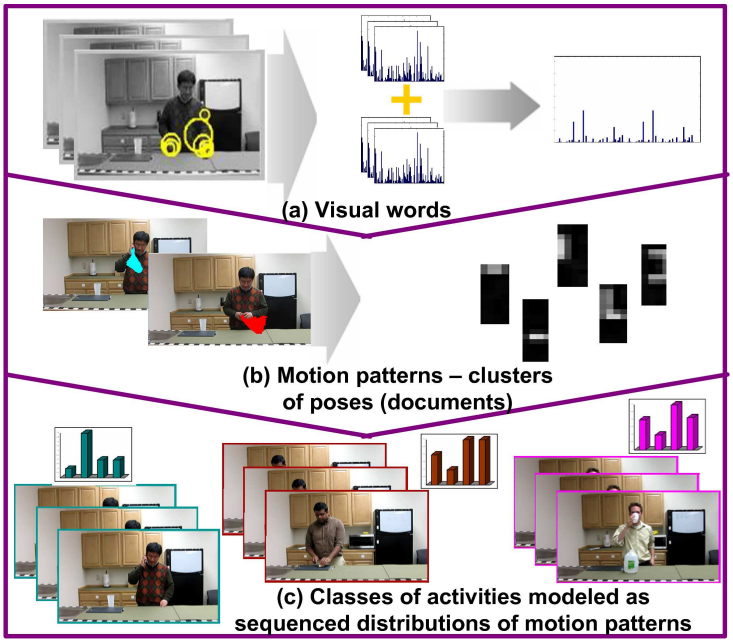


Figure 1: Our general framework abstracts low-level visual features from videos and connects them to poses, motion patterns and classes of activity. (a) A video sequence is divided into short segments with a few frames only. In each segment, the space time interest points are computed. At the interest points, HoG and HoF are computed, concatenated and quantized to represent our low-level *visual words*. We discover and model a distribution over visual words which we refer to as *poses* (not shown in image). (b) Atomic motions are discovered and modeled as distributions over poses. We refer to these atomic motions as *motion patterns* or *sub-actions*. (c) Each video segment is modeled as a distribution over motion patterns. The time component is incorporated by modeling the transitions between the video segments, so that a complete video is modeled as a dynamic network of motion patterns. The distributions and transitions of underlying motion patterns in a video determine the final activity label assigned to that video.

3.1. Hierarchical Activity Modeling using Multi-class Markov Chain Latent Dirichlet Allocation (MCMCLDA)

We propose a supervised dynamic, hierarchical Bayesian model, the multi-class Markov chain latent Dirichlet allocation (MCMCLDA), which captures the temporal information of an activity by modeling it as sequence of motion patterns, based on the Markov assumption. We develop this generative learning framework in order to obtain statistical insight into the underlying motions patterns (sub-actions) involved in an activity. An important aspect of this model is that motion patterns are shared across activities. So although the model is generative in structure, it can act discriminatively as it specif-

ically learns which motion patterns are present in each activity. The fact that motion patterns are shared across activities was validated empirically (Messing et al., 2009) on the University of Rochester activities data set. Our proposed generative model harnesses both the temporal ordering power of DBNs and the automatic clustering power of hierarchical Bayesian models. The model correlates these motion patterns over time in order to define the signatures for classes of activities. Figure 1 shows an overview of the implementation network although we do not display *poses*, since they have no direct meaningful physical manifestations.

A given video is broken into motion segments comprising of either a combination of a fixed number of frames, or at the finest level, a single frame. Each motion segment can be represented as bag of vectorized descriptors (visual words) so that the input to the model (at time t) is the bag of visual words for motion segment t. Our model is similar in sprit to Hospedales et al. (2009), where the authors mine behaviors in video data from public scenes using an unsupervised framework. A major difference is that our MCMCLDA is a supervised version of their model in which motion-patterns/behaviors are shared across different classes, which makes it possible to handle a large number of different classes. If we assume that there exists only one class, then the motion-patterns are no longer shared, our model also becomes unsupervised and will thus be reduced to that of Hospedales et al. (2009).

We view MCMCLDA as a generative process and include a notation section before delving into the details of the LDA-type model:

m = any single video in the corpus,
z_t = motion pattern at time t (a video is assumed to be made up of motion patterns),
$y_{t,i}$ = the hidden variable representing a pose at motion pattern i, in time t (motion patterns are assumed to be made up of poses),
$x_{t,i}$ = the slices of the input video which we refer to as visual words and are the only observable variables,
ϕ_y = the visual word distribution for pose y,
θ_z = motion pattern specific pose distribution,
c_m is the class label for the video m; (for one-shot learning, one activity is represented by one video $(N_m = 1)$),
$\psi_j = j^{th}$ class-specific transition matrix for the transition from one motion pattern to the next,
γ_c = the transition matrix distribution for a video,
α,β = the hyperparameters of the priors.

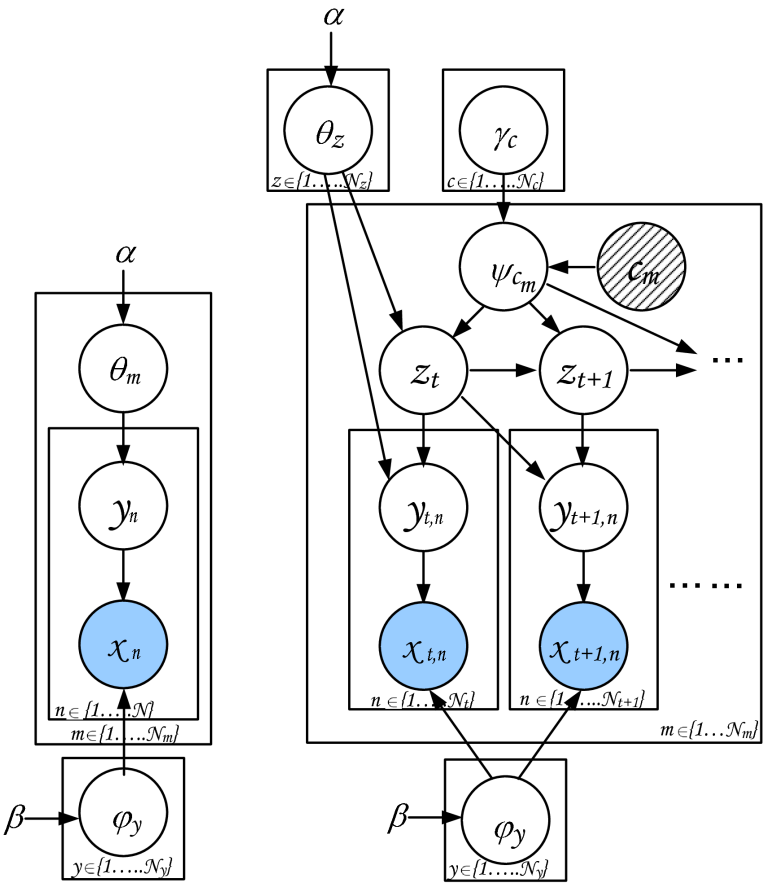


Figure 2: Left: plates diagram for standard LDA; right: plates diagram for a dynamic model which extends LDA to learning the states from sequence data.

The complete generative model is given by:

$$\begin{aligned}
\psi_j^z &\sim Dir(\psi_j^z|\gamma_j),\\
\theta_z &\sim Dir(\theta_z|\alpha),\\
\phi_y &\sim Dir(\phi_y|\beta),\\
z_t &\sim Mult(z_t|\psi_j^{z_{t-1}}),\\
y_{t,i} &\sim Mult(y_{t,i}|\theta_{z_t}),\\
x_{t,i} &\sim Mult(x_{t,i}|\phi_{y_{t,i}}),
\end{aligned}$$

where $Mult(\cdot)$ refers to a multinomial distribution.

Now, consider the Bayesian network of MCMCLDA shown in Figure 2. This can be interpreted as follows: For each video m in the corpus, a motion pattern indicator

z_t is drawn from $p(z_t|z_{t-1},\psi_{c_m})$, denoted by $\text{Mult}(\psi_{c_m}^{z_{t-1}})$, where c_m is the class label for the video m. Then the corresponding pattern specific pose distribution θ_{z_t} is used to draw visual words for that segment. That is, for each visual word, a pose indicator $y_{t,i}$ is sampled according to pattern specific pose distribution θ_{z_t}, and then the corresponding pose-specific word distribution $\phi_{y_{t,i}}$ is used to draw a visual word. The poses ϕ_y, motion patterns θ_z and transition matrices ψ_j are sampled once for the entire corpus.

The joint distribution of all known and hidden variables given the hyperparameters for a video is:

$$p(\{x_t,y_t,z_t\}_1^T,\phi,\psi_j,\theta|\alpha,\beta,\gamma_j)=p(\phi|\beta)p(\theta|\alpha)p(\psi|\gamma_j)\prod_t\prod_i p(x_{t,i}|y_{t,i})p(y_{t,i}|z_t)p(z_t|z_{t-1}).$$

3.2. Parameter Estimation and Inference of the MCMCLDA Model

As in the case with LDA, exact inference is intractable. We therefore use collapsed Gibbs sampler for approximate inference and learning. The update equation for pose from which the Gibbs sampler draws the hidden pose $y_{t,i}$ is obtained by integrating out the parameters θ,ϕ and noting that $x_{t,i}=x$ and $z_t=z$:

$$p(y_{t,i}=y|\mathbf{y}_{\neg(\mathbf{t},\mathbf{i})},\mathbf{z},\mathbf{x})\propto\frac{n_{x,y}^{\neg(t,i)}+\beta}{\sum_{x=1}^{N_x}n_{x,y}^{\neg(t,i)}+N_x\beta}(n_{y,z}^{\neg(t,i)}+\alpha), \tag{1}$$

where $n_{x,y}^{\neg(t,i)}$ denote the number of times that visual word x is observed with pose y excluding the token at (t,i) and $n_{y,z}^{\neg(t,i)}$ refers to the number of times that pose y is associated with motion pattern z excluding the token at (t,i). N_x is size of codebook and N_y is the number of poses.

The Gibbs sampler update for motion-pattern at time t is derived by taking into account that at time t, there can be many different poses associated to a single motion-pattern z_t and also the possible transition from z_{t-1} to z_{t+1}. The update equation for z_t can be expressed as :

$$p(z_t=z|\mathbf{y},\mathbf{z}_{\neg\mathbf{t}})\propto p(y_t|z_t=z,z_{\neg t},y_{\neg t})p(z_t=z|z_{\neg t}^{c_m},c_m). \tag{2}$$

The likelihood term $p(y_t|z_t=z,z_{\neg t},y_{\neg t})$ cannot be reduced to the simplified form as in LDA as the difference between $n_{y,z}^{\neg t}$ and $n_{y,z}$ is not one, since there will be multiple poses associated to the motion-pattern z_t. $n_{y,z}$ denotes the number of times pose y is associated with motion-pattern z and $n_{y,z}^{\neg t}$ refers to the number of times pose y is observed with motion-pattern z excluding the poses (multiple) at time t. Taking the above condition into account, the likelihood term can be obtained as below:

$$p(y_t|z_t=z,z_{\neg t},y_{\neg t})=\frac{\prod_y\Gamma(n_{y,z}+\alpha)}{\prod_y\Gamma(n_{y,z}^{\neg t}+\alpha)}\frac{\Gamma(\sum_y n_{y,z}^{\neg t}+N_y\alpha)}{\Gamma(\sum_y n_{y,z}+N_y\alpha)}.$$

Prior term $p(z_t = z|z_{\neg t}^{c_m}, c_m)$ is calculated as below depending on the values of z_{t-1}, z_t and z_{t+1}.

$$
\begin{aligned}
&if \quad z_{t-1} \neq z: \\
&= \frac{n_{z_{t-1},z,\neg t}^{(c_m)} + \gamma_{c_m}}{\sum_z n_{z_{t-1},z,\neg t}^{(c_m)} + N_z\gamma_{c_m}} \frac{n_{z,z_{t+1},\neg t}^{(c_m)} + \gamma_{c_m}}{\sum_{z_{t+1}} n_{z,z_{t+1},\neg t}^{(c_m)} + N_z\gamma_{c_m}}, \\
&if \quad z_{t-1} = z = z_{t+1}: \\
&= \frac{n_{z_{t-1},z,\neg t}^{(c_m)} + 1 + \gamma_{c_m}}{\sum_z n_{z_{t-1},z,\neg t}^{(c_m)} + 1 + N_z\gamma_{c_m}} \frac{n_{z,z_{t+1},\neg t}^{(c_m)} + \gamma_{c_m}}{\sum_{z_{t+1}} n_{z,z_{t+1},\neg t}^{(c_m)} + N_z\gamma_{c_m}}, \\
&if \quad z_{t-1} = z \neq z_{t+1}: \\
&= \frac{n_{z_{t-1},z,\neg t}^{(c_m)} + \gamma_{c_m}}{\sum_z n_{z_{t-1},z,\neg t}^{(c_m)} + N_z\gamma_{c_m}} \frac{n_{z,z_{t+1},\neg t}^{(c_m)} + \gamma_{c_m}}{\sum_{z_{t+1}} n_{z,z_{t+1},\neg t}^{(c_m)} + 1 + N_z\gamma_{c_m}}.
\end{aligned}
$$

Here $n_{z_{t-1},z,\neg t}^{(c_m)}$ denotes the count from all the videos with the label c_m where motion-pattern z is followed by motion-pattern z_{t-1} excluding the token at t. $n_{z,z_{t+1},\neg t}^{(c_m)}$ denotes the count from all the videos with label c_m where motion-pattern z_{t+1} is followed by motion-pattern z_t excluding the token at t. N_z is the number of motion-patterns. The Gibbs sampling algorithm iterates between Equations 1 and 2 and finds the approximate posterior distribution. To obtain the resulting model parameters $\{\phi, \theta, \phi\}$ from the Gibbs sampler, we use the expectation of their distribution (Heinrich, 2008), and collect N_s such samples of the model parameters.

For inference, we need to find the best motion-pattern sequence for a new video. The Gibbs sampler draws N_s samples of parameters during the learning phase. We assume that these are sufficient statistics for the model and that no further adaptation of parameters is necessary. We then adopt the Viterbi decoding algorithm to find the best motion-pattern sequence. We approximate the integral over ϕ, θ, ψ using the point estimates obtained during learning. To formulate the recursive equation for the Viterbi algorithm, we can define the quantity

$$
\begin{aligned}
\delta_t(i) &= \max_{z_1,\ldots,z_{t-1}} \int_{\phi,\theta,\psi_{c_m}} p(z_{1:(t-1)}, z_t = i, x_{1:t}|\phi, \theta, \psi_{c_m}), \\
&\approx \max_{z_1,\ldots,z_{t-1}} \left(\frac{1}{N_s} \sum_s p(z_{1:(t-1)}, z_t = i, x_{1:t}|\phi^s, \theta^s, \psi_{c_m}^s) \right),
\end{aligned}
$$

that is $\delta_t(i)$ is the best score at time t, which accounts for first t motion-segments and ends in motion-pattern i. By induction we have

$$
\delta_{t+1}(j) \approx \max_i \delta_t(i) \frac{1}{N_s} \sum_s p(z_{t+1} = j|z_t = i, \psi_{c_m}^s) p(x_{t+1}|z_{t+1} = j, \theta^s, \phi^s). \tag{3}
$$

To find the best motion-pattern, we need to keep track of the arguments that maximized Equation 3. For the classification task we calculate the likelihood $p^\star$ for each class and

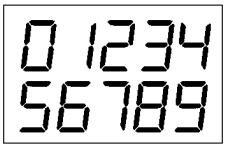


Figure 3: Digital digits for simulations.

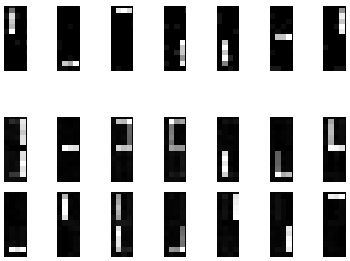


Figure 4: The top row shows seven poses discovered from clustering words. The middle and bottom rows show the fourteen motion patterns discovered and modeled from the poses. A motion pattern captures one or more strokes in the order they are written.

assign the label which has maximum value in:

$$p^{\star} = \max_{1 \leq j \leq N_z} \delta_T(j).$$

4. Experiments and Results using MCMCLDA

In this section, we present our observations as well as the results of applying our proposed language-motivated hierarchical model to sub-action analysis as well as to activity classification, using both simulated data as well as a publicly available benchmark data set.

4.1. Study Performed on Simulated Digit Data

To flesh out the details of our proposed hierarchical classification model, we present a study performed on simulated data. The ten simulated dynamic activity classes were the writing of the ten digital digits, 0-9 as shown in Figure 3. The word vocabulary was made up of all the pixels in a 13×5 grid and the topics or poses represented the distribution over the words. An activity class therefore consisted of the steps needed to simulate the writing of each digit and the purpose of the simulation was to visually observe the clusters of motion patterns involved in the activities.

4.1.1. ANALYSIS OF RESULTS

A total of seven clusters were discovered and modeled, as shown in Figure 4. These represent the simulated strokes (or topics) involved in writing each digit. There were fourteen motion patterns discovered, as shown in the two bottom rows of Figure 4. These are the probabilistic clusters of the stroke motions. An activity or digit written was therefore classified based on the sequences of distributions of these motion patterns over time.

4.2. Study Performed on the Daily Activities Data Set

The *Daily Activities data set* contains high resolution (1280×760 at 30 fps) videos, with 10 different complex daily life activities such as *eating banana, answering phone, drinking water, etc.*. Each activity was performed by five subjects three times, yielding a total of 150 videos. The duration of each video varied between 10 and 60 seconds.

We generated visual words for the MCMCLDA model in a manner similar to Laptev (2005), where the Harris3D detector (Laptev and Lindeberg, 2003) was used to extract space-time interest points at multiple scales. Each interest point was described by the concatenation of HoF and HoG (Laptev, 2005) descriptors. After the extraction of these descriptors for all the training videos, we used the k-means clustering algorithm to form a codebook of descriptors (or visual words (VW)). Furthermore, we vector-quantized each descriptor by calculating its membership with respect to the codebook. We used the original implementation available online[2] with the standard parameter settings to extract interest points and descriptors.

Due to the limitations of the distributed implementation of space-time interest points (Laptev et al., 2008), we reduced the video resolution to 320 x 180. In our experimental setup, we used 100 videos for training and 50 videos for testing exactly as pre-specified by the original publishers of this data set (Messing et al., 2009). Both the training and testing sets had a uniform distribution of samples for each activity. We learned our MCMCLDA model on the training videos, with a motion segment size of 15 frames. We ran a Gibbs sampler for a total of 6000 iterations, ignoring the first 5000 sweeps as burn-in, then took 10 samples at a lag of 100 sweeps. The hyperparameters were fixed initially with values $(\alpha = 5, \beta = 0.01, \gamma = 1)$ and after burn-in, these values were empirically estimated using maximum-likelihood estimation (Heinrich, 2008) as $(\alpha = 0.34, \beta = 0.001$ and $\gamma = \{0.04, 0.05, 0.16, 0.22, 0.006, 0.04, 0.13, 0.05, 0.14, 0.45\})$. We set the number of motion-patterns, poses and codebook size experimentally as $N_z = 100, N_y = 100$ and $N_x = 1000$.

The confusion matrix computed from this experiment is given in Figure 5 and a comparison with other activity recognition methods on the Daily Activities data set is given in Table 1. Because the data set was already pre-divided, the other recognition methods reported in Table 1 were trained and tested on the same sets of training and testing videos.

2. Implementation can be found at http://www.irisa.fr/vista/Equipe/People/Laptev/download.html#stip.

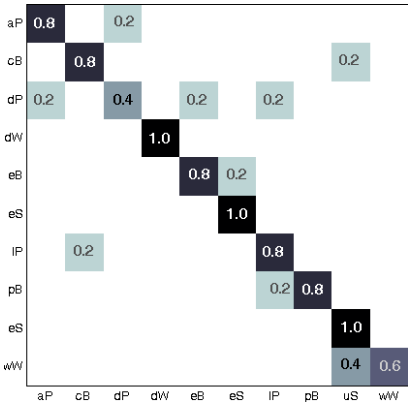


Figure 5: Confusion matrix for analyzing the University of Rochester daily activities data set. Overall accuracy is 80.0%. Zeros are omitted for clarity. The labels and their corresponding meaning are: ap-answer phone; cB-chop banana; dP-dial phone; dW-drink water; eB-eat banana; eS-eat snack; lP-lookup in phonebook; pB-peel banana; uS-use silverware; wW-write on whiteboard.

Technique	**Focus**	**Accuracy**
Latent velocity trajectory features (Messing et al., 2009)[3]	motion feature enhancement	67%
Naive-Bayes pairwise trajectory features (Matikainen et al., 2010)	motion feature enhancement	70%
Salient region tracking features (Bilen et al., 2011)	motion feature enhancement	74%
Video temporal cropping technique	motion feature enhancement	80%
Our supervised dynamic hierarchical model	dynamic hierarchical modeling	80%
Direction of motion features (Benabbas et al., 2010)	motion feature enhancement	81%

Table 1: The accuracy numbers reported in literature from applying different activity recognition techniques on the daily activities data set.

Qualitatively, Figure 7 pictorially illustrates some examples of different activities having the same underlying shared motion patterns.

3. The authors also reported velocity trajectory feature augmented with prior spatial layout information, resulting in an accuracy of 89%.

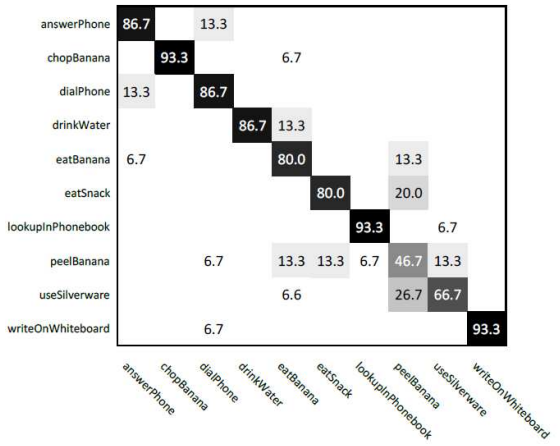


Figure 6: Confusion matrix results from Benabbas et al. (2010) on the University of Rochester daily activities data set.

4.2.1. ANALYSIS OF RESULTS

We present comparative results with other systems in Table 1. The results show that the approach based on computing a distribution mixture over motion orientations at each spatial location of the video sequence (Benabbas et al., 2010), slightly outperformed our hierarchical model. Interestingly, in our test, one activity, the *write on whiteboard (wW)* activity is quite confused with *use silverware (uS)* activity, significantly bringing down the overall accuracy. The confusion matrix for Benabbas et al. (2010) is presented in Figure 6 and it shows several of the classes being confused, no perfect recognition scores and also one of the class recognition rates being below 50%. Being a generative model, the MCMCLDA model performs comparably to other discriminative models in a class labeling task.

Figure 7 pictorially illustrates some examples of different activities having the same underlying shared motion patterns. For example, the activity of answering the phone shares a common motion pattern (#85) with the activities of dialing the phone and drinking water. Semantically, we observe that this shared motion is related to the *lifting* sub-action.

5. A Language-Motivated Model for Gesture Recognition and Spotting

Few methods have been proposed for gesture spotting and among them include the work of Yuan et al. (2009), who represented a video as a 3D volume and activities-of-interest as sub-volumes. The task of gesture spotting was therefore reduced to performing an optimal search for gestures in the video. Another work in spotting was presented by Derpanis et al. (2010) who introduced a local descriptor of video dynamics based on

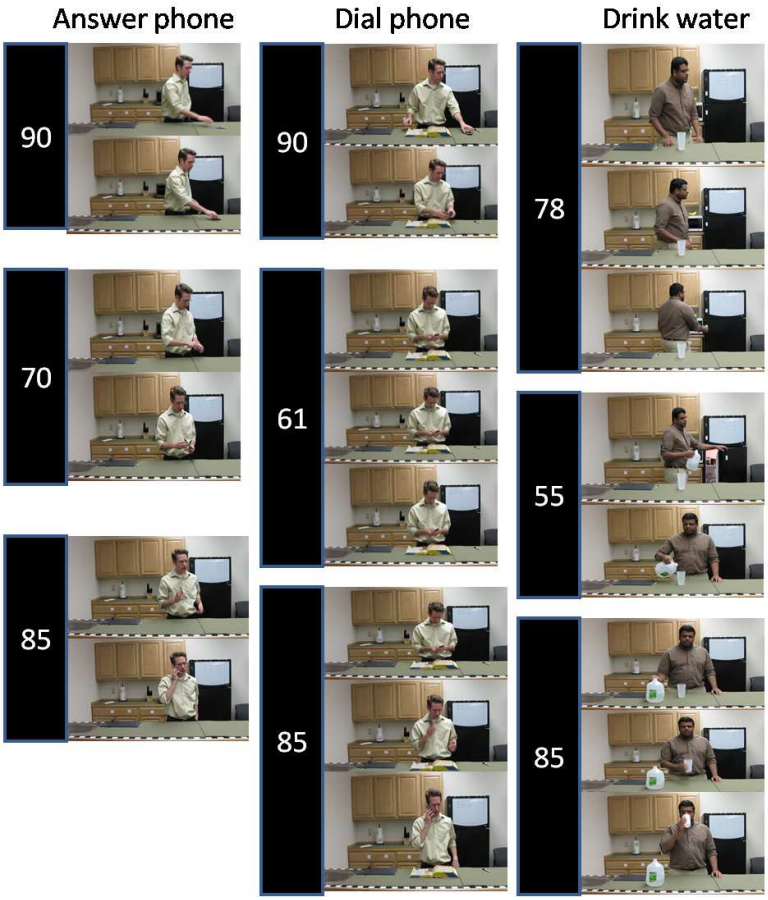


Figure 7: Different activities showing shared underlying motion patterns. The shared motion patterns are 85 and 90, amidst other underlying motion patterns shown.

visual space-time oriented energy measures. Similar to the previous work, their input was also a video in which a specific action was searched for. The limitation in these techniques is their inability to adapt to changes in view points, scale, appearance, etc. Rather than being defined on the motion patterns involved in an activity, these methods performed a type of 3D template matching on sequential data; such methods do not readily generalize to new environments exhibiting the known activity. *We therefore propose to develop a probabilistic framework for gesture spotting that can be learned with very little training data and can readily generalize to different environments*.

Justification: Although the proposed framework is a generative probabilistic model, it performs comparably to the state-of-the-art activity techniques which are typically discriminative in nature, as demonstrated in Tables 2 and 3. An additional benefit of

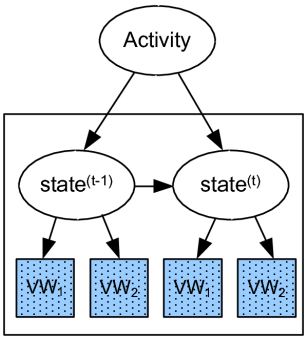


Figure 8: Plates model for mcHMM showing the relationship between activities or gestures, states and the two channels of observed visual words (VW).

the framework is its usefulness for gesture spotting based on learning from only one, or few training examples.

Background: In speech recognition, unconstrained keyword spotting refers to the identification of specific words uttered, when those words are not clearly separated from other words, and no grammar is enforced on the sentence containing them. Our proposed spotting framework uses the Viterbi decoding algorithm and is motivated by the *keyword-filler HMM for spotting keywords in continuous speech*. The current state of the art keyword filler HMM dates back to the seminal papers of Rohlicek et al. (1989) as well as Rose and Paul (1990), where the basic idea is to create one HMM of the keyword and a separate HMM of the filler or non keyword regions. These two models are then combined to form a composite filler HMM that is used to annotate speech parts using the Viterbi decoding scheme. Putative decisions arise when the Viterbi path crosses the keyword portion of the model. The ratio between the likelihood of the Viterbi path that passes through the keyword model and the likelihood of an alternate path that passes solely through the filler portion can be used to score the occurrence of keywords. In a similar manner, we compute the probabilistic signature for a gesture class, and using the filler model structure, we test for the presence of that gesture within a given video. For one-shot learning, the parameters of the single training video are considered to be sufficiently representative of the class.

5.1. Gesture Recognition using a Multichannel Dynamic Bayesian Network

In a general sense, the spotting model can be interpreted as an HMM (whose random variables involve hidden states and observed input nodes) but unlike the classic HMM, this model has multiple input channels, where each channel is represented as a distribution over the visual words corresponding to that channel. In contrast to the classic HMM, our model can have multiple observations per state and channel, and we refer to this as the multiple channel HMM (mcHMM). Figure 8 shows a graphical representation of the mcHMM.

5.2. Parameter Estimation for the Gesture Recognition Model

To determine the probabilistic signature of an activity class, one mcHMM is trained for each activity. The generative process for mcHMM involves first sampling a state from an activity, based on the transition matrix for that activity; then a frame-feature comprising of the distribution of visual words is sampled according to a multinomial distribution for that state[4] and this is repeated for each frame. Similar to a classic HMM, the parameters for the mcHMM are therefore:

1. Initial state distribution $\pi = \{\pi_i\}$,
2. State transition probability distribution $A = \{a_{ij}\}$,
3. Observation densities for each state and descriptor $B = \{b_i^d\}$.

The joint probability distribution of observations (O) and hidden state sequence (Q) given the parameters of the multinomial representing a hidden state (λ) can be expressed as:

$$P(O,Q|\lambda) = \pi_{q_1} b_{q_1}(O_1) \prod_{t=2}^{T} a_{q_{t-1}q_t} \cdot b_{q_t}(O_t),$$

where $b_{q_t}(O_t)$ is modeled as follows:

$$\begin{aligned} b_{q_t}(O_t) = & \quad \prod_{d=1}^{D} b_q^d(O_t^d), \\ = & \quad \prod_{d=1}^{D} Mult(O_t^d|b_q^d), \end{aligned}$$

and D is the number of descriptors.

EM is implemented to find the maximum likelihood estimates. The update equations for the model parameters are:

$$\hat{\pi} = \sum_{r=1}^{R} \gamma_1^r(i),$$

$$\hat{a}_{ij} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T} \eta_t^r(i,j)}{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_t^r(i)},$$

$$\hat{b}_j^d(k) = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_t^r(j) \cdot \frac{n_t^{d,k}}{n_t^{d,\cdot}}}{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_t^r(j)},$$

where R is number of videos and $\gamma_1(i)$ is the expected number of times the activity being modeled started with state i;

$\eta_t^r(i,j)$ is the expected number of transitions from state i to state j and $\gamma_t^r(i)$ is the expected number of transitions from state i;

$n_t^{d,k}$ is the number of times that visual word k occurred in descriptor d at time t and $n_t^{d,\cdot}$ is the total number of visual words that occurred in descriptor d at time t.

4. States are modeled as multinomials since our input observables are discrete values.

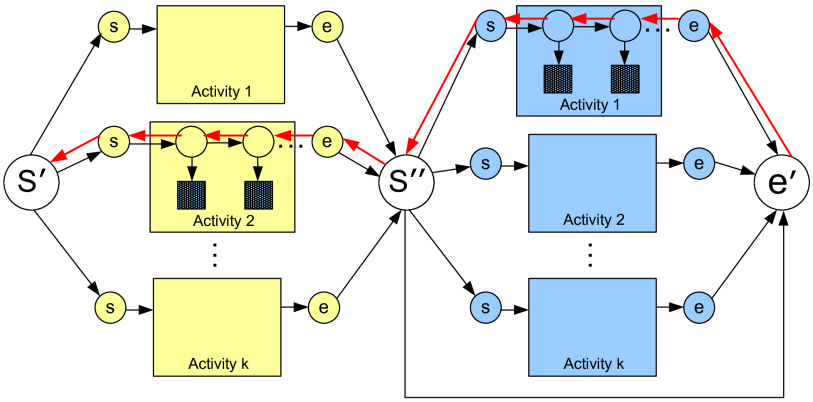


Figure 9: Activity spotting by computing likelihoods via Viterbi decoding. The toy example shown assumes there are at most two activities in any test video, where the first activity is from the set of activities that start from s' and end at s'', followed by one from the set that start from s'' and end at e'. The image also shows an example of a putative decision path from e' to s', after the decoding is completed.

5.3. Gesture Spotting via Inference on the Model

The gesture spotting problem is thus reduced to an inference problem where, given a new not-previously-seen test video, and the model parameters or probabilistic signatures of known activity classes, the goal is to establish which activity class distributions most likely generated the test video. This type of inference can be achieved using the Viterbi algorithm.

We constructed our spotting network such that there could be a maximum of five gestures in a video. This design choice was driven by our participation in the Chalearn competition where there was a maximum of five gestures in every test video. Each of these gesture classes was seen during training, hence, there were no random gestures inserted into the test video. This relaxed our network, compared to the original filler model in speech analysis, where there can exist classes that have not been previously seen. Figure 9 shows an example of the stacked mcHMMs involved the gesture spotting task. This toy example shown in the figure can spot gestures in a test video comprised of at most two gestures. This network has a non-emitting start state S'. This state does not have any observation density associated with it. From this state, we can enter any of K gestures, which is shown by having edges from S' to K mcHMMs. All the gestures are then connected to non-emitting state S'' which represents the end of first gesture. Similarly we can enter the second gesture from S'' and end at e' or directly go from S'' to e' which handles the case for a video having only one gesture. This can be easily extended to the case where there are at most five gestures.

The Viterbi decoding algorithm was implemented to traverse the stacked network and putative decisions arose when the Viterbi path crosses the keyword portion of the model. The ratio between the likelihood of the Viterbi path that passed through the keyword model and the likelihood of an alternate path that passes through the non-keyword portion was then used to score the occurrence of a keyword, where a keyword here referred to a gesture class. An empirically chosen threshold value was thus used to select the occurrence of a keyword in the video being decoded.

6. Experiments and Results using mcHMM

In this section, we present our approach on generating visual words and our observations as well as the results of applying proposed mcHMM model to activity classification and gesture spotting, using publicly available benchmark data sets.

6.1. Generating Visual Words

An important step in generating visual words is the the need to extract interest points from frames sampled from the videos at 30 fps. Interest points were obtained from the KTH and HMDB data set by sampling dense points in every frame in the video and then tracking these points for the next L frames. These are known as dense trajectories. For each of these tracks, motion boundary histogram descriptors based on HoG and HoF descriptors were extracted. These features are similar to the ones used in dense trajectories (Wang et al., 2011), although rather than sampling interest points at every L frames or when the current point is lost before being tracked for L frames, we sampled at every frame. By so doing, we obtained a better representation for each frame, whereas the original work used the features to represent the whole video and was not frame-dependent.

Because the HMDB data set is comprised of real-life scenes which contain people and activities occurring at multiple scales, the frame-size in the video was reduced by a factor of two repeatedly, and motion boundary descriptors were extracted at multiple scales. In the Chalearn data set, since the videos were comprised of RGB-depth frames, we extracted interest points by (i) taking the difference between two consecutive depth frames and/or (ii) calculating the centroid of the depth foreground in every frame and computing the extrema points (from that centroid) in the depth foreground. The second process ensured that extrema points such as the hands, elbows, top-of-the-head, etc., were always included in the superset of interest points. The top and bottom image pairs in Figure 10 show examples of consecutive depth frames from the Chalearn data set, with the interest points obtained via the two different methods, superimposed. Again, HoG and HoF descriptors were extracted at each interest point so that similar descriptors could be obtained in all the cases. We used a patch size of 32×32 and a bin size of 8 for HoG and 9 for HoF implementation.

The feature descriptors were then clustered to obtain visual words. In general, from the literature (Wang et al., 2011; Laptev et al., 2008), in order to limit complexity, researchers randomly select a finite number of samples (roughly in the order of 100,000)

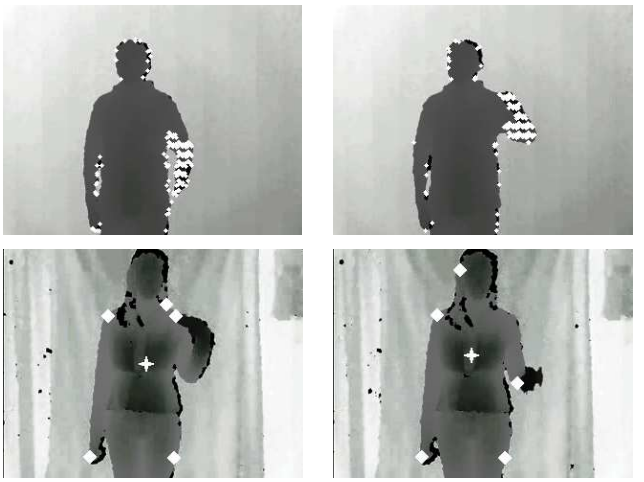


Figure 10: Interest points for 2 consecutive video frames. Top: Depth-subtraction interest points; bottom: extrema interest points (with centroid).

and cluster these to form visual words. This could prove reasonable when the number of samples is a few orders of magnitude greater than 100,000. But in dealing with densely sampled interest points at every frame, the amount of descriptors generated especially at multiple scales become significantly large. We therefore divided the construction of visual words for HMDB data set into a two step process where visual words were first constructed for each activity class separately, and then the visual words obtained for each class were used as the input samples to cluster the final visual words. For the smaller data sets such as KTH and Chalearn Gesture Data Set, we randomly sampled 100,000 points and clustered them to form the visual words.

6.2. Study Performed on the HMDB and KTH Data Sets

In order to compare our framework to the other current state-of-the-art methods, we performed activity classification on video sequences created from the KTH database (Schüldt et al., 2004); KTH is a relatively simplistic data set comprised of 2391 video clips used to train/test six human actions. Each action is performed several times by 25 subjects in various outdoor and indoor scenarios. We split the data into training set of 16 subjects and test set of 9 subjects, which is exactly the same setup used by the authors of the initial paper (Schüldt et al., 2004). Table 2 shows the comparison of accuracies obtained.

Similarly, we performed activity classification tests on Human Motion Database (HMDB) (Kuehne et al., 2011). HMDB is currently the most realistic database for human activity recognition comprising of 6766 video clips and 51 activities extracted from a wide range of sources like YouTube, Google videos, digitized movies and other videos available on the Internet. We follow the original experimental setup using three

Method	Accuracy
(Laptev et al., 2008)	91.8%
(Yuan et al., 2009)	93.3%
(Wang et al., 2011)	94.2%
(Gilbert et al., 2011)	94.5%
(Kovashka and Grauman, 2010)	94.53%
proposed mcHMM	**94.67**

Table 2: Comparison of our proposed model and features for KTH data set.

Method	Accuracy
Best results on 51 activities (original) (Kuehne et al., 2011)	23.18%
Proposed mcHMM on 51 activities (original)	**25.64%**
Best results on 10 activities (original) (Kuehne et al., 2011)	54.3%
Proposed mcHMM on 10 activities (original)	**57.67%**
Proposed mcHMM on 10 activities (stabilized)	**66.67%**

Table 3: Comparison of our proposed model and features for the HMDB data set.

train-test splits (Kuehne et al., 2011). Each split has 70 video for training and 30 videos for testing for each class. All the videos in the data set are stabilized to remove the camera motion and the authors of the initial paper (Kuehne et al., 2011) report results on both original and stabilized videos for 51 activities. The authors also selected 10 common actions from HMDB data set that were similar to action categories in the UCF50 data set (University of Central Florida) and compared the recognition performance. Table 3 summarizes the performance of proposed mcHMM method on 51 activities as well as 10 activities for both original and stabilized videos.

6.2.1. ANALYSIS OF RESULTS

For both the case of simple actions as found in the KTH data set and the case of significantly more complex actions as found in the HMDB data set, the mcHMM model performs comparably with other methods, outperforming them in the activity recognition task. Our evaluation against state-of-the-art data sets suggest that performance is not significantly affected over a range of factors such as camera position and motion as well as occlusions. This suggests that the overall framework (combination of dense descriptors and a state-based probabilistic model) is fairly robust with respect to these low-level video degradations. At the time of this submission, although we outperformed the only currently reported accuracy results on the HMDB data set, as shown by the ccuracy scores reported, the framework is still limited in its representative power to capture the complexity of human actions.

6.3. Study Performed on the ChaLearn Gesture Data Set

Lastly, we present our results of gesture spotting from the ChaLearn gesture data set (ChaLearn). The ChaLearn data set consisted of video frames with RGB-Depth information. Since the task-at-hand was gesture spotting via one-shot learning, only one video per class was provided to train an activity (or gesture). The data set was divided into three parts: development, validation and final. In the first phase of the competition, participants initially developed their techniques against the development data set. Ground truth was not provided during the development phase. Once the participants had a working model, they then ran their techniques against the validation data set and uploaded their predicted results to the competition website, where they could receive feedback (scores based on edit distances) on the correctness of the technique. In the last phase of the competition, the final data set was released so that participants could test against it and upload their predicted results. Similarly, edit scores were used to measure the correctness of the results and the final rankings were published on the competition website.

We reported results using two methods i) mcHMM ii) mcHMM with LDA (Blei et al., 2003). For mcHMM method, we constructed visual words as described in Section 6.1 and represented each frame as two histograms of visual words. This representation was input to the model to learn parameters of the mcHMM model. In the augmented framework, mcHMM + LDA, the process of applying LDA to the input data can be viewed as a type of dimensionality reduction step since the number of topics are usually significantly smaller than the number of unique words. In our work, a frame is analogous to a document and visual words are analogous to words in a text document. Hence, in the combined method, we performed the additional step of using LDA to represent each frame as a histogram of topics. These reduced-dimension features were input to the mcHMM model. Gesture spotting was then performed by creating a spotting network made up of connected mcHMM models, one for each gesture learned, as explained in Section 5.3.

For the mcHMM model, we experimentally fixed the number of states to 10. The number of visual words was computed as the number of classes multiplied by a factor of 10, for example if the number of classes is 12, then then number of visual words generated will be 120. The dimensionality of the input features to the mcHMM model was the number of visual words representing one training sample. For the augmented model the dimension of the features was reduced by a factor of 1.25, that is in the previous example, the length of feature vector would be reduced from 120 to 96. All the above parameters were experientially found using the development set. The same values were then used for the validation and final sets.

6.3.1. ANALYSIS OF RESULTS

Table 4 shows the results of one-shot-learning on the ChaLearn data at the three different stages of the competition. We present results based on the two variants of our framework—the mcHMM model framework and the augmented mcHMM + LDA framework. Our results indicate that the framework augmented with LDA outperforms

Method	Data Set	edit distance
proposed mcHMM	Development	0.26336
proposed mcHMM + LDA	Development	0.2409
baseline	Validation	0.59978
proposed mcHMM	Validation	0.26036
proposed mcHMM + LDA	Validation	0.23328
Top Ranking Participant	Validation	0.20287
Top Ranking Participant	Final	0.09956
proposed mcHMM + LDA	Final	0.18465

Table 4: Results for ChaLearn gesture data set.

the unaugmented one, two out of three times. During implementation, the computational performance for the augmented framework was also significantly better than the unaugmented model due the reduced number of features needed for training and for inference. It is also interesting to observe how the edit distances reduced from the development phase through the final phase, dropping by up to six percentage points, due to parameter tuning. We placed fourth place in the final results of round 1 of the Chalearn 2012 gesture challenge using the augmented method.

7. Conclusion and Future Work

In the course of this paper, we have investigated the use of motion patterns (representing sub-actions) exhibited during different complex human activities. Using a language-motivated approach we developed a dynamic Bayesian model which combined the temporal ordering power of dynamic Bayesian networks with the automatic clustering power of hierarchical Bayesian models such as the LDA word-topic clustering model. We also showed how to use the Gibbs samples for rapid Bayesian inference of video segment clip category. Being a generative model, we can detect abnormal activities based on low likelihood measures. This framework was validated by its comparable performance on tests performed on the daily activities data set, a naturalistic data set involving everyday activities in the home.

We also investigated the use of a multichannel HMM as a generative probabilistic model for single activities and it performed comparably to the state-of-the-art activity classification techniques which are typically discriminative in nature, on two extreme data sets—the simplistic KTH, and the very complex and realistic HMDB data sets. An additional benefit of this framework was its usefulness for gesture spotting based on learning from only one, or few training examples. We showed how the use of the generative dynamic Bayesian model naturally lent itself to the spotting task, during inference. The efficacy of this model was shown by the results obtained from participating in ChaLearn Gesture Challenge where an implementation of the model finished top-5 in the competition.

In the future, we will consider using the visual words learned from a set of training videos to automatically segment a test video. The use of auto-detected video segments could prove useful both in activity classification and gesture spotting. It will also be interesting to explore the use of different descriptors available in the literature, in order to find those best-suited for representing naturalistic videos.

Acknowledgments

The authors wish to thank the associate editors and anonymous referees for all their advice about the structure, references, experimental illustration and interpretation of this manuscript. The work benefited significantly from our participation in the ChaLearn challenge as well as the accompanying workshops.

References

J.K. Aggarwal and M.S. Ryoo. Human activity analysis: A review. *ACM Computing Surveys*, 43:16:1–16:43, Apr 2011.

Yassine Benabbas, Adel Lablack, Nacim Ihaddadene, and Chabane Djeraba. Action Recognition Using Direction Models of Motion. In *Proceedings of the 2010 International Conference on Pattern Recognition*, pages 4295–4298, 2010.

Hakan Bilen, Vinay P. Namboodiri, and Luc Van Gool. Action recognition: A region based approach. In *Proceedings of the 2011 IEEE Workshop on the Applications of Computer Vision*, pages 294 –300, 2011.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003.

M. Bregonzio, Shaogang Gong, and Tao Xiang. Recognising action as clouds of space-time interest points. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1948–1955, 2009.

ChaLearn. ChaLearn Gesture Dataset (CGD2011), ChaLearn, California, 2011. URL http://gesture.chalearn.org/2011-one-shot-learning.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

Konstantinos G. Derpanis, Mikhail Sizintsev, Kevin Cannons, and Richard P. Wildes. Efficient action spotting based on a spacetime oriented structure representation. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1990–1997, 2010.

Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *Proceedings of the 2005 IEEE Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 65–72, 2005.

Andrew Gilbert, John Illingworth, and Richard Bowden. Action recognition using mined hierarchical compound features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):883–897, 2011.

Shaogang Gong and Tao Xiang. Recognition of group activities using dynamic probabilistic networks. In *Proceedings of the 2003 IEEE Conference on Computer Vision and Pattern Recognition*, pages 742–749 vol.2, 2003.

Gregor Heinrich. Parameter estimation for text analysis,. Technical report, University of Leipzig, 2008.

Timothy Hospedales, Shao-Gang Gong, and Tao Xiang. A Markov Clustering Topic Model for Mining Behaviour in Video. In *Proceedings of the 2009 International Conference on Computer Vision*, pages 1165–1172, 2009.

Alexander Kläser, Marcin Marszalek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *Proceedings of the 2008 British Machine Vision Conference*, 2008.

Adriana Kovashka and Kristen Grauman. Learning a hierarchy of discriminative space-time neighborhood features for human action recognition. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2046–2053, 2010.

H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the 2011 International Conference on Computer Vision*, 2011.

Ivan Laptev. On Space-Time Interest Points. *International Journal of Computer Vision*, 64:107–123, September 2005.

Ivan Laptev and Tony Lindeberg. Space-time interest points. In *Proceedings of the 2003 International Conference on Computer Vision*, pages 432–439, 2003.

Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning Realistic Human Actions From Movies. In *Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

Pyry Matikainen, Martial Hebert, and Rahul Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *Proceedings of the 2009 IEEE Workshop on Video-Oriented Object and Event Classification*, Sep 2009.

Pyry Matikainen, Martial Hebert, and Rahul Sukthankar. Representing Pairwise Spatial and Temporal Relations for Action Recognition. In *Proceedings of the 2010 European Conference on Computer Vision*, September 2010.

Ross Messing, Chris Pal, and Henry Kautz. Activity Recognition Using the Velocity Histories of Tracked Keypoints. In *Proceedings of the 2009 International Conference on Computer Vision*, 2009.

Pradeep Natarajan and Ramakant Nevatia. Coupled hidden semi markov models for activity recognition. In *Proceedings of the IEEE Workshop on Motion and Video Computing*, 2007.

Nam T. Nguyen, Dinh Q. Phung, and Svetha Venkatesh. Learning and detecting activities from movement trajectories using the hierarchical hidden markov models. In *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition*, pages 955–960, 2005.

Eric Nowak, Frederic Jurie, and Bill Triggs. Sampling strategies for bag-of-features image classification. In *Proceedings of the 2006 European Conference on Computer Vision*, pages 490–503, 2006.

Nuria Oliver, Eric Horvitz, and Ashutosh Garg. Layered representations for human activity recognition. In *Proceedings of the 2002 IEEE International Conference on Multimodal Interfaces*, pages 3–8, 2002.

Nuria M. Oliver, Barbara Rosario, and Alex P. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.

J. R. Rohlicek, W. Russell, S. Roukos, and H. Gish. Continuous hidden Markov modeling for speaker-independent word spotting. In *Proceedings of the 1989 International Conference on Acoustics, Speech, and Signal Processing*, pages 627–630, 1989.

R. Rose and D. Paul. A Hidden Markov Model based keyword recognition system. In *Proceedings of the 1990 International Conference on Acoustics, Speech, and Signal Processing*, 1990.

Christian Schüldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: A local svm approach. In *Proceedings of the 2004 International Conference on Pattern Recognition*, pages 32–36, 2004.

Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Procedings of the ACM International Conference on Multimedia*, pages 357–360, 2007.

University of Central Florida. *University of Central Florida, Computer Vision Lab*, 2010. URL http://server.cs.ucf.edu/~vision/data/UCF50.rar.

Heng Wang, Muhammad Muneeb Ullah, Alexander Kläser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *Proceedings of the 2009 British Machine Vision Conference*, sep 2009.

Heng Wang, Alexander Kläser, Cordelia Schmid, and Liu Cheng-Lin. Action Recognition by Dense Trajectories. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3169–3176, Jun 2011.

Geert Willems, Tinne Tuytelaars, and Luc Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *Proceedings of the 2008 European Conference on Computer Vision*, pages 650–663, 2008.

J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden Markov model. In *Proceedings of the 1992 IEEE Conference on Computer Vision and Pattern Recognition*, pages 379–385, 1992.

L. Yeffet and L. Wolf. Local trinary patterns for human action recognition. In *Proceedings of the 2009 International Conference on Computer Vision*, 2009.

Junsong Yuan, Zicheng Liu, and Ying Wu. Discriminative subvolume search for efficient action detection. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

JMLR: Workshop and Conference Proceedings 13:1589–1608, 2012 Published 5/12

A Model of the Perception of Facial Expressions of Emotion by Humans: Research Overview and Perspectives

Aleix Martinez ALEIX@ECE.OSU.EDU
Shichuan Du DUS@ECE.OSU.EDU
Department of Electrical and Computer Engineering
The Ohio State University
2015 Neil Avenue
Columbus, OH 43210, USA

Editor: Isabelle Guyon and Vassilis Athitsos

Abstract

In cognitive science and neuroscience, there have been two leading models describing how humans perceive and classify facial expressions of emotion—the continuous and the categorical model. The continuous model defines each facial expression of emotion as a feature vector in a face space. This model explains, for example, how expressions of emotion can be seen at different intensities. In contrast, the categorical model consists of C classifiers, each tuned to a specific emotion category. This model explains, among other findings, why the images in a morphing sequence between a happy and a surprise face are perceived as either happy or surprise but not something in between. While the continuous model has a more difficult time justifying this latter finding, the categorical model is not as good when it comes to explaining how expressions are recognized at different intensities or modes. Most importantly, both models have problems explaining how one can recognize combinations of emotion categories such as happily surprised versus angrily surprised versus surprise. To resolve these issues, in the past several years, we have worked on a revised model that justifies the results reported in the cognitive science and neuroscience literature. This model consists of C distinct continuous spaces. Multiple (compound) emotion categories can be recognized by linearly combining these C face spaces. The dimensions of these spaces are shown to be mostly configural. According to this model, the major task for the classification of facial expressions of emotion is precise, detailed detection of facial landmarks rather than recognition. We provide an overview of the literature justifying the model, show how the resulting model can be employed to build algorithms for the recognition of facial expression of emotion, and propose research directions in machine learning and computer vision researchers to keep pushing the state of the art in these areas. We also discuss how the model can aid in studies of human perception, social interactions and disorders.

Keywords: vision, face perception, emotions, computational modeling, categorical perception, face detection

1. Introduction

The face is an object of major importance in our daily lives. Faces tell us the identity of the person we are looking at and provide information on gender, attractiveness and age, among many others. Of primary interest is the production and recognition of facial expressions of emotion. Emotions play a fundamental role in human cognition (Damasio, 1995) and are thus essential in studies of cognitive science, neuroscience and social psychology. Facial expressions of emotion could also play a pivotal role in human communication (Schmidt and Cohn, 2001). And, sign languages use facial expressions to encode part of the grammar (Wilbur, 2011). It has also been speculated that expressions of emotion were relevant in human evolution (Darwin, 1872). Models of the perception of facial expressions of emotion are thus important for the advance of many scientific disciplines.

A first reason machine learning and computer vision researchers are interested in creating computational models of the perception of facial expressions of emotion is to aid studies in the above sciences (Martinez, 2003). Furthermore, computational models of facial expressions of emotion are important for the development of artificial intelligence (Minsky, 1988) and are essential in human-computer interaction (HCI) systems (Pentland, 2000).

Yet, as much as we understand how facial expressions of emotion are produced, very little is known on how they are interpreted by the human visual system. Without proper models, the scientific studies summarized above as well as the design of intelligent agents and efficient HCI platforms will continue to elude us. A HCI system that can easily recognize expressions of no interest to the human user is of limited interest. A system that fails to recognize emotions readily identified by us is worse.

In the last several years, we have defined a computational model consistent with the cognitive science and neuroscience literature. The present paper presents an overview of this research and a perspective of future areas of interest. We also discuss how machine learning and computer vision should proceed to successfully emulate this capacity in computers and how these models can aid in studies of visual perception, social interactions and disorders such as schizophrenia and autism. In particular, we provide the following discussion.

- A model of human perception of facial expressions of emotion: We provide an overview of the cognitive science literature and define a computational model consistent with it.
- Dimensions of the computational space: Recent research has shown that human used mostly shape for the perception and recognition of facial expressions of emotion. In particular, we show that configural features are of much use in this process. A configural feature is defined as a non-rotation invariant modeling of the distance between facial components; for example, the vertical distance between eyebrows and mouth.

- We argue that to overcome the current problems of face recognition algorithms (including identity and expressions), the area should make a shift toward a more shape-based modeling. Under this model, the major difficulty for the design of computer vision and machine learning systems is that of precise detection of the features, rather than classification. We provide a perspective on how to address these problems.

The rest of the paper is organized as follows. Section 2 reviews relevant research on the perception of facial expressions of emotion by humans. Section 3 defines a computational model consistent with the results reported in the previous section. Section 4 illustrates the importance of configural and shape features for the recognition of emotions in face images. Section 5 argues that the real problem in machine learning and computer vision is a detection one and emphasizes the importance of research in this domain before we can move forward with improved algorithms of face recognition. In Section 6, we summarize some of the implications of the proposed model. We conclude in Section 7.

2. Facial Expressions: From Production to Perception

The human face is an engineering marvel. Underneath our skin, a large number of muscles allow us to produce many configurations. The face muscles can be summarized as Action Unit (AU) (Ekman and Friesen, 1976) defining positions characteristic of facial expressions of emotion. These face muscles are connected to the motor neurons in the cerebral cortex through the corticobulbar track. The top muscles are connected bilaterally, while the bottom ones are connected unilaterally to the opposite hemisphere. With proper training, one can learn to move most of the face muscles independently. Otherwise, facial expressions take on predetermined configurations.

There is debate on whether these predetermined configurations are innate or learned (nature vs. nurture) and whether the expressions of some emotions is universal (Izard, 2009). By universal, we mean that people from different cultures produce similar muscle movements when expressing some emotions. Facial expressions typically classified as universal are joy, surprise, anger, sadness, disgust and fear (Darwin, 1872; Ekman and Friesen, 1976). Universality of emotions is controversial, since it assumes facial expressions of emotion are innate (rather than culturally bound). It also favors a categorical perception of facial expressions of emotion. That is, there is a finite set of predefined classes such as the six listed above. This is known as the *categorical model*.

In the categorical model, we have a set of C classifiers. Each classifier is specifically designed to recognize a single emotion label, such as surprise. Several psychophysical experiments suggest the perception of emotions by humans is categorical (Ekman and Rosenberg, 2005). Studies in neuroscience further suggest that distinct regions (or pathways) in the brain are used to recognize different expressions of emotion (Calder et al., 2001).

An alternative to the categorical model is the *continuous model* (Russell, 2003; Rolls, 1990). Here, each emotion is represented as a feature vector in a multidimen-

sional space given by some characteristics common to all emotions. One such model is Russell's 2-dimensional circumplex model (Russell, 1980), where the first basis measures pleasure-displeasure and the second arousal. This model can justify the perception of many expressions, whereas the categorical model needs to define a class (i.e., classifier) for every possible expression. It also allows for intensity in the perception of the emotion label. Whereas the categorical model would need to add an additional computation to achieve this goal (Martinez, 2003), in the continuous model the intensity is intrinsically defined in its representation. Yet, morphs between expressions of emotions are generally classified to the closest class rather than to an intermediate category (Beale and Keil, 1995). Perhaps more interestingly, the continuous model better explains the caricature effect (Rhodes et al., 1987; Calder et al., 1997), where the shape features of someone's face are exaggerated (e..g, making a long nose longer). This is because the farther the feature vector representing that expression is from the mean (or center of the face space), the easier it is to recognize it (Valentine, 1991).

In neuroscience, the multidimensional (or continuous) view of emotions was best exploited under the limbic hypothesis (Calder et al., 2001). Under this model, there should be a neural mechanism responsible for the recognition of all facial expressions of emotion, which was assumed to take place in the limbic system. Recent results have however uncovered dissociated networks for the recognition of most emotions. This is not necessarily proof of a categorical model, but it strongly suggests that there are at least distinct groups of emotions, each following distinct interpretations.

Furthermore, humans are only very good at recognizing a number of facial expressions of emotion. The most readily recognized emotions are happiness and surprise. It has been shown that joy and surprise can be robustly identified extremely accurately at almost any resolution (Du and Martinez, 2011). Figure 1 shows a happy expression at four different resolutions. The reader should not have any problem recognizing the emotion in display even at the lowest of resolutions. However, humans are not as good at recognizing anger and sadness and are even worse at fear and disgust.

A major question of interest is the following. Why are some facial configurations more easily recognizable than others? One possibility is that expressions such as joy and surprise involve larger face transformations than the others. This has recently proven not to be the case (Du and Martinez, 2011). While surprise does have the largest deformation, this is followed by disgust and fear (which are poorly recognized). Learning why some expressions are so readily classified by our visual system should facilitate the definition of the form and dimensions of the computational model of facial expressions of emotion.

The search is on to resolve these two problems. First, we need to determine the *form* of the computational space (e.g., a continuous model defined by a multidimensional space). Second, we ought to define the *dimensions* of this model (e.g., the dimensions of this multidimensional face space are given by configural features). In the following sections we overview the research we have conducted in the last several years leading to a solution to the above questions. We then discuss on the implications of this model. In particular, we provide a perspective on how machine learning and computer vision

Figure 1: Happy faces at four different resolutions. From left o right: 240 by 160, 120 by 80, 60 by 40, and 30 by 20 pixels. All images have been resized to a common image size for visualization.

researcher should move forward if they are to define models based on the perception of facial expressions of emotion by humans.

3. A Model of the Perception of Facial Expressions of Emotion

In cognitive science and neuroscience researchers have been mostly concerned with models of the perception and classification of the six facial expressions of emotion listed above. Similarly, computer vision and machine learning algorithms generally employ a face space to represent these six emotions. Sample feature vectors or regions of this feature space are used to represent each of these six emotion labels. This approach has a major drawback—it can only detect one emotion from a single image. In machine learning, this is generally done by a winner-takes-all approach (Torre and Cohn, 2011). This means that when a new category wants to be included, one generally needs to provide labeled samples of it to the learning algorithm.

Yet, everyday experience demonstrates that we can perceive more than one emotional category in a single image (Martinez, 2011), even if we have no prior experience with it. For example, Figure 2 shows images of faces expressing different surprises—happily surprised, angrily surprised, fearfully surprised, disgustedly surprised and the typically studied surprise.

If we were to use a continuous model, we would need to have a very large number of labels represented all over the space; including all possible types of surprises. This would require a very large training set, since each possible combination of labels would have to be learned. But this is the same problem a categorical model would face. In such a case, dozens if not hundreds of sample images for each possible category would be needed. Alternatively, Susskind et al. (2007) have shown that the appearance of a continuous model may be obtained from a set of classifiers defining a small number of categories.

Figure 2: Faces expressing different surprise. From left to right: happily surprised, sadly surprised, angrily surprised, fearfully surprised, disgustedly surprised, and surprise.

If we define an independent computational (face) space for a small number of emotion labels, we will only need sample faces of those few facial expressions of emotion. This is indeed the approach we have taken. Details of this model are given next.

Key to this model is to note that we can define new categories as linear combinations of a small set of categories. Figure 3 illustrates this approach. In this figure, we show how we can obtain the above listed different surprises as a linear combination of known categories. For instance, happily surprised can be defined as expressing 40% joy plus 60% surprise, that is, expression = .4 happy + .6 surprise. A large number of such expressions exist that are a combination of the six emotion categories listed above and, hence, the above list of six categories is a potential set of basic emotion classes. Also, there is some evidence form cognitive science to suggest that these are important categories for humans (Izard, 2009) Of course, one needs not base the model on this set of six emotions. This is an area that will undoubtedly attract lots of interest. A question of particular interest is to determine not only which basic categories to include in the model but how many. To this end both, cognitive studies with humans and computational extensions of the proposed model will be necessary, with the results of one area aiding the research of the other.

The approach described in the preceding paragraph would correspond to a categorical model. However, we now go one step further and define each of these face spaces as continuous feature spaces, Figure 3. This allows for the perception of each emotion at different intensities, for example, less happy to exhilarant (Neth and Martinez, 2010). Less happy would correspond to a feature vector (in the left most face space in the figure) closer to the mean (or origin of the feature space). Feature vectors farther from the

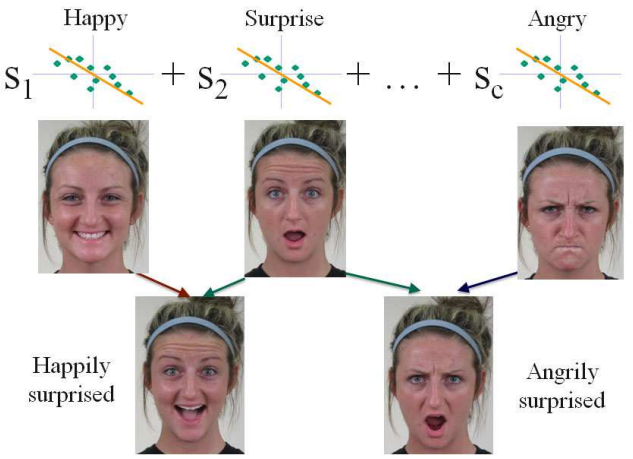


Figure 3: This figure shows how to construct linear combinations of known categories. At the top of the figure, we have the known or learned categories (emotions). The coefficients s_i determine the contribution of each of these categories to the final perception of the emotion.

mean would be perceived as happier. The proposed model also explains the caricature effect, because within each category the face space is continuous and exaggerating the expression will move the feature vector representing the expression further from the mean of that category.

Furthermore, the proposed model can define new terms, for example, “hatred" which is defined as having a small percentage of disgust and a larger percentage of anger; still linear. In essence, the intensity observed in this *continuous representation* defines the weight of the contribution of each basic category toward the final decision (classification). It also allows for the representation and recognition of a very large number of emotion categories without the need to have a categorical space for each or having to use many samples of each expression as in the continuous model.

The proposed model thus bridges the gap between the categorical and continuous ones and resolves most of the debate facing each of the models individually. To complete the definition of the model, we need to specify what defines each of the dimensions of the continuous spaces representing each category. We turn to this problem in the next section.

4. Dimensions of the Model

In the early years of computer vision, researchers derived several feature- and shape-based algorithms for the recognition of objects and faces (Kanade, 1973; Marr, 1976; Lowe, 1983). In these methods, geometric, shape features and edges were extracted

from an image and used to build a model of the face. This model was then fitted to the image. Good fits determined the class and position of the face.

Later, the so-called appearance-based approach, where faces are represented by their pixel-intensity maps or the response of some filters (e.g., Gabors), was studied (Sirovich and Kirby, 1987). In this alternative texture-based approach, a metric is defined to detect and recognize faces in test images (Turk and Pentland, 1991). Advances in pattern recognition and machine learning have made this the preferred approach in the last two decades (Brunelli and Poggio, 1993).

Inspired by this success, many algorithms developed in computer vision for the recognition of expressions of emotion have also used the appearance-based model (Torre and Cohn, 2011). The appearance-based approach has also gained momentum in the analysis of AUs from images of faces. The main advantage of the appearance-based model is that one does not need to predefine a feature or shape model as in the earlier approaches. Rather, the face model is inherently given by the training images.

The appearance-based approach does provide good results from near-frontal images of a reasonable quality, but it suffers from several major inherent problems. The main drawback is its sensitivity to image manipulation. Image size (scale), illumination changes and pose are all examples of this. Most of these problems are intrinsic to the definition of the approach since this cannot generalize well to conditions not included in the training set. One solution would be to enlarge the number of training images (Martinez, 2002). However, learning from very large data sets (in the order of millions of samples) is, for the most part, unsolved (Lawrence, 2005). Progress has been made in learning complex, non-linear decision boundaries, but most algorithms are unable to accommodate large amounts of data—either in space (memory) or time (computation).

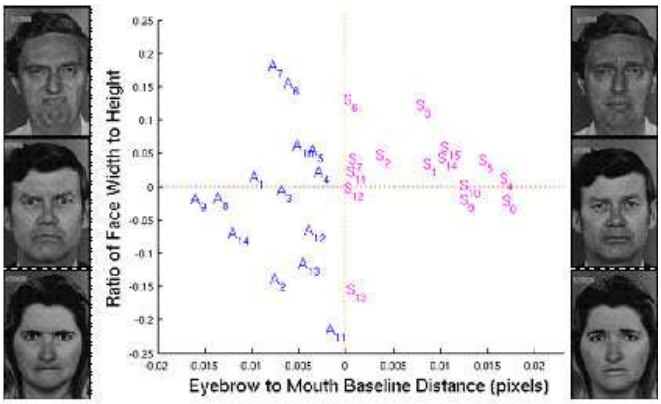
This begs the question as to how the human visual system solves the problem. One could argue that, throughout evolution, the homo genus (and potentially before it) has been exposed to trillions of faces. This has facilitated the development of simple, yet robust algorithms. In computer vision and machine learning, we wish to define algorithms that take a shorter time to learn a similarly useful image representation. One option is to decipher the algorithm used by our visual system. Research in face recognition of identity suggests that the algorithm used by the human brain is not appearance-based (Wilbraham et al., 2008). Rather, it seems that, over time, the algorithm has identified a set of robust features that facilitate rapid categorization (Young et al., 1987; Hosie et al., 1988; Barlett and Searcy, 1993).

This is also the case in the recognition of facial expressions of emotion (Neth and Martinez, 2010). Figure 4 shows four examples. These images all bear a neutral expression, that is, an expression associated to no emotion category. Yet, human subjects perceive them as expressing sadness, anger, surprise and disgust. The most striking part of this illusion is that these faces do not and cannot express any emotion, since all relevant AUs are inactive. This effect is called over-generalization (Zebrowitz et al., 2010), since human perception is generalizing the learned features defining these face spaces over to images with a different label.

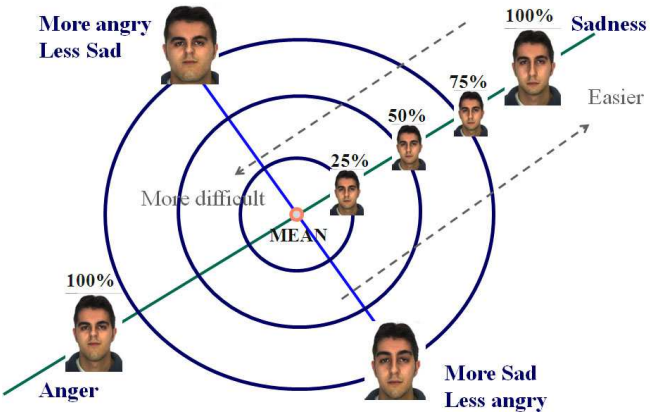
Figure 4: The four face images and schematics shown above all correspond to neutral expressions (i.e., the sender does not intend to convey any emotion to the receiver). Yet, most human subjects interpret these faces as conveying anger, sadness, surprise and disgust. Note that although these faces look very different from one another, three of them are actually morphs from the same (original) image.

The images in Figure 4 do have something in common though—they all include a configural transformation. What the human visual system has learned is that faces do not usually look like those in the image. Rather the relationship (distances) between brows, nose, mouth and the contour of the face is quite standard. They follow a Gaussian distribution with small variance (Neth and Martinez, 2010). The images shown in this figure however bear uncanny distributions of the face components. In the sad-looking example, the distance between the brows and mouth is larger than normal (Neth and Martinez, 2009) and the face is thinner than usual (Neth and Martinez, 2010). This places this sample face, most likely, outside the 99% confidence interval of all Caucasian faces on these two measures. The angry-looking face has a much-shorter-than-average brow to mouth distance and a wide face. While the surprise-looking face has a large distance between eyes and brows and a thinner face. The disgust-looking face has a shorter distance between brows, eyes, nose and mouth. These effects are also clear in the schematic faces shown in the figure.

Yet, configural cues alone are not sufficient to create an impressive, lasting effect. Other shape changes are needed. For example, the curvature of the mouth in joy or the opening of the eyes—showing additional sclera—in surprise. Note how the surprise-looking face in Figure 4 appears to also express disinterest or sleepiness. Wide-open eyes would remove these perceptions. But this can only be achieved with a shape change. Hence, our face spaces should include both, configural and shape features. It is important to note that configural features can be obtained from an appropriate representation of shape. Expressions such as fear and disgust seem to be mostly (if not solely) based on shape features, making recognition less accurate and more susceptible



(a)



(b)

Figure 5: (a) Shown here are the two most discriminant dimensions of the face shape vectors. We also plot the images of anger and sadness of Ekman and Friesen (1976). In dashed are simple linear boundaries separating angry and sad faces according to the model. The first dimension (distance between brows and mouth) successfully classifies 100% of the sample images. This continuous model is further illustrated in (b). Note that, in the proposed computational model, the face space defining sadness corresponds to the right-bottom quadrant, while that of anger is given by the left-top quadrant. The dashed arrows in the figure reflect the fact that as we move away from the "mean" (or norm) face, recognition of that emotion become easier.

to image manipulation. We have previously shown (Neth and Martinez, 2010) that configural cues are amongst the most discriminant features in a classical (Procrustes) shape representation, which can be made invariant to 3D rotations of the face (Hamsici and Martinez, 2009a).

Thus, each of the six categories of emotion (happy, sad, surprise, angry, fear and disgust) is represented in a shape space given by classical statistical shape analysis. First the face and the shape of the major facial components are automatically detected. This includes delineating the brows, eyes, nose, mouth and jaw line. The shape is then sample with d equally spaced landmark points. The mean (center of mass) of all the points is computed. The $2d$-dimensional shape feature vector is given by the x and y coordinates of the d shape landmarks subtracted by the mean and divided by its norm. This provides invariance to translation and scale. 3D rotation invariance can be achieved with the inclusion of a kernel as defined in Hamsici and Martinez (2009a). The dimensions of each emotion category can now be obtained with the use of an appropriate discriminant analysis method. We use the algorithm defined by Hamsici and Martinez (2008) because it minimizes the Bayes classification error.

As an example, the approach detailed in this section identifies the distance between the brows and mouth and the width of the face as the two most important shape features of anger and sadness. It is important to note that, if we reduce the computational spaces of anger and sadness to 2-dimensions, they are almost indistinguishable. Thus, it is possible that these two categories are in fact connected by a more general one. This goes back to our question of the number of basic categories used by the human visual system. The face space of anger and sadness is illustrated in Figure 5, where we have also plotted the feature vectors of the face set of Ekman and Friesen (1976).

As in the above, we can use the shape space defined above to find the two most discriminant dimensions separating each of the six categories listed earlier. The resulting face spaces are shown in Figure 6. In each space, a simple linear classifier in these spaces can successfully classify each emotion very accurately. To test this, we trained a linear support vector machine (Vapnik, 1998) and use the leave-one-out test on the data set of images of Ekman and Friesen (1976). Happiness is correctly classified 99% of the time. Surprise and disgust 95% of the time. Sadness 90% and anger 94%. While fear is successfully classified at 92%. Of course, adding additional dimensions in the feature space and using nonlinear classifiers can readily achieve perfect classification (i.e., 100%). The important point from these results is to note that simple configural features can *linearly* discriminate most of the samples in each emotion. These features are very robust to image degradation and are thus ideal for recognition in challenging environments (e.g., low resolution)—a message to keep in mind for the development of machine learning and computer vision systems.

5. Precise Detection of Faces and Facial Features

As seen thus far, human perception is extremely tuned to small configural and shape changes. If we are to develop computer vision and machine learning systems that can emulate this capacity, the real problem to be addressed by the community is that of *precise detection of faces and facial features* (Ding and Martinez, 2010). Classification is less important, since this is embedded in the detection process; that is, we want to precisely detect changes that are important to recognize emotions.

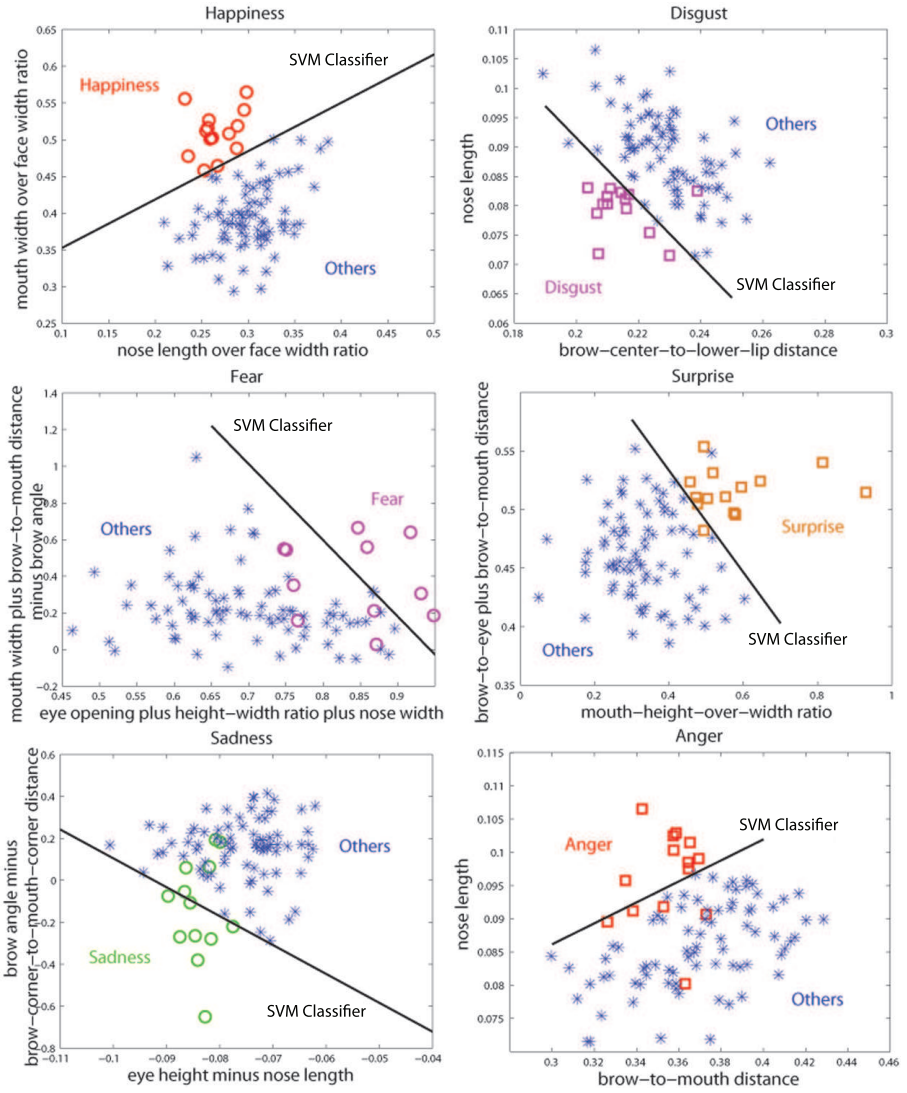


Figure 6: Shown in the above are the six feature spaces defining each of the six basic emotion categories. A simple linear Support Vector Machine (SVM) can achieve high classification accuracies; where we have used a one-versus-all strategy to construct each classifier and tested it using the leave-one-out strategy. Here, we only used two features (dimensions) for clarity of presentation. Higher accuracies are obtained if we include additional dimensions and training samples.

Figure 7: Two example of imprecise detections of a face with a state of the art algorithm.

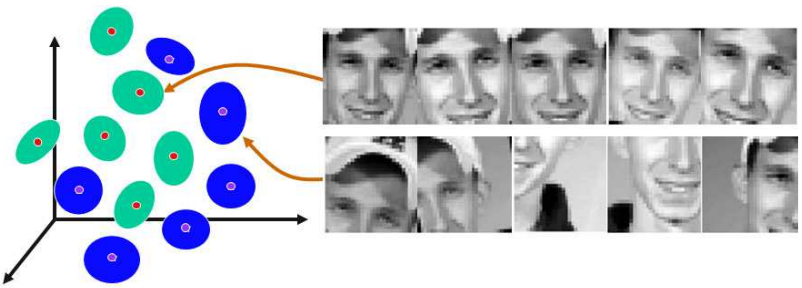


Figure 8: The idea behind the features versus context approach is to learn to discriminate between the feature we wish to detect (e.g., a face, an eye, etc.) and poorly detected versions of it. This approach eliminates the classical overlapping of multiple detections around the object of interest at multiple scales. At the same time, it increases the accuracy of the detection because we are moving away from poor detections and toward precise ones.

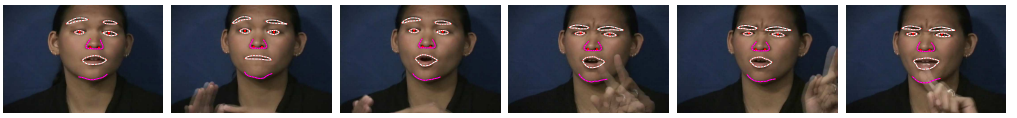


Figure 9: Precise detections of faces and facial features using the algorithm of (Ding and Martinez, 2010).

Most computer vision algorithms defined to date provide, however, inaccurate detections. One classical approach to detection is template matching. In this approach, we first define a template (e.g., the face or the right eye or the left corner of the mouth or any other feature we wish to detect). This template is learned from a set of sample images; for example, estimating the distribution or manifold defining the appearance (pixel map) of the object (Yang et al., 2002). Detection of the object is based on a window search. That is, the learned template is compared to all possible windows in the image. If the template and the window are similar according to some metric, then the bounding box defining this window marks the location and size (scale) of the face. The major drawback of this approach is that it yields imprecise detections of the learned object, because a window of an non-centered face is more similar to the learned template than a window with background (say, a tree). An example of this result is shown in Figure 7.

A solution to the above problem is to learn to discriminate between non-centered windows of the objects and well centered ones (Ding and Martinez, 2010). In this alternative, a non-linear classifier (or some density estimator) is employed to discriminate the region of the feature space defining well-centered windows of the objects and non-centered ones. We call these non-centered windows the context of the object, in the sense that these windows provide the information typically found around the object but do not correspond to the actual face. This features versus context idea is illustrated in Figure 8. This approach can be used to precisely detect faces, eyes, mouth, or any other facial feature where there is a textural discrimination between it and its surroundings. Figure 9 shows some sample results of accurate detection of faces and facial features with this approach.

The same features versus context idea can be applied to other detection and modeling algorithms, such as Active Appearance Models (AAM) (Cootes et al., 2001). AAM use a linear model—usually based on Principal Component Analysis (PCA)—to learn the relationship between the shape of an object (e.g., a face) and its texture. One obvious limitation is that the learned model is linear. A solution to this problem is to employ a kernel map. Kernel PCA is one option. Once we have introduced a kernel we can move one step further and use it to address additional issues of interest. A first capability we may like to add to a AAM is the possibility to work with three-dimensions. The second could be to omit the least-squares iterative nature of the Procrustes alignment required in most statistical shape analysis methods such as AAM. An approach that success-

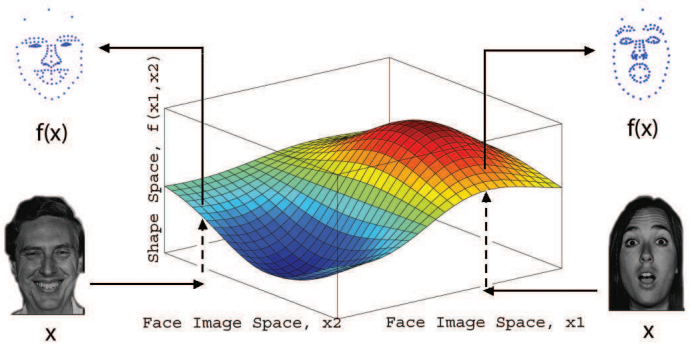


Figure 10: Manifold learning is ideal for learning mappings between face (object) images and their shape description vectors.

fully addresses these problem uses a set of kernels called Rotation Invariant Kernels (RIK) (Hamsici and Martinez, 2009a). RIK add yet another important advantage to shape analysis: they provide rotation invariance. Thus, once the shape is been mapped to the RIK space, objects (e.g., faces) are invariant to translation, scale and rotation. These kernels are thus very attractive for the design of AAM algorithms (Hamsici and Martinez, 2009b).

By now we know that humans are very sensitive to small changes. But we do not yet know how sensitive (or accurate). Of course, it is impossible to be pixel accurate when marking the boundaries of each facial feature, because edges blur over several pixels. This can be readily observed by zooming in the corner of an eye. To estimate the accuracy of human subjects, we performed the following experiment. First, we designed a system that allows users to zoom in at any specified location to facilitate delineation of each of the facial features manually. Second, we asked three people (herein referred to as judges) to manually delineate each of the facial components of close to $4,000$ images of faces. Third, we compared the markings of each of the three judges. The within-judge variability was (on average) 3.8 pixels, corresponding to a percentage of error of 1.2% in terms of the size of the face. This gives us an estimate of the accuracy of the manual detections. The average error of the algorithm of Ding and Martinez (2010) is 7.3 pixels (or 2.3%), very accurate but still far short of what humans can achieve. Thus, further research is needed to develop computer vision algorithms that can extract even more accurate detection of faces and its components.

Another problem is what happens when the resolution of the image diminishes. Humans are quite robust to these image manipulations (Du and Martinez, 2011). One solution to this problem is to use manifold learning. In particular, we wish to define a non-linear mapping $f(.)$ between the image of a face and its shape. This is illustrated in Figure 10. That is, given enough sample images and their shape feature vectors described in the preceding section, we need to find the function which relates the two.

Figure 11: Shape detection examples at different resolutions. Note how the shape estimation is almost as good regardless of the resolution of the image.

This can be done, for example, using kernel regression methods (Rivera and Martinez, 2012). One of the advantages of this approach is that this function can be defined to detect shape from very low resolution images or even under occlusions. Occlusions can be "learned" by adding synthetic occlusions or missing data in the training samples but leaving the shape feature vector undisturbed (Martinez, 2002). Example detections using this approach are shown in Figure 11.

One can go one step further and recover the three-dimensional information when a video sequence is available (Gotardo and Martinez, 2011a). Recent advances in non-rigid structure from motion allow us to recover very accurate reconstructions of both the shape and the motion even under occlusion. A recent approach resolves the nonlinearity of the problem using kernel mappings (Gotardo and Martinez, 2011b).

Combining the two approaches to detection defined in this section should yield even more accurate results in low-resolution images and under occlusions or other image manipulations. We hope that more research will be devoted to this important topic in face recognition.

The approaches defined in this section are a good start, but much research is needed to make these systems comparable to human accuracies. We argue that research in machine learning should address these problems rather than the typical classification one. A first goal is to define algorithms that can detect face landmarks very accurately even at low resolutions. Kernel methods and regression approaches are surely good

solutions as illustrated above. But more targeted approaches are needed to define truly successful computational models of the perception of facial expressions of emotion.

6. Discussion

In the real world, occlusions and unavoidable imprecise detections of the fiducial points, among others, are known to affect recognition (Torre and Cohn, 2011; Martinez, 2003). Additionally, some expressions are, by definition, ambiguous. Most importantly though seems to be the fact that people are not very good at recognizing facial expressions of emotion even under favorable condition (Du and Martinez, 2011). Humans are very robust at detection joy and surprise from images of faces; regardless of the image conditions or resolution. However, we are not as good at recognizing anger and sadness and are worst at fear and disgust.

The above results suggest that there could be three groups of expressions of emotion. The first group is intended for conveying emotions to observers. These expressions have evolved a facial construct (i.e., facial muscle positions) that is distinctive and readily detected by an observer at short or large distances. Example expressions in this group are happiness and surprise. A computer vision system—especially a HCI—should make sure these expressions are accurately and robustly recognized across image degradation. Therefore, we believe that work needs to be dedicated to make systems very robust when recognizing these emotions.

The second group of expressions (e.g., anger and sadness) is reasonably recognized at close proximity only. A computer vision system should recognize these expressions in good quality images, but can be expected to fail as the image degrades due to resolution or other image manipulations. An interesting open question is to determine why this is the case and what can be learned about human cognition from such a result.

The third and final group of emotions constitutes those at which humans are not very good recognizers. This includes expressions such as fear and disgust. Early work (especially in evolutionary psychology) had assumed that recognition of fear was primal because it served as a necessary survival mechanism (LeDoux, 2000). Recent studies have demonstrated much the contrary. Fear is generally poorly recognized by healthy human subjects (Smith and Schyns, 2009; Du and Martinez, 2011). One hypothesis is that expressions in this group have evolved for other than communication reasons. For example, it has been proposed that fear opens sensory channels (i.e., breathing in and wide open eye), while disgust closes them (i.e., breathing out and closed eyes) (Susskind et al., 2008). Under this model, the receiver has learned to identify those face configurations to some extent, but without the involvement of the sender—modifying the expression to maximize transmission of information through a noisy environment—the recognition of these emotions has remained poor. Note that people can be trained to detect such changes quite reliably (Ekman and Rosenberg, 2005), but this is not the case for the general population.

Another area that will require additional research is to exploit other types of facial expressions. Facial expressions are regularly used by people in a variety of setting. More research is needed to understand these. Moreover, it will be important to test the

model in natural occurring environments. Collection and handling of this data poses several challenges, but the research described in these pages serves as a good starting point for such studies. In such cases, it may be necessary to go beyond a linear combination of basic categories. However, without empirical proof for the need of something more complex than linear combinations of basic emotion categories, such extensions are unlikely. The cognitive system has generally evolved the simplest possible algorithms for the analysis or processing of data. Strong evidence of more complex models would need to be collected to justify such extensions. One way to do this is by finding examples that cannot be parsed by the current model, suggesting a more complex structure is needed.

It is important to note that these results will have many applications in studies of agnosias and disorders. Of particular interest are studies of depression or anxiety disorders. Depression afflicts a large number of people in the developed countries. Models that can help us better understand its cognitive processes, behaviors and patterns could be of great importance for the design of coping mechanisms. Improvements may also be possible if it were to better understand how facial expressions of emotion affect these people. Other syndromes such as autism are also of great importance these days. More children than ever are being diagnosed with the disorder (CDC, 2012; Prior, 2003). We know that autistic children do not perceive facial expressions of emotion as others do (Jemel et al., 2006) (but see Castelli, 2005). A modified computational model of the perception of facial expressions of emotion in autism could help design better teaching tools for this group and may bring us closer to understanding the syndrome.

There are indeed many great possibilities for machine learning researchers to help move these studies forward. Extending or modifying the modeled summarized in the present paper is one way. Developing machine learning algorithms to detect face landmark more accurately is another. Developing statistical tools that more accurately represent the underlying manifold or distribution of the data is yet another great way to move the state of the art forward.

7. Conclusions

In the present work we have summarized the development of a model of the perception of facial expressions of emotion by humans. A key idea in this model is to linearly combine a set of face spaces defining some basic emotion categories. The model is consistent with our current understanding of human perception and can be successfully exploited to achieve great recognition results for computer vision and HCI applications. We have shown how, to be consistent with the literature, the dimensions of these computational spaces need to encode configural and shape features.

We conclude that to move the state of the art forward, face recognition research has to focus on a topic that has received little attention in recent years—precise, detailed detection of faces and facial features. Although we have focused our study on the recognition of facial expressions of emotion, we believe that the results apply to most face recognition tasks. We have listed a variety of ways in which the machine learning

community can get involved in this research project and briefly discussed applications in the study of human perception and the better understanding of disorders.

Acknowledgments

This research was supported in part by the National Institutes of Health, grants R01 EY 020834 and R21 DC 011081.

References

J. C. Barlett and J. Searcy. Inversion and configuration of faces. *Cognitive Psychology*, 25(3):281–316, 1993.

J. M. Beale and F. C. Keil. Categorical effects in the perception of faces. *Cognition*, 57:217–239, 1995.

R. Brunelli and T. Poggio. Face recognition: features versus templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1042–1052, 1993.

A. J. Calder, A. W. Young, D. Rowland, and D. I. Perrett. Computer-enhanced emotion in facial expressions. *Proceedings of the Royal Society of London B*, 264:919–925, 1997.

A. J. Calder, A. D. Lawrence, and A. W. Young. Neuropsychology of fear and loathing. *Nature Review Neuroscience*, 2:352–363, 2001.

F. Castelli. Understanding emotions from standardized facial expressions in autism and normal development. *Autism*, 9:428–449, 2005.

CDC. Center for Disease Control and Prevention. Prevalence of autism spectrum disorders: Autism and developmental disabilities monitoring network, 14 sites, united states, 2008. *Morbidity and Mortality Weekly Report (MMWR)*, 61, 2012.

T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

A. R. Damasio. *Descartes' Error: Emotion, Reason, and the Human Brain*. G. P. Putnam's Sons, New York, 1995.

C. Darwin. *The Expression of the Emotions in Man and Animal*. J. Murray., London, 1872.

L. Ding and A. M. Martinez. Features versus context: An approach for precise and detailed detection and delineation of faces and facial features. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32:2022–2038, 2010.

S. Du and A. M. Martinez. The resolution of facial expressions of emotion. *Journal of Vision*, 11(13):24, 2011.

P. Ekman and W.V. Friesen. *Pictures of Facial Affect*. Consulting Psychologists Press, Palo Alto, CA, 1976.

P. Ekman and E.L. Rosenberg. *What the Face Reveals: Basic and Applied Studies of Spontaneous Expression Using the Facial Action Coding System (FACS)*. Oxford University Press, New York, 2nd edition, 2005.

P. F. U. Gotardo and A. M. Martinez. Computing smooth time-trajectories for camera and deformable shape in structure from motion with occlusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10):2051–2065, 2011a.

P. F. U. Gotardo and A. M. Martinez. Kernel non-rigid structure from motion. In *Proceedings of the IEEE International Conference on Computer Vision*, 2011b.

O. C. Hamsici and A. M. Martinez. Bayes optimality in linear discriminant analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30:647–657, 2008.

O. C. Hamsici and A. M. Martinez. Rotation invariant kernels and their application to shape analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31:1985–1999, 2009a.

O. C. Hamsici and A. M. Martinez. Active appearance models with rotation invariant kernels. In *IEEE Proc. International Conference on Computer Vision*, 2009b.

J. A. Hosie, H. D. Ellis, and N. D. Haig. The effect of feature displacement on the perception of well-known faces. *Perception*, 17(4):461–474, 1988.

C. E. Izard. Emotion theory and research: Highlights, unanswered questions, and emerging issues. *Annual Review of Psychology*, 60:1–25, 2009.

B. Jemel, L. Mottron, and M. Dawson. Impaired face processing in autism: Fact or artifact? *Journal of Autism and Developmental Disorders*, 36:91–106, 2006.

T. Kanade. *Picture Processing System by Computer Complex and Recognition of Human Faces*. PhD thesis, Kyoto University, Japan, November 1973.

N. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of Machine Learning Research*, 6:1783–1816, 2005.

J.E. LeDoux. Emotion circuits in the brain. *Annual Review of Nueroscience*, 23:155–184, 2000.

D. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31(3):355–395, 1983.

D. Marr. Early processing of visual information. *Philosophical Transactions of the Royal Society of London*, 275(942):483–519, 1976.

A. M. Martinez. Recognizing imprecisely localized, partially occluded and expression variant faces from a single sample per class. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(6):748–763, 2002.

A. M. Martinez. Matching expression variant faces. *Vision Research*, 43:1047–1060, 2003.

A. M. Martinez. Deciphering the face. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition, workshop*, 2011.

M. Minsky. *The Society of Mind*. Simon & Schuster, New York, N.Y., 1988.

D. Neth and A. M. Martinez. Emotion perception in emotionless face images suggests a norm-based representation. *Journal of Vision*, 9(1):1–11, 2009.

D. Neth and A. M. Martinez. A computational shape-based model of anger and sadness justifies a configural representation of faces. *Vision Research*, 50:1693–1711, 2010.

A. Pentland. Looking at people: Sensing for ubiquitous and wearable computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):107–119, 2000.

M. Prior. Is there an increase in the prevalence of autism spectrum disorders? *Journal of Paediatrics and Child Health*, 39:81–82, 2003.

G. Rhodes, S. Brennan, and S. Carey. Identification and ratings of caricatures: implications for mental representations of faces. *Cognitive Psychology*, 19:473–497, 1987.

S. Rivera and A. M. Martinez. Learning shape manifolds. *Pattern Recognition*, 45(4): 1792–1801, 2012.

E. T. Rolls. A theory of emotion, and its application to understanding the neural basis of emotion. *Cognition and Emotion*, 4:161–190, 1990.

J. A. Russell. A circumplex model of affect. *J. Personality Social. Psych.*, 39:1161–1178, 1980.

J. A. Russell. Core affect and the psychological construction of emotion. *Psychological Review*, 110:145–172, 2003.

K.L. Schmidt and J.F. Cohn. Human facial expressions as adaptations: Evolutionary questions in facial expression. *Yearbook of Physical Anthropology*, 44:3–24, 2001.

L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *J. Optical Soc. Am. A*, 4:519–524, 1987.

F.W. Smith and P.G. Schyns. Smile through your fear and sadness: Transmitting and identifying facial expression signals over a range of viewing distances. *Psychology Science*, 20(10):1202–1208, 2009.

J. Susskind, D. Lee, A. Cusi, R. Feinman, W. Grabski, and A.K. Anderson. Expressing fear enhances sensory acquisition. *Nature Neuroscience*, 11(7):843–850, 2008.

J.M. Susskind, G. Littlewort, M.S. Bartlett, and A.K. Anderson J. Movellanb. Human and computer recognition of facial expressions of emotion. *Neuropsychologia*, 45: 152–162, 2007.

F. Dela Torre and J. F. Cohn. Facial expression analysis. In Th. B. Moeslund, A. Hilton, V. Kruger, and L. Sigal, editors, *Guide to Visual Analysis of Humans: Looking at People*, pages 377–410. Springer, 2011.

M. Turk and A. Pentland. Eigenfaces for recognition. *J. Cognitive Neuroscience*, 3: 71–86, 1991.

T. Valentine. A unified account of the effects of distinctiveness, inversion, and race in face recognition. *Quarterly Journal of Experimental Psychology A: Human Experimental Psychology*, 43:161–204, 1991.

V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, NY, 1998.

D. A. Wilbraham, J. C. Christensen, A. M. Martinez, and J. T. Todd. Can low level image differences account for the ability of human observers to discriminate facial identity? *Journal of Vision*, 8(5):1–12, 2008.

R. B. Wilbur. Nonmanuals, semantic operators, domain marking, and the solution to two outstanding puzzles in asl. In *Nonmanuals in Sign Languages*. John Benjamins, 2011.

M.-H. Yang, D. J. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(1):34–58, 2002.

A. W. Young, D. Hellawell, and D. C. Hay. Configurational information in face perception. *Perception*, 16(6):747–759, 1987.

L.A. Zebrowitz, M. Kikuchi, and J.M. Fellous. Facial resemblance to emotions: Group differences, impression effects, and race stereotypes. *Journal of Personality and Social Psychology*, 98(2):175–189, 2010.

JMLR: Workshop and Conference Proceedings 13:2589–2615, 2012 Published 9/12

Finding Recurrent Patterns from Continuous Sign Language Sentences for Automated Extraction of Signs

Sunita Nayak SNAYAK@TAAZ.COM
Taaz Inc.
4250 Executive Square, Suite 420
La Jolla, CA 92037 USA
Kester Duncan KKDUNCAN@CSE.USF.EDU
Sudeep Sarkar SARKAR@CSE.USF.EDU
Department of Computer Science & Engineering
University of South Florida
Tampa, FL 33620, USA
Barbara Loeding BARBARA@USF.EDU
Department of Special Education
University of South Florida
Lakeland, FL 33803, USA

Editor: Isabelle Guyon

Abstract

We present a probabilistic framework to automatically learn models of recurring signs from multiple sign language video sequences containing the vocabulary of interest. We extract the parts of the signs that are present in most occurrences of the sign in context and are robust to the variations produced by adjacent signs. Each sentence video is first transformed into a multidimensional time series representation, capturing the motion and shape aspects of the sign. Skin color blobs are extracted from frames of color video sequences, and a probabilistic relational distribution is formed for each frame using the contour and edge pixels from the skin blobs. Each sentence is represented as a trajectory in a low dimensional space called the space of relational distributions. Given these time series trajectories, we extract signemes from multiple sentences concurrently using iterated conditional modes (ICM). We show results by learning single signs from a collection of sentences with one common pervading sign, multiple signs from a collection of sentences with more than one common sign, and single signs from a mixed collection of sentences. The extracted signemes demonstrate that our approach is robust to some extent to the variations produced within a sign due to different contexts. We also show results whereby these learned sign models are used for spotting signs in test sequences.

Keywords: pattern extraction, sign language recognition, signeme extraction, sign modeling, iterated conditional modes

1. Introduction

Sign language research in the computer vision community has primarily focused on improving recognition rates of signs either by improving the motion representation and similarity measures (Yang et al., 2002; Al-Jarrah and Halawani, 2001; Athitsos et al., 2004; Cui and Weng, 2000; Wang et al., 2007; Bauer and Hienz, 2000) or by adding linguistic clues during the recognition process (Bowden et al., 2004; Derpanis et al., 2004). Ong and Ranganath (2005) presented a review of the automated sign language research and also highlighted one important issue in continuous sign language recognition. While signing a sentence, there exists transitions of the hands between two consecutive signs that do not belong to either sign. This is called movement epenthesis (Liddell and Johnson, 1989). This needs to be dealt with first before dealing with any other phonological issues in sign language (Ong and Ranganath, 2005). Most of the existing work in sign language assumes that the training signs are already available and often signs used in the training set are the isolated signs with the boundaries chopped off, or manually selected frames from continuous sentences. The ability to recognize isolated signs does not guarantee the recognition of signs in continuous sentences. Unlike isolated signs, a sign in a continuous sentence is strongly affected by its context in the sentence. Figure 1 shows two sentences 'I BUY TICKET WHERE?' and 'YOU CAN BUY THIS FOR HER' with a common sign 'BUY' between them. The frames representing the sign 'BUY' and the neighboring signs are marked. The unmarked frames between the signs indicate the frames corresponding to movement epenthesis. It can be observed that the same sign 'BUY' is preceded and succeeded by movement epenthesis that depends on the end and start of the preceding and succeeding sign respectively. The movement epenthesis also affects how the sign is signed. This effect makes the automated extraction, modeling and recognition of signs from continuous sentences more difficult when compared to just plain gestures, isolated signs, or finger spelling.

In this paper, we address the problem of automatically extracting the part of a sign that is most common in all occurrences of the sign, and hence expected to be robust with respect to the variation of adjacent signs. These common parts can be used for spotting or recognition of signs in continuous sign language sentences. They can also be used by sign language experts for teaching or studying variations between instances of signs in continuous sign language sentences, or in automated sign language tutoring systems. Furthermore, they can be used even in the process of translating sign language videos directly to spoken words.

In a related work inspired by the success of the use of phonemes in speech recognition, the authors sought to extract common parts in different instances of a sign and thus arrive at a phoneme-analogue for signs (Bauer and Kraiss, 2002). But unlike speech, sign language does not have a completely defined set of phonemes. Hence, we consider extracting commonalities at the sentence and sub-sentence level.

A different but a closely related problem is the extraction of common subsequences, also called motifs, from very long multiple gene sequences in biology (Bailey and Elkan, 1995; Lawrence et al., 1993; Pevzner and Sze, 2000; Rigoutsos and Floratos,

(a) Continuous Sentence 'I BUY TICKET WHERE?'

(b) Continuous Sentence 'YOU CAN BUY THIS FOR HER'

Figure 1: Movement epenthesis in sign language sentences. Frames corresponding to the common sign 'BUY' are marked in red. Signs adjacent to BUY are marked in magenta. Frames between marked frames represent movement epenthesis that is, the transition between signs. Note that the sign itself is also affected by having different signs preceding or following it.

1998). Lawrence et al. (1993) used a Gibbs sampling approach based on discrete matches or mismatches of subsequences that were strings of symbols of gene sequences. Bailey and Elkan (1995) used expectation maximization to find common subsequences in univariate biopolymer sequences. In biology, researchers deal with univariate discrete sequences, and hence their algorithms are not always directly applicable to other multivariate continuous domains in time series like speech or sign language. Some researchers tried to symbolize a continuous time series into discrete sequences and used existing algorithms from bioinformatics. For example, Chiu et al. (2003) symbolized the time series into a sequence of symbols using local approximations and used random projections to extract common subsequences in noisy data. Tanaka et al. (2005) extended their work by performing principal component analysis on the multivariate time series data and projected them onto a single dimension and symbolized the data into discrete sequences. However, it is not always possible to get all the important information in the first principal component alone. Further extending his work, Duchene et al.

(2007) find recurrent patterns from multivariate discrete data using time series random projections.

Due to the inherent continuous nature of many time series data like gesture and speech, new methods were developed that do not require approximating the data to a sequence of discrete symbols. Denton (2005) used a continuous random-walk noise model to cluster similar substrings. Nayak et al. (2005) and Minnen et al. (2007) use continuous multivariate sequences and dynamic time warping to find distances between the substrings. Oates (2002); Nayak et al. (2005) and Nayak et al. (2009a) are among the few works in finding recurrent patterns that address non-uniform sampling of time series. The recurrent pattern extraction approach proposed in this paper is based on multivariate continuous time series, uses dynamic time warping to find distances between substrings, and handles length variations of common patterns.

Following the success of Hidden Markov Models (HMMs) in speech recognition, they were used by sign language researchers (Vogler and Metaxas, 1999; Starner and Pentland, 1997; Bowden et al., 2004; Bauer and Hienz, 2000; Starner et al., 1998) for representing and recognizing signs. However, HMMs require a large number of training data and unlike speech, data from native signers is not as easily available as speech data. Hence, non-HMM-based approaches have been used (Farhadi et al., 2007; Nayak et al., 2009a; Yang et al., 2010; Buehler et al., 2009; Nayak et al., 2009b; Oszust and Wysocki, 2010; Han et al., 2009). In this paper, we use a continuous trajectory representation of signs in a multidimensional space and use dynamic time warping to match subsequences. The relative configuration of the two hands and face in each frame is represented by a relational distribution (Vega and Sarkar, 2003; Nayak et al., 2005), which in itself is a probability density function. The motion dynamics of the signer is captured as changes in the relational distributions. It also allows us to interpolate motion, if required, for data sets with lower frame capture rates. It should also be noted that, unlike many of the previous works in sign language that perform tracking of the hands using 3D magnetic trackers or color gloves (Fang et al., 2004; Vogler and Metaxas, 2001; Wang et al., 2002; Ma et al., 2000; Cooper and Bowden, 2009), our representation does not require tracking and relies on skin segmentation.

We present a Bayesian framework to extract the common subsequences or signemes from all the given sentences simultaneously. Figure 2 depicts the overview of our approach. With this framework, we can extract the first most common sign, the second most common sign, the third most common sign and so on. We represent each sentence as a trajectory in a multi-dimensional space that implicitly captures the shape and motion in the video. Skin color blobs are extracted from frames of color video, and a relational distribution is formed for each frame using the edge pixels in the skin blobs. Each sentence is then represented as a trajectory in a low dimensional space called the space of relational distributions, which is arrived at by performing principal component analysis (PCA) on the relational distributions. There are other alternatives to PCA that are possible and discussed in Nayak et al. (2009b). The other choices do not change the nature of the signeme finding approach, they only affect the quality of the features. The starting locations $(a_1, ...a_n)$ and widths $(w_1, ...w_n)$ of the candidate signemes in all

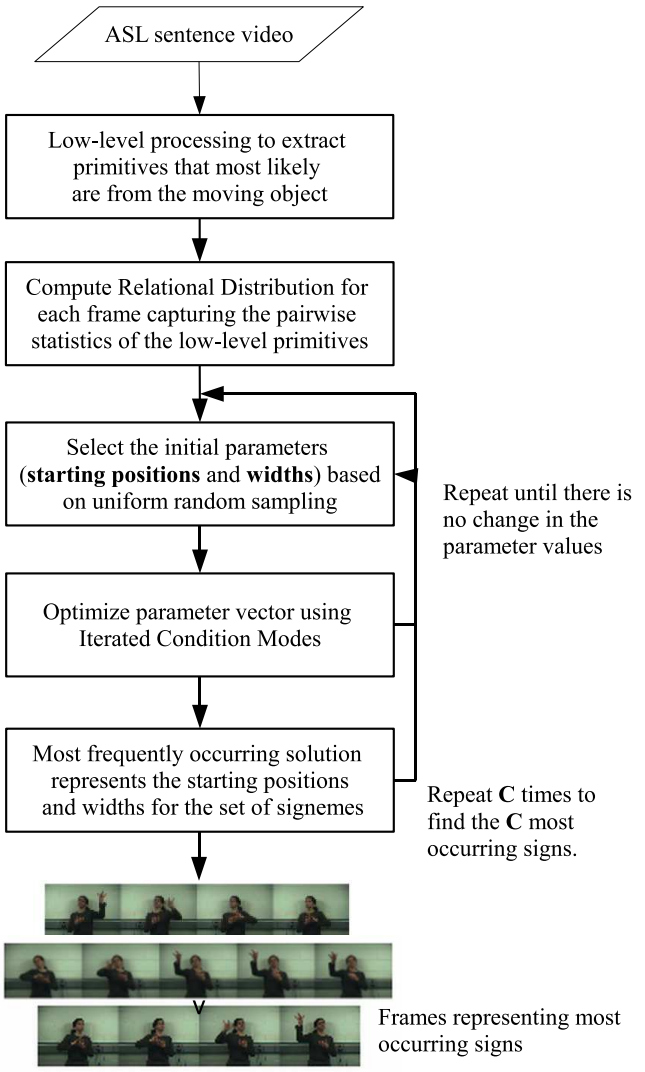


Figure 2: Overview of our approach. Each of the n sentences is represented as a sequence in the Space of Relational Distributions, and common patterns are extracted using iterated conditional modes (ICM). The parameter set $\{a_1, w_1, ...a_n, w_n\}$ is initialized using uniform random sampling and the conditional density corresponding to each sentence is updated in a sequential manner.

the n sentences are together represented by a parameter vector. The starting locations are initialized with random starting locations, based on uniform random sampling from each sentence, and the initial width values are randomly selected from a given range of values. The parameter vector is updated sequentially by sampling the starting point

and width of the possible signeme in each sentence from a joint conditional distribution that is based on the locations and widths of the target possible signeme in all other sentences. The process is iterated till the parameter values converge to a stable solution. Monte Carlo approaches like Gibbs sampling (Robert and Casella, 2004; Gilks et al., 1998; Casella and George, 1992), which is a special case of the Metropolis-Hastings algorithm (Chib and Greenberg, 1995) can be used for global optimization while updating the parameter vector by performing importance sampling on the conditional probability distribution. However, this has a high burn-in period.

In this paper, we adopt a greedy approach based on the use of iterated conditional modes (ICM) (Besag, 1986). ICM converges much faster than a Gibbs sampler, but is known to be largely dependent on the initialization. We overcome this limitation by performing ICM a number of times equal to the average length of the n sentences, with different initializations. The most frequently occurring solution from all the ICM runs is considered as the final solution.

The work in this paper builds on the work of Nayak et al. (2009a) and is different in multiple respects. We propose a system that is generalized to extract more than one common sign from a collection of sentences (first most common sign, second most common sign and so on), whereas in the previous work, only single signs were extracted. We also extract single signs from a mixed collection of sentences where there are more than one common sign in context. In addition to this, we present a more in-depth exposition of the underlying theory.

The contributions of this paper can be summarized as follows: (i) we present an unsupervised approach to automatically extract parts of signs that are robust to the variation of adjacent signs simultaneously from multiple sign language sentences, (ii) our approach does not consider all possible parameter combinations, instead samples each of them in a sequential manner until convergence, which saves a lot of computation, (iii) we show results on extracting signs from plain color videos of continuous sign language sentences without using any color gloves or magnetic trackers, and (iv) we show results whereby the learned signs are used for spotting signs in test sequences.

We organize the paper as follows. Section 2 presents a short review of relational distributions. In Section 3, we present the definition of signeme and then formulate the problem of finding signemes from a given set of sequences in a probabilistic framework. We describe how we solve it using iterated conditional modes. It is then followed by a description of our experiments and results in Section 4. Finally, Section 5 concludes the paper and discusses possible future work.

2. Relational Distributions

We use relational distributions to capture the global and relative configuration of the hands and the face in an image. Motion is then captured as the changes in the relational distributions. They were originally introduced by Vega and Sarkar (2003) for human gait recognition. They have also been used before for representing sign language sentences without the use of color gloves or magnetic trackers (Nayak et al., 2005, 2009b). We briefly review them here in this section.

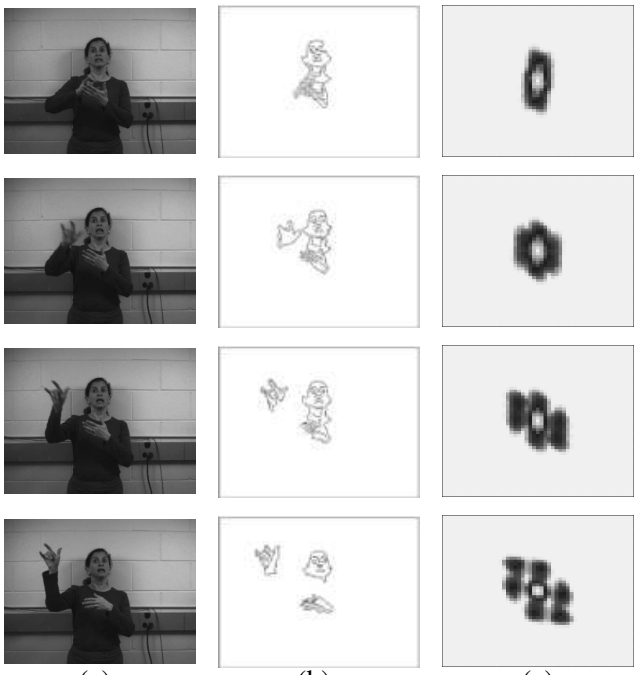


Figure 3: Variations in relational distributions with motion. (a) Motion sequence. (b) Edge pixels from the skin color blobs. (c) Relational distributions constructed from the low level features (edge pixels) of the images in the motion sequence. The horizontal axis of the relational distribution represents the horizontal distance between the edge pixels and its vertical axis represents the vertical distance between edge pixels.

How do we capture the global configuration of the object? We start with low-level primitives that are most likely to come from the articulated object. The exact nature of the low-level primitives can vary. Some common choices include edges, salient points, Gabor filter outputs and so on. We use edges in this work. We start from some level of segmentation of the object from the scene. These processes are fairly standard and have been used widely in gesture and sign recognition. They may involve color-based segmentation, skin-color segmentation, or background subtraction. In this work, we perform skin-color segmentation using histogram-based Bayesian classification (Phung et al., 2005). We use the contours of the skin blobs and Canny edges within the blobs as our low-level image primitives. The global configuration is captured by considering the relationships between these primitives.

We use the distance between two primitives in the vertical and horizontal directions (dx, dy) as relational attributes. Let vector $\mathbf{u} = \{dx, dy\}$ represent the vector of relational attributes. The joint probability function $P(\mathbf{u})$ then describes the distribu-

tion of primitives within an image and captures the shape of the pattern in the image. This probability is called a *relational distribution*. It captures the global configuration of the low-level primitives. Figure 3(c) illustrates how motion is captured using relational distributions. It shows the top view of the distributions. The region near to center represents points closer to each other, for example, the edge points within the face or within the hand, while farther from center represents the farther away points, for example, the relationship between edge points of a hand and the face. Notice the change in the relational distribution as the signer moves one of her hands. To be able to discriminate symmetrically opposite motion, we maintain the signs (or directions) of the horizontal and vertical distances between the edge pixels in each ordered pair. This leads to representing the probability distribution in a four quadrant system. Given that these relational distributions exhibit complicated shapes that are difficult to be modeled readily using a combination of simple shaped distributions such as Gaussian mixtures, we adopt non-parametric histogram-based representation. For better discrimination of the probabilities, we do not add counts to the center of the histogram which represents the distance of the edge pixels from itself or very close adjacent pixels. Each bin then counts the pairs of edge pixels between which the horizontal and vertical distances each lie in some fixed range that depends on the location of the bin in the histogram.

In our experiments, we found that an empirically-determined fixed histogram size of 51×51 was sufficient. The above range is then defined using linear mapping between the image size and the histogram size, for example, image size along the horizontal direction corresponds to half the histogram size in the horizontal direction. One could use histogram bin size optimization techniques for optimizing the histograms, but we do not address them in this paper. We then reduce the dimensionality of the relational distributions by performing PCA on the set of relational distributions from all the input sentences and retain the number of dimensions required to keep a certain percentage of energy, typically 95%. The new subspace arrived at is called the space of relational distributions (SoRD). Each video sequence is thus represented as a sequence of points in the SoRD space.

Note that the choice of the relational distribution is not a central requirement for the signeme learning process discussed in this paper. We use relational distributions to enable us to work with pure video data, without the use of markers or colored gloves. If magnetic markers or colored gloves are available then one could use their attributes to construct a different feature space and consider trajectories in them. One advantage of our representation is that the face and head locations are implicitly taken into account in addition to the hands. In short, the first step of the process is to construct a time series representation in an appropriate feature space.

3. Problem Formulation

Sign language sentences are series of signs. Figure 4 illustrates the traces of the first vs. second dimension in the feature space, of three sentences S_1, S_2 and S_3 with only one common sign, R, among them. The signeme represents the portion of the sign that is most similar across the sentences.

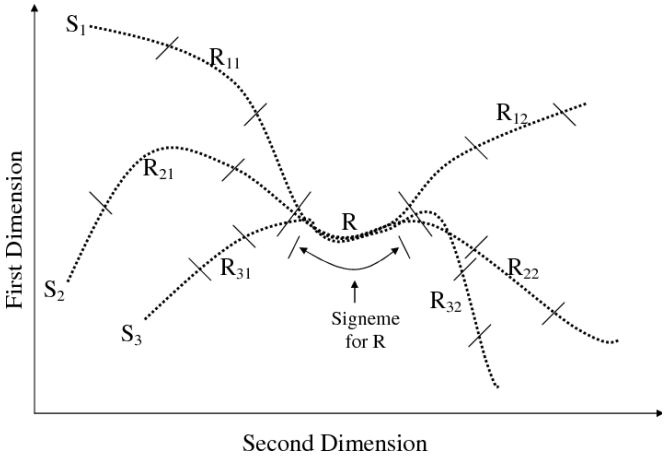


Figure 4: Concept of signemes. First vs. second dimensions of sentences S_1 with signs R_{11}, R, R_{12} in order, S_2 with signs R_{21}, R, R_{22} and S_3 with signs R_{31}, R, R_{32}. The common sign is R. The portion of R that is most similar across sentences is the signeme representative of R.

$\{S_1, \cdots, S_n\}$	Set of n sentences with at least one common sign present in all the sentences. The index within a sentence could represent time or arc length in configuration shape space
L_i	Length of sentence S_i
$s_{a_j}^{w_j}$	Subsequence of sentence S_j starting from index a_j to $a_j + w_j - 1$. We may sometimes use $s_{j,a}^{w}$ to make explicit the j-th index if it is not represented along with any other superscript or subscript of this term.
A, B	Possible choices of width for signemes of a sign include all integers from A to B. The values of A and B are decided based on the dynamics involved in the sign.
θ	Set of parameters $\{a_1, w_1, \cdots, a_n, w_n\}$ defining a set of substrings of the given sentences
$\theta_{(a_i)}$	Set of all parameters *excluding* the parameter a_i. We have similar interpretations for $\theta_{(w_i)}$ or $\theta_{(i)}$.
$d(x,y)$	Distance between the subsequences x and y based on a mapping found using dynamic time warping (DTW). This distance has to be calculated carefully so that it is not biased towards finding short subsequences only.

Table 1: Notations

Table 1 defines the notations that will be used in this paper. We formulate the signeme extraction problem as finding the most recurring patterns among a set of n sentences $\{S_1, \cdots, S_n\}$, that have at least one common sign present in all the sentences.

The commonality concept underlying the definition of a signeme can be cast in terms of distances. Let $s_{a_i}^{w_i}$ represent a substring from the sequence S_i consisting of the points with indices $a_i, \cdots a_i + w_i - 1$, and $d(x,y)$ denote the distance between two substrings x and y based on dynamic time warping. We define the set of signemes to be the set of substrings denoted by $\{s_{a_1}^{w_1}, \cdots, s_{a_n}^{w_n}\}$ that is most similar among all possible substrings from the given set of sentences. In the generalized case where C

most common signs are sought, the set of signemes are defined as $\{s_{a_{11}}^{w_{11}}, s_{a_{12}}^{w_{11}}, \cdots, s_{a_{nC}}^{w_{nC}}\}$. In theory, C can extend to the number of words in the shortest sentence.

Let $\theta = \{a_1, w_1, \cdots a_n, w_n\}$ denote the parameter set representing a set of substrings, at least one from each of the n sentences, and θ_m denote the parameter set representing the target set of signemes in the n sentences. We find θ_m using the probabilistic framework of Equation 1.

$$\theta_m = \arg\max_{\theta} p(\theta) \tag{1}$$

Note that $p(\theta)$ is a probability over the space of all possible substrings. We define this probability to be a function of the inter-substring distances in Equation 2:

$$p(\theta) = \frac{g(\theta)}{\sum_\theta g(\theta)}. \tag{2}$$

The term $g(\theta)$ is defined in Equation 3 as follows:

$$g(\theta) = \exp\left(-\beta \sum_{i=1}^{n} \sum_{j=1}^{n} d(s_{a_i}^{w_i}, s_{a_j}^{w_j})\right) \tag{3}$$

with β being a positive constant.

Note that $g(\theta)$ varies inversely with the summation of the pair-wise distances of all the subsequences given by θ. Also note that $p(\theta)$ is hard to compute or even sample from because it is computationally expensive to compute the denominator in Equation 2, as it involves the summation over all possible parameter combinations. β acts as a scale parameter, which controls the slopes of the peaks in the probability space. It can also be looked upon as the smoothing parameter. If probability sampling algorithms like Gibbs sampling (Casella and George, 1992) are used in later steps, then the rate of convergence would be determined by this parameter.

Let θ_i represent the parameters from the i^{th} sentence, that is, $\{a_i, w_i\}$ and $\theta_{(i)}$ represent the rest of the parameters, $\{a_1,w_1 \cdots a_{i-1},w_{i-1},a_{i+1},w_{i+1} \cdots a_n,w_n\}$. To make sampling easier, we construct a *conditional* density function of the parameters from each sentence, that is, θ_i, *given* the values of the rest of the parameters, that is, $\theta_{(i)}$. In other words, we construct a probability density function of the possible starting points and widths in each sentence, given the estimated starting points and widths of the common pattern in all other sentences, that is, $f(\theta_i|\theta_{(i)})$. Of course, this conditional density function has to be *derived* from the joint density function specified in Eq. 2. This is outlined in Equation 4 as follows:

$$f(\theta_i|\theta_{(i)}) \quad = \quad \frac{p(\theta)}{p(\theta_{(i)})} = \frac{p(\theta)}{\sum_{\theta_i} p(\theta)} = \frac{g(\theta)}{\sum_{\theta_i} g(\theta)}. \tag{4}$$

Since the normalization to arrive at this conditional density function involves summation over one parameter, it is now easier to compute and sample from. The specific

form for this conditional density function using the dynamic time warping (DTW) distances as described in Equation 5 is

$$f(\theta_i|\theta_{(i)}) = \frac{\exp\left(-\beta \sum_{k=1}^{n} d(s_{a_i}^{w_i}, s_{a_k}^{w_k})\right)}{\sum_{\theta_i} \exp\left(-\beta \sum_{k=1}^{n} d(s_{a_i}^{w_i}, s_{a_k}^{w_k})\right)}. \tag{5}$$

Note that the distance terms that do not involve a_i and w_i, that is, do not involve the i-th sentence appear both in the numerator and the denominator and so cancel out. For notational convenience, this is sometimes represented using conditional g functions described below in Equation 6 as:

$$f(\theta_i|\theta_{(i)}) = \frac{g(\theta_i|\theta_{(i)})}{\sum_{\theta_i} g(\theta_i|\theta_{(i)})}, \tag{6}$$

where $g(\theta_i|\theta_{(i)}) = \exp\left(-\beta \sum_{k=1}^{n} d(s_{a_i}^{w_i}, s_{a_k}^{w_k})\right)$.

3.1. Distance Measure

The distance function d in the above equations needs to be chosen carefully such that it is not biased towards the shorter subsequences. Here, we briefly describe how we compute the distance between two substrings using dynamic time warping. Let l_1 and l_2 represent the length of the two substrings and $e(i,j)$ represent the Euclidean distance between the i^{th} data point from the first substring and the j^{th} data point from the second substring. Let D represent the score matrix of size $(l_1+1)\times(l_2+1)$. The 0^{th} row and 0^{th} column of D are initialized to infinity, except $D(0,0)$, which is initialized to 0. The rest of the score matrix, D, is completed using the following recursion of Equation 7:

$$D(i,j) = e(i,j) + \min\{D(i-1,j), D(i-1,j-1), D(i,j-1)\}, \tag{7}$$

where $1 \le i \le l_1$ and $1 \le j \le l_2$. The optimal warp path is then traced back from $D(l_1,l_2)$ to $D(0,0)$. The distance measure between the two substrings is then given by $D(l_1,l_2)$ normalized by the length of the optimal warping path.

3.2. Parameter Estimation

In order to extract the common signs from a given set of sign language sentences, we need to compute θ_i for each of the sentences sequentially. Gibbs sampling (Casella and George, 1992) is a Markov Chain Monte Carlo approach (Gilks et al., 1998) that allows us to sample the conditional probability density $f(\theta_i|\theta_{(i)})$ for all the sequences sequentially and then iterate the whole process until convergence. Gibbs sampling results in a global optimum, but its convergence is very slow. The burn-in period is typically thousands of iterations. Therefore, we perform the optimization using iterated conditional modes (ICM), first proposed by Besag (1986). ICM has much faster convergence, but it is also known to be heavily dependent on the initialization. We address this limitation

by running the optimization multiple times with different initializations and choosing the most frequently occurring solution as the final solution.

Algorithm 1: ITERATED CONDITIONAL MODES($\{a_1^0, w_1^0, \cdots, a_n^0, w_n^0\}$)

comment: Choose $\{a_1, w_1, \cdots, a_n, w_n\}$ that maximizes distribution $p(a_1, w_1, \cdots, a_n, w_n)$

comment: Initialization:

$\theta_0 \leftarrow \{a_1^0, w_1^0, \cdots, a_n^0, w_n^0\}$

repeat

- **for** $i \leftarrow 0$ **to** n
  - **do**
    - **comment:** Jointly sample a_i, w_i. L_i is the length of sequence S_i
    - **for** $w_i \leftarrow A$ **to** B
      - **do**
        - **for** $a_i \leftarrow 0$ **to** $L_i - w_i + 1$
          - **do** $g(a_i, w_i|\theta_{(a_i,w_i)}) \leftarrow \exp\left(-\beta \sum_{k=1}^{n} d(s_{a_i}^{w_i}, s_{a_k}^{w_k})\right)$
    - **comment:** Normalize
    - **for** $w_i \leftarrow A$ **to** B
      - **do**
        - **for** $a_i \leftarrow 0$ **to** $L_i - w_i + 1$
          - **do** $f(a_i, w_i|\theta_{(a_i,w_i)}) \leftarrow \frac{g(a_i,w_i|\theta_{(a_i,w_i)})}{\sum_{a_i,w_i} g(a_i,w_i|\theta_{(a_i,w_i)})}$
    - $a_i, w_i \leftarrow$ ARG MAX $(f(a_i, w_i|\theta_{(a_i,w_i)}))$

until CHANGE IN PARAMETERS$(\{a_1, w_1, \cdots, a_n, w_n\}) == 0$

Algorithm 1 outlines the process of ICM to extract the common patterns or signemes from a set of sentences with a given initial parameter vector. We aim to select the set of parameters that maximizes the probability $p(\theta)$ or $p(a_1, w_1, \cdots, a_n, w_n)$. We do that by estimating each of the parameters $a_1, w_1, \cdots a_n, w_n$ in a sequential manner. Since we expect the starting location and width of a subsequence representing the common sign to be strongly correlated, we estimate a_i and w_i jointly. First we compute $g(\theta_i|\theta_{(i)})$ that is, $g(a_i, w_i|\theta_{(a_i,w_i)})$ from which we compute the conditional density functions $f(\theta_i|\theta_{(i)})$ that is, $f(a_i, w_i|\theta_{(a_i,w_i)})$. Note that it involves a summation over a_i and w_i only, which involves much less computation than that required for computing $p(\theta)$ which involves a summation over $a_1, w_1, \cdots a_n, w_n$. The values for a_i and w_i are updated with those that maximize the conditional density $f(\theta_i|\theta_{(i)})$. The process is carried out sequentially for $i = 1$ to n, and then repeated iteratively till the values of the parameter vector $\{a_1, w_1, a_2, w_2, \cdots a_n, w_n\}$ do not change any more.

Figure 5 depicts the sampling process for a single iteration, r. Note the conditional and sequential nature of sampling from various sentences within the single iteration. In Figure 6, we show an example of how the conditional probability $f(\theta_{a_i,w_i}|\theta_{(a_i,w_i)})$ changes for the first seven sentences from a given set of fourteen video sentences containing a common sign 'DEPART'. The vertical axis in the probabilities represents the

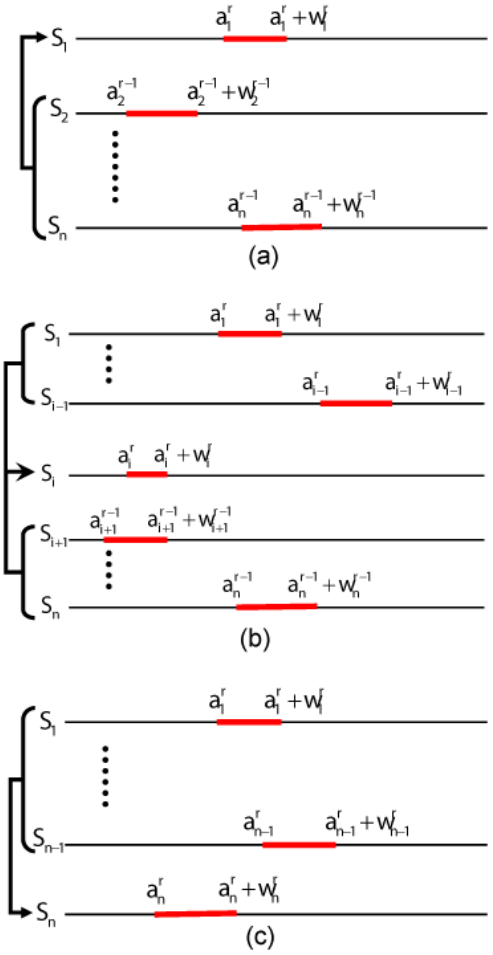


Figure 5: Sequential update of the parameter values using ICM. (a), (b) and (c) respectively show the parameter updates in the first sentence, the i^{th} and the n^{th} sentences. In the r^{th} iteration, the parameters of the common sign in i^{th} sentence is computed based on the parameter values of the previous $(i-1)$ sentences obtained in the same iteration, and those of the $(i+1)^{th}$ to n^{th} sentences obtained in the previous, that is, the $(r-1)^{th}$ iteration.

starting locations and the horizontal axis represents the possible widths. The brighter regions represent a higher probability value. Note that the probabilities are spread out in the first iteration for each sentence and it slowly converges to a fixed starting location for each of them. They remain more spread out across the horizontal (width) axis because we vary the width only in a small range of A to B for each sign, that is decided based on the amount of motion present in the sign.

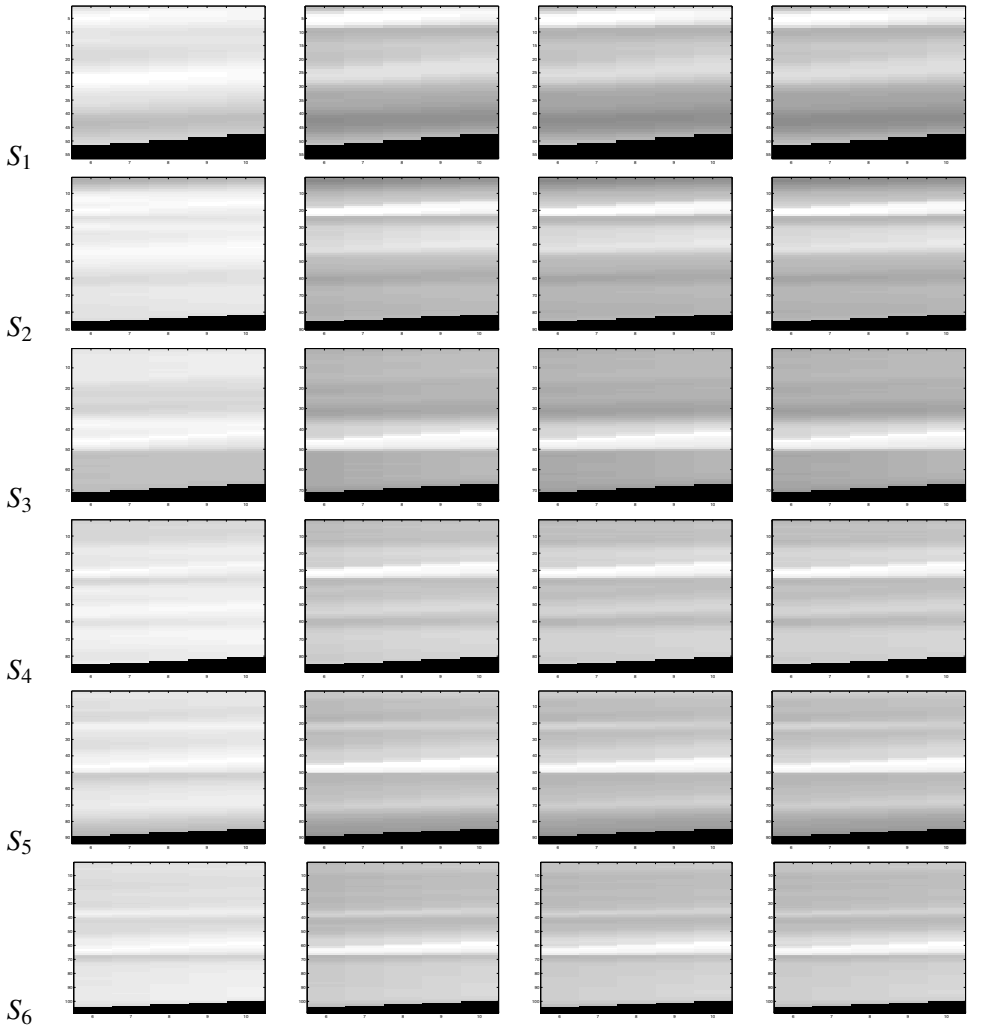


Figure 6: Convergence of the conditional probability density $f(\theta_i|\theta_{(i)})$ for sentences $S_1 ... S_6$ from a given set of sentences $S_1 ... S_{14}$. The brighter regions represent a higher probability value. The vertical axis in the probabilities represents the starting locations and the horizontal axis represents the possible widths. Note that the probabilities are spread out in the first iteration and it slowly converges to a particular starting location. They are still spread across the horizontal (width) axis because we vary the width only in a small range that is decided based on the amount of motion present in the sign.

Figure 7 plots the typical convergence of the parameter values in a single ICM run. It plots the norm of difference between consecutive parameter vectors versus the parameter vector update count, which is incremented each time a parameter is sampled or selected from the probability distribution $f(\theta_i|\theta_{(i)})$. It shows that ICM converges

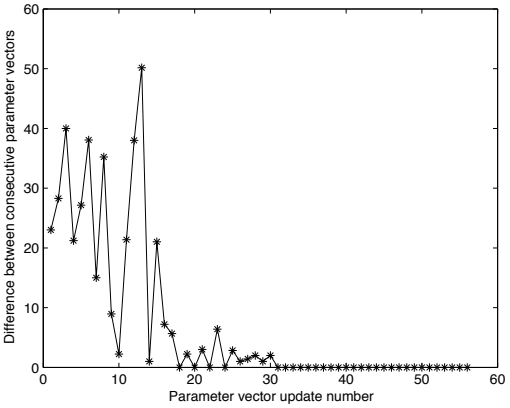


Figure 7: Convergence of values of the parameter set. The above plot shows the norm of the difference between two consecutive parameter vectors representing the set of starting points and widths of the common subsequence in the given set of sequences. It shows the typical convergence with a given initialization vector. ICM is repeated with multiple initializations and the most frequently occurring solution is considered as the final solution.

in less than $56/14 = 4$ iterations. This, in turn, also indicates the local nature of the optimization achieved with ICM. The initialization is very important in this case. In the next subsection, we describe how we address this problem.

3.3. Sampling Starting Points For ICM

In order to address the local convergence nature of ICM, we adopt a uniform random sampling-based approach. We start by randomly assigning values to the parameter vector θ. The width w_i^0 is obtained by sampling a width value based on uniform random distribution from the set of all possible widths in a given range $[A,B]$. The value for a_i^0 is obtained by sampling a starting point based on uniform random distribution from the set of all possible starting points in the i^{th} sequence, that is, from the set $\{1 \cdots (L_i - w_i^0 + 1)\}$.

Different initial parameter vectors are obtained by independently sampling the sentences multiple times. ICM is run using each initial parameter vector generated and the most common solution is considered as the final solution. The uniform sampling of the frames in the sentences for selecting the starting locations ensures the whole parameter space is covered uniformly. The number of times we sample the initial parameter vector and run the ICM algorithm decides how densely we cover the whole parameter space. We run it the number of times equal to the average number of frames in each sentence from the given set of sentences for extracting the sign. One could choose to run a multiple of the average number of times as well, but we found the average number

to be sufficient to show the stability of the solution in our experiments. Algorithm 2 presents the process as a pseudocode.

Algorithm 2: EXTRACT SIGNEMES($L_1, \cdots L_n, A, B$)

comment: Generate multiple initialization vectors and call ICM with each of them.

$N = \text{MEAN}(L_1, L_2, \cdots, L_n)$
for $j \leftarrow 1$ **to** N
do $\begin{cases} \textbf{for } i \leftarrow 1 \textbf{ to } n \\ \quad \textbf{do} \begin{cases} w_i^0 = \text{UNIFORM}(A \cdots B) \\ a_i^0 = \text{UNIFORM}(1 \cdots L_i - w_i^0 + 1) \end{cases} \\ \{a_1^j, w_1^j, \cdots, a_n^j, w_n^j\} = \text{ITERATED CONDITIONAL MODES}(a_1^0, w_1^0, \cdots, a_n^0, w_n^0) \end{cases}$
for $i \leftarrow 1$ **to** n
do $\begin{cases} \textbf{comment:} \text{ Assign most frequently occurring value as the final value.} \\ w_i = \text{MODE}(w_i^j) \\ a_i = \text{MODE}(a_i^j) \end{cases}$

For extracting the sign 'DEPART' from 14 sentences, we had 89 frames per sentence on an average. Hence we ran 89 different ICM runs for extracting the common subsequence representing 'DEPART'. Figure 8 shows the plots of histograms of start and end location of the sign in each of the 14 sentences from the 89 runs. It should be noted that in most of the sentences, more than 50% of the total number of runs result in the same solution.

4. Experiments And Results

In this section, we present visual and quantitative results of our approach for extracting signemes from video sequences representing sentences from American Sign Language. We first describe the data set used then present the results of the automatic common pattern extraction.

4.1. Data Set

Our data set consists of 155 American Sign Language (ASL) video sequences organized into 12 groups (collections) based on the vocabulary (word that pervades the sentences of the group). For instance, the 'DEPART' group is comprised of all the sentences containing the word 'DEPART', the 'PASSPORT' group is comprised of all the sentences containing the word 'PASSPORT' and so on. The breakdown of these 'pure' groups and the number of sentences (sequences) in each are as follows.

- DEPART - 14 sentences

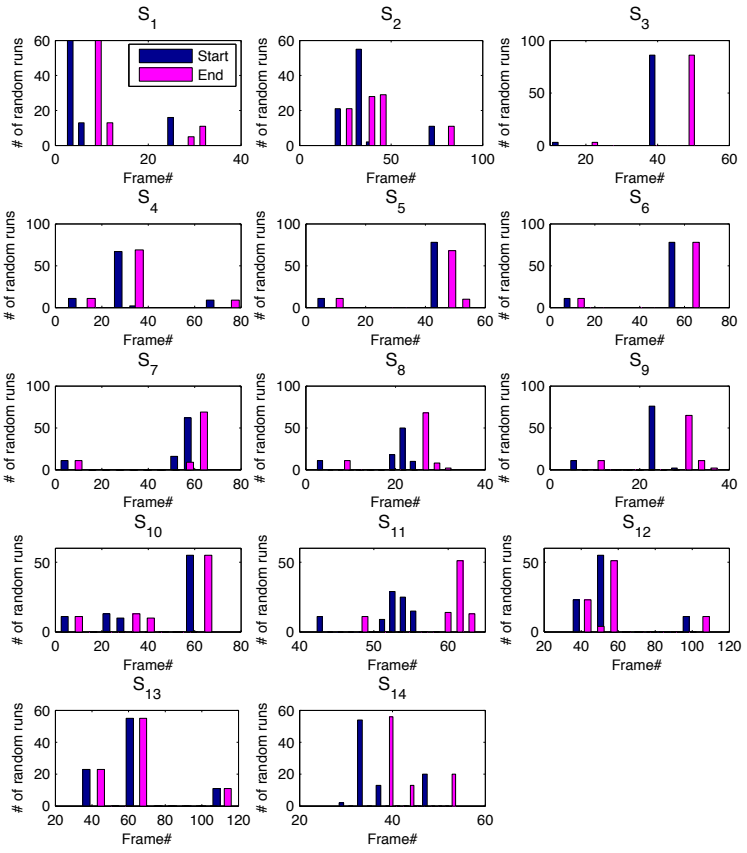


Figure 8: Histograms showing the start and end locations of signs extracted from 14 different sentences using multiple ICM runs. The initial parameter vector for each ICM run was chosen independently using uniform random sampling. As it can be seen the start and end points found by most of the runs converge to the same solution (denoted by single high bars in most of sentences). The legend shown in the plot for the first sentence, S_1, holds for other sentences as well.

- BAGGAGE - 14 sentences
- CANT - 14 sentences
- BUY - 11 sentences
- SECURITY - 16 sentences
- HAVE - 6 sentences
- MOVE - 11 sentences

- TIME - 14 sentences
- FUTURE - 12 sentences
- TABLE - 13 sentences
- PASSPORT - 14 sentences
- TICKET - 16 sentences

This data set was used to extract 12 common subsequences when we searched for the first most common sign, and 24 common subsequences when we searched for the second most common sign. We also organized the video sequences into 10 groups by combining two 'pure' groups of sentences as described above. This was used to investigate the power of our framework for selecting the common sequences in a 'mixed' collection. The breakdown of these 'mixed' groups and the number of sentences in each are as follows:

- DEPART (14 sentences) + BAGGAGE (14 sentences)
- CANT (14 sentences) + BUY (11 sentences)
- TIME (14 sentences) + TABLE (13 sentences)
- PASSPORT (14 sentences) + TICKET (16 sentences)
- SECURITY (16 sentences) + FUTURE (12 sentences)
- MOVE (11 sentences) + HAVE (6 sentences)
- BUY (11 sentences) + TABLE (13 sentences)
- DEPART (14 sentences) + FUTURE (12 sentences)
- BAGGAGE (14 sentences) + TICKET (16 sentences)
- SECURITY (16 sentences) + PASSPORT (14 sentences)

All of the signs were performed by the same signer with plain clothing and background. The video sequences were captured at 25 frames per second with a frame resolution of 490×370.

4.2. Common Pattern Extraction Results

In this section, we present the results of our method for extracting common patterns from sign language sentences. We first present results for extracting the single most common sign and multiple common signs from the 'pure' sentence groups, followed by results for the most common patterns from the 'mixed' groups.

4.2.1. EXTRACTING THE MOST COMMON PATTERN

We perform extraction of the most common patterns from the 'pure' sentence groups. We possess a priori knowledge of the most common word due to the organization of the sentence groups. However, our goal is to extract the most common sequences automatically. As an example, Figure 9 depicts the result of extraction of the sign 'DEPART' from 14 video sequences. It plots the SoRD first dimension coefficients of the

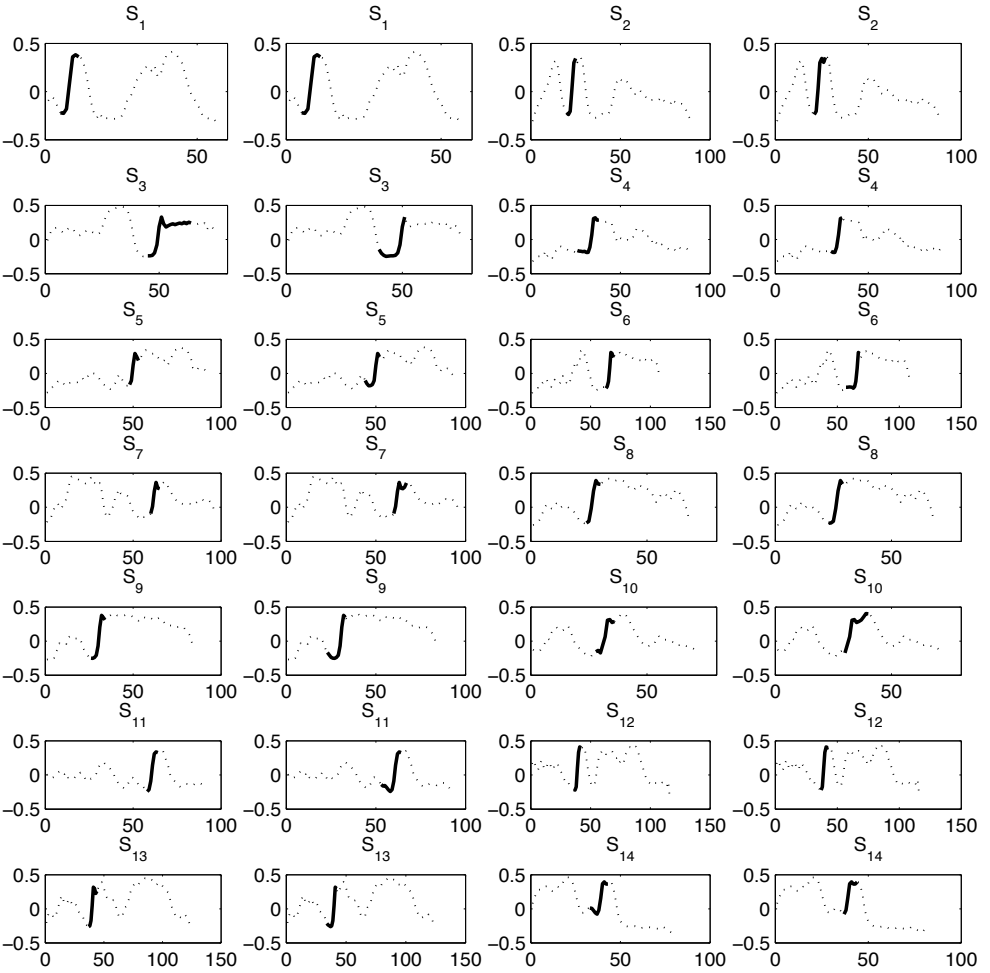
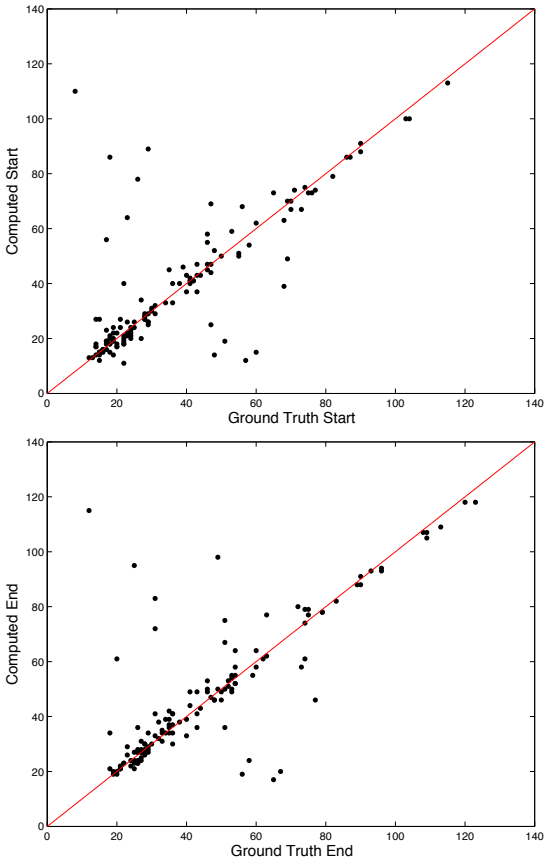


Figure 9: The first dimension of the video sequences containing a common sign 'DEPART'. The sequences are indicated by the dotted curves and the solid lines on each of them indicate the common pattern or signeme. The odd columns represent the ground truth and the even columns show the results.

frames vs. the frame number for each sentence. The highlighted portions represent

the signeme. The odd columns show the ground truth and the even columns show the corresponding results. As can be seen, the extracted patterns and the corresponding ground truth patterns are quite similar, except for a few frames at the beginning and end of the some of the patterns. Note that since we deal with continuous video sequences, a difference of one or two frames between the ground truth and the extracted pattern is not considered a problem.



(a) Video Start Point Estimation
(b) Video End Point Estimation

Figure 10: Extraction of the most common patterns or signemes from the 'pure' sentence groups. The closer the points are to the diagonal, the closer the result is to the ground truth.

Figure 10(a) shows the scatter plot of the ground truth start positions vs. the estimated start positions of the pattern extracted from each of the 155 sentences in the video data set. Figure 10(b) shows the corresponding scatter plot for the end position

of the patterns in the sentences. As can be seen most of the points in the scatter plots lie along the diagonal. This indicates that very few of the extracted patterns are wrong. Incorrect results correspond to the points positioned far from the diagonal. Figures 11

(a) BUY

(b) CANT

(c) DEPART

(d) FUTURE

(e) MOVE

Figure 11: Signemes extracted from sentences

and 12 show one instance of the signeme extracted from group of sentences.

4.2.2. EXTRACTING MULTIPLE COMMON SIGNS

In this section we present some visual results for the extraction of the two most common signs from the 'pure' groups of sentences. We focused on extracting only two signs because the shortest ASL sentence contained two signs. Figure 13 shows the results for the two most common signs extracted from the sentence 'BAGGAGE THERE NOT MINE THERE'. The extracted subsequences correspond to the ASL words 'BAGGAGE' and

(f) PASSPORT

(g) SECURITY

(h) TICKET

(i) TIME

(j) TABLE

Figure 12: Signemes extracted from sentences

'MINE'. Consequently, the word 'BAGGAGE' appears in all the 14 sentences of the group, whereas the word 'MINE' (or 'MY') shows up in 11 sentences coinciding with what was expected. Similarly, Figure 14 shows the results for the two most common signs extracted from the sentence 'MY PASSPORT THERE STILL GOOD THERE'. The extracted subsequences correspond to the ASL words 'MY' and 'PASSPORT'. The word 'MY' appears in all the 11 sentences of the group, whereas the word 'PASSPORT' appears in all 14 sentences. These results are encouraging.

4.2.3. EXTRACTING THE MOST COMMON PATTERNS FROM MIXED SENTENCES

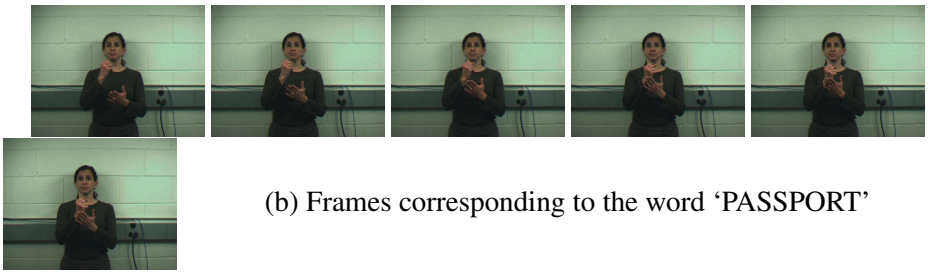
We perform extraction of the most common patterns from the collection of 'mixed' sentences as outlined in Section 4.1. Figure 15(a) shows the scatter plot of the ground truth

(a) Frames corresponding to the word 'BAGGAGE'

(b) Frames corresponding to the word 'MINE'

Figure 13: Extraction of the two most common patterns or signemes from the sentence 'BAGGAGE THERE NOT MINE THERE'.

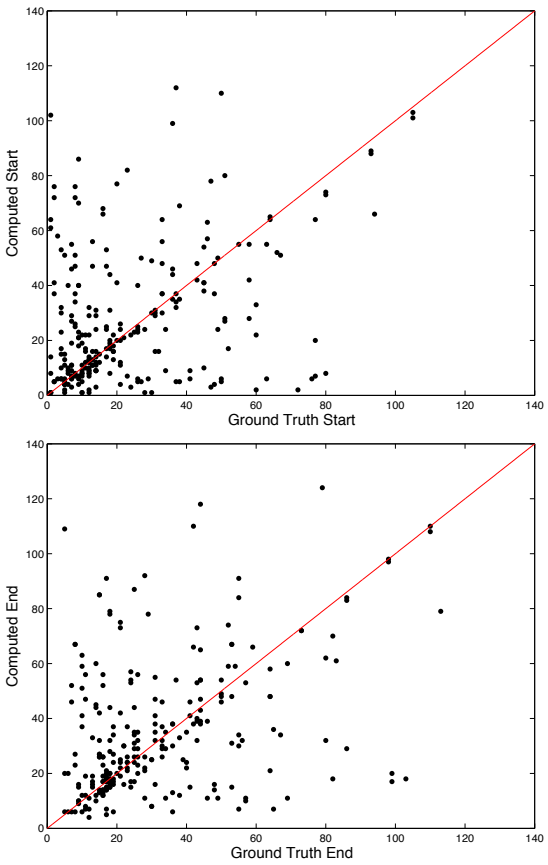
(a) Frames corresponding to the word 'MY'



(b) Frames corresponding to the word 'PASSPORT'

Figure 14: Extraction of the two most common patterns or signemes from the sentence 'MY PASSPORT THERE STILL GOOD THERE'.

start positions vs. the estimated start positions of the pattern extracted from each of the sentences. Similarly, Figure 15(b) shows the corresponding scatter plot for the end position of the patterns in the sentences. As can be seen, the points are more scattered as compared to the results shown in Figure 10 where the sentences used were known to contain common words. However, this result is still encouraging. A large proportion



(a) Video Start Point Estimation
(b) Video End Point Estimation

Figure 15: Extraction of the most common patterns or signemes from the 'mixed' sentence groups. The closer the points are to the diagonal, the closer the result is to the ground truth.

of the extracted patterns are incorrect, but there are many relatively near the diagonal. This result demonstrates the robustness of our algorithm for finding similarities in the presence of great dissimilarity. We believe that the incorrect patterns extracted are due to the differences in the frame width ranges for the mixed sentence sets. For example, sentences containing the word 'MOVE' were combined with sentences containing the word 'HAVE'. The frame width range for the sign 'HAVE' is between 4 and 6 frames with 4 being the minimum width and 6 being the maximum width. On the other hand, the frame width range for the sign 'MOVE' is between 19 and 27 frames. Combining these width ranges could be done using an average of the two or by selecting the mini-

mum and maximum values between the two. However, these methods produced similar results. The correct combination of these range widths is a priority for future work.

4.3. Sign Localization

We used the extracted signemes to localize or spot signs in test sentences. The same process that is used for training sign models is used for sign localization. However, rather than randomly assigning initial parameter values, we use the parameters learned. We tested with 12 test sentences from the 'pure' group specified in Section 4.1 and their lengths varied from 4 to 12 signs. These test sentences were not used during training. The set of points representing the signeme were matched with the segments of the SoRD points from the test sentences to find the segment with the minimum matching score, which would represent the sign in the test sentence. The SoRD points of the signeme retrieved from the test sentence are mapped to their nearest frames and compared with the ground truth frame series representing the sign in the sentence. Localization performance is characterized as follows. Let a_1 and b_1 denote the start and

Test Group	Precision	Recall
Buy	1.0	0.70
Depart	1.0	0.64
Future	0.71	0.756
Move	1.0	0.60
Passport	1.0	0.47
Security	0.57	0.67
Ticket	1.0	0.58
Time	0.63	1.0

Table 2: Localization Performance

end frame numbers of the underlying ground truth sign in the test sentence, and a_2 and b_2 denote the start and end frame numbers of the subsequence retrieved as the signeme for the test sentence. We calculate the precision and recall values of each test sentence as $\frac{m}{a_2-a_1+1}$ and $\frac{m}{b_2-a_2+1}$ respectively where m is the number of overlapping frames. Table 2 displays the results acquired. The 'Baggage', 'Cant', 'Have', and 'Table' test sequences were failure cases where there was no overlap between the extracted model frames and the localization frames (see Figure 16). Notice that the localization results heavily depend on the extracted signeme models. For a visual representation of this information, we define the Start Offset, ΔS, and End Offset, ΔE, as $\Delta S = a_1 - a_2$ and $\Delta E = b_1 - b_2$. The plot of the Start Offset vs. the End Offset is shown in Figure 16. Ideally, both the offsets should be zero. The points for different signs are scattered in the four quadrants depending on the nature of the overlap between the ground truth sign and the retrieved signeme. Each point in the plot corresponds to a separate test sign. Its distance from the origin indicates the localizing quality of the signeme in its test sentence. The closer it is to the origin, the better the quality.

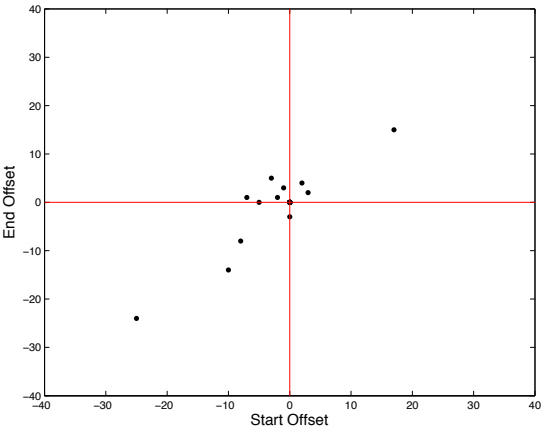


Figure 16: Start Offset vs. End Offset of Localized Signs

5. Conclusion And Future Work

We presented a novel algorithm to extract signemes, that is, the common pattern representing a sign, from multiple long video sequences of American Sign Language (ASL). A signeme is a part of the sign that is robust to the variations of the adjacent signs and the associated movement epenthesis. We first represent each sequence as a series of points in a low dimensional space of relational distributions, and then use a probabilistic framework to locate the signemes in each sequence concurrently. We use iterative conditional modes (ICM) to sample the parameters, that is, the starting location and width of the signemes in each sentence in a sequential manner. We show results on ASL video sequences that do not involve using any magnetic trackers or gloves for extracting the most common signs. The extracted signemes demonstrate that our approach is robust to some extent to the variations produced within a sign due to different contexts.

The approach in this paper can be used to speed up training set generation for ASL algorithms by drastically reducing the manual aspect of the process. Rather than manually demarcating signs in continuous sentences, which for our work took an expert approximately 5 minutes, we would just need instances of sentences containing the sign whose model is sought and based on our experiments this can be generated in approximately 2 minutes. Another contribution of this work is an empirically derived robust representation of the sign that is stable with respect to the variations due to neighboring signs and sentence context. These stable representations could be useful for detection of signs and gestures in extended gesture sequences.

There are some ways we can advance the work in this paper. One issue is the precision of the features used for representing the video sequences. Relational distributions when used as fixed size histograms perform well for discriminating global motion. However, optimizing the bin size of the histograms to the required precision might improve the accuracy. Additionally, we plan to extend our work to address the

challenge of handling the large variations encountered when automatically recognizing signemes across different signers. Also, the algorithm is dependent to a large extent on the distance measure and conventional dynamic time warping cannot deal with the amplitude variations in the signs, which are very common across signers. We plan to work on a variation of dynamic time warping that is robust to amplitude differences between various instances of signs.

Acknowledgments

This work was supported in part by funds from University of South Florida's College of Engineering Interdisciplinary Scholarship Program and the National Science Foundation under ITR grant IIS 0312993.

References

O. Al-Jarrah and A. Halawani. Recognition of gestures in Arabic Sign Language using neuro-fuzzy systems. *Artificial Intelligence*, 133:117–138, 2001.

V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. Boostmap: A method for efficient approximate similarity rankings. *IEEE Conference On Computer Vision And Pattern Recognition*, pages 268–275, 2004.

T Bailey and C. Elkan. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, 21:51–80, 1995.

B. Bauer and H. Hienz. Relevant features for video-based continuous sign language recognition. In *Automatic Face And Gesture Recognition*, pages 440–445, 2000.

B. Bauer and K. F. Kraiss. Video-based sign recognition using self-organizing subunits. In *International Conference On Pattern Recognition*, volume 2, pages 434–437, 2002.

J. Besag. On the statistical analysis of dirty pictures. *Journal Of The Royal Statistical Society*, pages 259–302, 1986.

R. Bowden, D. Windridge, T. Kadir, A. Zisserman, and M. Brady. A linguistic feature vector for the visual interpretation of sign language. In *European Conference On Computer Vision*, volume 1, pages 390–401, 2004.

P. Buehler, A. Zisserman, and M. Everingham. Learning sign language by watching tv (using weakly aligned subtitles). In *IEEE Conference On Computer Vision And Pattern Recognition*, pages 2961–2968, June 2009.

G. Casella and E.I. George. Explaining the Gibbs sampler. *The American Statistician*, 46:167–174, 1992.

S. Chib and E. Greenberg. Understanding the Metropolis–Hastings algorithm. *The American Statistician*, 49:327–335, 1995.

B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. *ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, pages 493–498, 2003.

H. Cooper and R. Bowden. Learning signs from subtitles: A weakly supervised approach to sign language recognition. In *IEEE Conference On Computer Vision And Pattern Recognition*, pages 2568–2574, June 2009.

Y. Cui and J. Weng. Appearance-based hand sign recognition from intensity image sequences. *Computer Vision And Image Understanding*, 78:157–176, 2000.

A. Denton. Kernel-density-based clustering of time series subsequences using a continuous random-walk noise model. *International Conference On Data Mining*, 2005.

K.G. Derpanis, R.R. Wildes, and J.K. Tsotsos. Hand gesture recognition within a linguistics-based framework. *European Conference On Computer Vision*, pages 282–296, 2004.

F. Duchene, C. Garbay, and V. Rialle. Learning recurrent behaviors from heterogeneous multivariate time-series. *Artificial Intelligence In Medicine*, 39(1):25–47, 2007.

G. Fang, X. Gao, W. Gao, and Y. Chen. A novel approach to automatically extracting basic units from Chinese Sign Language. *International Conference On Pattern Recognition*, 4:454–457, 2004.

A. Farhadi, D.A. Forsyth, and R. White. Transfer learning in sign language. In *Computer Vision And Pattern Recognition*, pages 1–8, 2007.

W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. *Markov chain Monte Carlo in practice*. Chapman and Hall, 1998.

J. Han, G. Awad, and A. Sutherland. Modelling and segmenting subunits for sign language recognition based on hand motion analysis. *Pattern Recognition Letters*, 30(6):623–633, 2009.

C.E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton. Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. *Science*, 262:208–214, 1993.

S.K. Liddell and R.E. Johnson. American Sign Language: The phonological base. *Sign Language Studies*, pages 195–277, 1989.

J. Ma, W. Gao, C. Wang, and J. Wu. A continuous Chinese Sign Language recognition system. *International Conference On Automatic Face And Gesture Recognition*, pages 428–433, 2000.

D. Minnen, C.L. Isbell, I. Essa, and T. Starner. Discovering multivariate motifs using subsequence density estimation and greedy mixture learning. *Conference On Artificial Intelligence*, 2007.

S. Nayak, S. Sarkar, and B. Loeding. Unsupervised modeling of signs embedded in continuous sentences. *IEEE Workshop On Vision For Human-Computer Interaction*, 2005.

S. Nayak, S. Sarkar, and B. Loeding. Automated extraction of signs from continuous sign language sentences using iterated conditional modes. In *IEEE Conference On Computer Vision And Pattern Recognition*, pages 2583–2590, June 2009a.

S. Nayak, S. Sarkar, and B. Loeding. Distribution-based dimensionality reduction applied to articulated motion recognition. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 31(5):795–810, May 2009b.

T. Oates. PERUSE: An unsupervised algorithm for finding recurring patterns in time series. *International Conference On Data Mining*, pages 330–337, 2002.

S.C.W. Ong and S. Ranganath. Automatic sign language analysis: A survey and the future beyond lexical meaning. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 27:873–891, June 2005.

M. Oszust and M. Wysocki. Determining subunits for sign language recognition by evolutionary cluster-based segmentation of time series. In *Artifical Intelligence And Soft Computing*, volume 6114 of *Lecture Notes In Computer Science*, pages 189–196. Springer Berlin / Heidelberg, 2010.

P. A. Pevzner and S. H. Sze. Combinatorial approaches to finding subtle signals in DNA sequences. *International Conference On Intelligent Systems For Molecular Biology*, pages 269–278, 2000.

S.L. Phung, A. Bouzerdoum, and D. Chai. Skin segmentation using color pixel classification: Analysis and comparison. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 27:148–154, January 2005.

I. Rigoutsos and A. Floratos. Combinatorial pattern discovery in biological sequences: The Teiresias algorithm. *Bioinformatics*, 14:55–67, 1998.

C.P. Robert and G. Casella. *Monte Carlo Statistical Methods*. New York: Springer-Verlag, 2004.

T. Starner and A. Pentland. Real-time American Sign Language recognition from video using Hidden Markov Models. *Computational Imaging And Vision*, 9:227–244, 1997.

T. Starner, J. Weaver, and A. Pentland. Real-time American Sign Language recognition using desk and wearable computer based video. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 20(12):1371–1375, 1998.

Yoshiki Tanaka, Kazuhisa Iwamoto, and Kuniaki Uehara. Discovery of time-series motif from multidimensional data based on MDL principle. *Machine Learning*, 58 (2-3):269–300, 2005.

I.R. Vega and S. Sarkar. Statistical motion model based on the change of feature relationships: Human gait-based recognition. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 25:1323–1328, October 2003.

C. Vogler and D. Metaxas. Parallel Hidden Markov Models for American Sign Language Recognition. *International Conference On Computer Vision*, 1:116–122, 1999.

C. Vogler and D. Metaxas. A framework of recognizing the simultaneous aspects of American Sign Language. *Computer Vision And Image Understanding*, 81:358–384, 2001.

C. Wang, W. Gao, and S. Shan. An approach based on phonemes to large vocabulary Chinese Sign Language recognition. *International Conference On Automatic Face And Gesture Recognition*, pages 393–398, 2002.

Q. Wang, X. Chen, L.G. Zhang, C. Wang, and W Gao. Viewpoint invariant sign language recognition. In *Computer Vision And Image Understanding*, volume 108, pages 87–97, 2007.

M.H. Yang, N. Ahuja, and M. Tabb. Extraction of 2d motion trajectories and its application to hand gesture recognition. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 24:1061–1074, 2002.

R. Yang, S. Sarkar, and B. Loeding. Handling movement epenthesis and hand segmentation ambiguities in continuous sign language recognition using nested dynamic programming. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 32(3):462–477, March 2010.

JMLR: Workshop and Conference Proceedings 14:1627–1663, 2013 Published 6/13

Dynamic Affine-Invariant Shape-Appearance Handshape Features and Classification in Sign Language Videos

Anastasios Roussos TROUSSOS@EECS.QMUL.AC.UK
Queen Mary, University of London
School of Electronic Engineering and Computer Science
Mile End Road, London E1 4NS, UK
Stavros Theodorakis STH@CS.NTUA.GR
Vassilis Pitsikalis VPITSIK@CS.NTUA.GR
Petros Maragos MARAGOS@CS.NTUA.GR
National Technical University of Athens
School of Electrical and Computer Engineering
Zografou Campus, Athens 15773, Greece

Editor: Isabelle Guyon and Vassilis Athitsos

Abstract

We propose the novel approach of dynamic affine-invariant shape-appearance model (Aff-SAM) and employ it for handshape classification and sign recognition in sign language (SL) videos. Aff-SAM offers a compact and descriptive representation of hand configurations as well as regularized model-fitting, assisting hand tracking and extracting handshape features. We construct SA images representing the hand's shape and appearance *without* landmark points. We model the variation of the images by linear combinations of eigenimages followed by affine transformations, accounting for 3D hand pose changes and improving model's compactness. We also incorporate static and dynamic handshape priors, offering robustness in occlusions, which occur often in signing. The approach includes an *affine signer adaptation* component at the visual level, without requiring training from scratch a new singer-specific model. We rather employ a short development data set to adapt the models for a new signer. Experiments on the Boston-University-400 continuous SL corpus demonstrate improvements on handshape classification when compared to other feature extraction approaches. Supplementary evaluations of sign recognition experiments, are conducted on a multi-signer, 100-sign data set, from the Greek sign language lemmas corpus. These explore the fusion with movement cues as well as signer adaptation of Aff-SAM to multiple signers providing promising results.

Keywords: affine-invariant shape-appearance model, landmarks-free shape representation, static and dynamic priors, feature extraction, handshape classification

1. Introduction

Sign languages (SL), that is, languages that convey information via visual patterns, commonly serve as an alternative or complementary mode of human communication. The visual patterns of SL are formed mainly by handshapes and manual motion, as well

as by non-manual patterns. The hand localization and tracking in a sign video as well as the derivation of features that reliably describe the configuration of the signer's hand are crucial for successful handshape classification. All the above are essential components for automatic sign language recognition systems or for gesture based human-computer interaction. Nevertheless, these tasks still pose several challenges, which are mainly due to the fast movement and the great variation of the hand's 3D shape and pose.

In this article, we propose a novel modeling of the shape and dynamics of the hands during signing that leads to efficient handshape features, employed to train statistical handshape models and finally for handshape classification and sign recognition. Based on 2D images acquired by a monocular camera, we employ a video processing approach that outputs reliable and accurate masks for the signer's hands and head. We construct *Shape-Appearance (SA) images* of the hand by combining 1) the hand's shape, as determined by its 2D hand mask, with 2) the hand's appearance, as determined by a normalized mapping of the colors inside the hand mask. The proposed modeling does not employ any landmark points and bypasses the point correspondence problem. In order to design a model of the variation of the SA images, which we call *Affine Shape-Appearance Model* (Aff-SAM), we modify the classic linear combination of eigenimages by incorporating *2D affine transformations*. These effectively account for various changes in the 3D hand pose and improve the model's compactness. After developing a procedure for the training of the Aff-SAM, we design a robust hand tracking system by adopting regularized model fitting that exploits prior information about the handshape and its dynamics. Furthermore, we propose to use as handshape features the Aff-SAM's eigenimage weights estimated by the fitting process.

The extracted features are fed into statistical classifiers based on Gaussian mixture models (GMM), via a supervised training scheme. The overall framework is evaluated and compared to other methods in extensive handshape classification experiments. The SL data are from the Boston University BU400 corpus (Neidle and Vogler, 2012). The experiments are based on manual annotation of handshapes that contain 3D pose parameters and the American Sign Language (ASL) handshape configuration. Next, we define classes that account for varying dependency of the handshapes w.r.t. the orientation parameters. The experimental evaluation addresses first, in a qualitative analysis the feature spaces via a cluster quality index. Second, we evaluate via supervised training a variety of classification tasks accounting for dependency w.r.t. orientation/pose parameters, with/without occlusions. In all cases we also provide comparisons with other baseline approaches or more competitive ones. The experiments demonstrate improved feature quality indices as well as classification accuracies when compared with other approaches. Improvements in classification accuracy for the non-occlusion cases are on average of 35% over baseline methods and 3% over more competitive ones. Improvements by taking into account the occlusion cases are on average of 9.7% over the more competitive methods.

In addition to the above, we explore the impact of Aff-SAM features in a sign recognition task based on statistical data-driven subunits and hidden Markov models. These experiments are applied on data from the Greek Sign Language (GSL) lemmas corpus

(DictaSign, 2012), for two different signers, providing a test-bed for the fusion with movement-position cues, and as evaluation of the affine-adapted SA model to a new signer, for which there has been no Aff-SAM training. These experiments show that the proposed approach can be practically applied to multiple signers without requiring training from scratch for the Aff-SAM models.

2. Background and Related Work

The first step of a hand gesture analysis system is the localization of the hands. This is usually implemented using several types of visual features, as skin color, edge information, shape and motion. Color cues are applicable because of the characteristic colors of the human skin. Many methods, including the one presented here, use skin color segmentation for hand detection (Argyros and Lourakis, 2004; Yang et al., 2002; Sherrah and Gong, 2000). Some degree of robustness to illumination changes can be achieved by selecting color spaces, as the *HSV*, *YCbCr* or the *CIE-Lab*, that separate the chromaticity from the luminance components (Terrillon et al., 2000; Kakumanu et al., 2007). In our approach, we adopt the *CIE-Lab* color space, due to its property of being perceptually uniform. Cui and Weng (2000) and Huang and Jeng (2001) employ motion cues assuming the hand is the only moving object on a stationary background, and that the signer is relatively still.

The next visual processing step is the hand tracking. This is usually based on blobs (Starner et al., 1998; Tanibata et al., 2002; Argyros and Lourakis, 2004), hand appearance (Huang and Jeng, 2001), or hand boundary (Chen et al., 2003; Cui and Weng, 2000). The frequent occlusions during signing make this problem quite challenging. In order to achieve robustness against occlusions and fast movements, Zieren et al. (2002), Sherrah and Gong (2000) and Buehler et al. (2009) apply probabilistic or heuristic reasoning for simultaneous assignment of labels to the possible hand/face regions. Our strategy for detecting and labeling the body-parts shares similarities with the above. Nevertheless, we have developed a more elaborate preprocessing of the skin mask, which is based on the mathematical morphology and helps us separate the masks of different body parts even in cases of overlaps.

Furthermore, a crucial issue to address in a SL recognition system is hand feature extraction, which is the focus of this paper. A commonly extracted positional feature is the 2D or 3D center-of-gravity of the hand blob (Starner et al., 1998; Bauer and Kraiss, 2001; Tanibata et al., 2002; Cui and Weng, 2000), as well as motion features (e.g., Yang et al., 2002; Chen et al., 2003). Several works use geometric measures related to the hand, such as shape moments (Hu, 1962; Starner et al., 1998) or sizes and distances between fingers, palm, and back of the hand (Bauer and Kraiss, 2001), though the latter employs color gloves. In other cases, the contour that surrounds the hand is used to extract translation, scale, and/or in-plane rotation invariant features, such as Fourier descriptors (Chen et al., 2003; Conseil et al., 2007).

Segmented hand images are usually normalized for size, in-plane orientation, and/or illumination and afterwards principal component analysis (PCA) is often applied for dimensionality reduction and descriptive representation of handshape (Sweeney and

Downton, 1996; Birk et al., 1997; Cui and Weng, 2000; Wu and Huang, 2000; Deng and Tsui, 2002; Dreuw et al., 2008; Du and Piater, 2010). Our model uses a similar framework but differs from these methods mainly in the following aspects. First, we employ a more general class of transforms to align the hand images, namely affine transforms that extend both similarity transforms, used, for example, by Birk et al. (1997) and translation-scale transforms as in the works of Cui and Weng (2000), Wu and Huang (2000) and Du and Piater (2010). In this way, we can effectively approximate a wider range of changes in the 3D hand pose. Second, the estimation of the optimum transforms is done simultaneously with the estimation of the PCA weights, instead of using a pipeline to make these two sets of estimations. Finally, unlike all the above methods, we incorporate combined static and dynamic priors, which make these estimations robust and allow us to adapt an existing model on a new signer.

Closely related to PCA approaches, active shape and active appearance models (Cootes and Taylor, 2004; Matthews and Baker, 2004) are employed for handshape feature extraction and recognition (Ahmad et al., 1997; Huang and Jeng, 2001; Bowden and Sarhadi, 2002; Fillbrandt et al., 2003). Our proposed shape-appearance model follows the same paradigm with these methods but differs: the modeled images are Shape-Appearance images and the image warps are not controlled by the shape landmarks but more simply by the 6 parameters of the affine transformation. In this way, it avoids shape representation through landmarks and the cumbersome manual annotation related to that.

Other more general purpose approaches have also been seen in the literature. A method earlier employed for action-type features is the histogram of oriented gradients (HOG): these descriptors are used for the handshapes of a signer (Buehler et al., 2009; Liwicki and Everingham, 2009; Ong et al., 2012). Farhadi et al. (2007) employ the scale invariant feature transform (SIFT) descriptors. Finally, Thangali et al. (2011) take advantage of linguistic constraints and exploit them via a Bayesian network to improve handshape recognition accuracy. Apart from the methods that process 2D hand images, there are methods built on a 3D hand model, in order to estimate the finger joint angles and the 3D hand pose (Athitsos and Sclaroff, 2002; Fillbrandt et al., 2003; Stenger et al., 2006; Ding and Martinez, 2009; Agris et al., 2008). These methods have the advantage that they can potentially achieve view-independent tracking and feature extraction; however, their model fitting process might be computationally slow.

Finally, regarding our related work, Roussos et al. (2010b) have included a short description of an initial tracking system similar to the one we adopt here. A preliminary version of the Aff-SAM method was presented by Roussos et al. (2010a). This is substantially extended here in many aspects, the main of which are the following: 1) We incorporate dynamic and static handshape priors offering robustness in cases of occlusions, 2) We develop an affine signer adaptation component, exploring the adaptation of Aff-SAM to multiple signers, 3) Extensive handshape classification experiments are presented, 4) Sign recognition experiments are conducted on a multi-signer database. In the sign recognition experiments of Section 8, we employ the handshape subunits construction presented by Roussos et al. (2010b). Finally, Theodorakis et al. (2012)

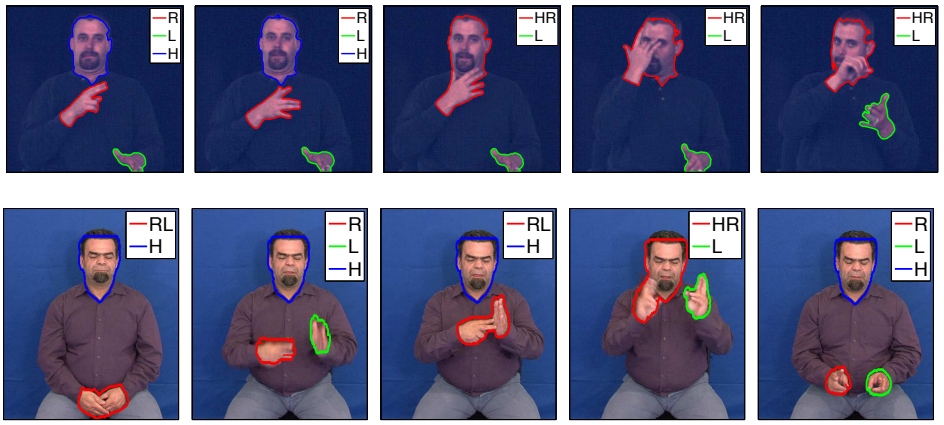


Figure 1: Output of the initial hands and head tracking in two videos of two different signers, from different databases. Example frames with extracted skin region masks and assigned body-part labels H (head), L (left hand), R (right hand).

and Theodorakis et al. (2011) present preliminary results on movement-handshape integration for continuous sign recognition.

3. Visual Front-End Preprocessing

The initial step of the visual processing is not the main focus of our method, nevertheless we describe it for completeness and reproducibility. The output of this subsystem at every frame is a set of skin region masks together with one or multiple *labels* assigned to every region, Figure 1. These labels correspond to the *body-parts of interest* for sign language recognition: head (H), left hand (L) and right hand (R). The case that a mask has multiple labels reflects an *overlap* of the 2D regions of the corresponding body-parts, that is, there is an *occlusion* of some body-parts. Referring for example to the right hand, there are the following cases: 1) The system outputs a mask that contains the right hand only, therefore there is *no occlusion* related to that hand, and 2) The output mask includes the right hand as well as other body-part region(s), therefore there is an *occlusion*. As presented in Section 4, the framework of SA refines this tracking while extracting handshape features.

3.1. Probabilistic Skin Color Modeling

We are based on the color cue for body-parts detection. We consider a Gaussian model of the signer's skin color in the perceptually uniform color space *CIE-Lab*, after keeping the two chromaticity components a^*, b^*, to obtain robustness to illumination (Cai and Goshtasby, 1999). We assume that the (a^*,b^*) values of skin pixels follow a bivariate Gaussian distribution $p_s(a^*,b^*)$, which is fitted using a training set of color samples

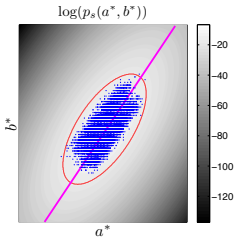


Figure 2: Skin color modeling. Training samples in the a^*-b^* space and fitted pdf $p_s(a^*, b^*)$. The ellipse bounds the colors that are classified to skin, according to the thresholding of $p_s(a^*(x), b^*(x))$. The straight line corresponds to the first PCA eigendirection on the skin samples and determines the projection that defines the mapping $g(I)$ used in the Shape-Appearance images formation.

(Figure 2). These samples are automatically extracted from pixels of the signer's face, detected using a face detector (Viola and Jones, 2003).

3.2. Morphological Processing of Skin Masks

In each frame, a first estimation of the skin mask S_0 is derived by thresholding at every pixel x the value $p_s(a^*(x), b^*(x))$ of the learned skin color distribution, see Figures 2, 3(b). The corresponding threshold is determined so that a percentage of the training skin color samples are classified to skin. This percentage is set to 99% to cope with training samples outliers. The skin mask S_0 may contain spurious regions or holes inside the head area due to parts with different color, as for instance eyes, mouth. For this, we regularize S_0 with tools from mathematical morphology (Soille, 2004; Maragos, 2005): First, we use the concept of *holes* $\mathscr{H}(S)$ of a binary image S, that is, the set of background components, not connected to the border of the image. In order to fill also some background regions that are not holes in the strict sense but are connected to the image border passing from a small "canal", we designed a filter that we call *generalized hole filling*. This filter yields a refined skin mask estimation $S_1 = S_0 \cup \mathscr{H}(S_0) \cup \{\mathscr{H}(S_0 \bullet B) \oplus B\}$ where B is a structuring element with size 5×5 pixels, and $\oplus$ and $\bullet$ denotes Minkowski dilation, closing respectively. The connected components (CCs) of relevant skin regions can be at most three (corresponding to the head and the two hands) and cannot have an area smaller than a threshold A_{min}, which corresponds to the smallest possible area of a hand region for the current signer and video acquisition conditions. Therefore, we apply an *area opening* with a varying threshold value: we find all CCs of S_1, compute their areas and finally discard all the components whose area is not on the top 3 or is less than A_{min}. This yields the final skin mask S_2, see Figure 3(c).

(a) Input (b) S_0 (c) S_2 (d) $S_2 \ominus B_c$ (e) Segmented S_2

Figure 3: Results of skin mask extraction and morphological segmentation. (a) Input. (b) Initial skin mask estimation S_0. (c) Final skin mask S_2 (morphological refinement). (d) Erosion $S_2 \ominus B_c$ of S_2 and separation of overlapped regions. (e) Segmentation of S_2 based on competitive reconstruction opening.

3.3. Morphological Segmentation of the Skin Masks

In the frames where S_2 contains three CCs, these yield an adequate segmentation. On the contrary, when S_2 contains less than three CCs, the skin regions of interest occlude each other. In such cases though, the occlusions are not always essential: different skin regions in S_2 may be connected via a thin connection, Figure 3(c). Therefore we further segment the skin masks of some frames by separating occluded skin regions with thin connections: If S_2 contains $N_{cc} < 3$ connected components, we find the CCs of $S_2 \ominus B_c$, Figure 3(d), for a structuring element B_c of small radius, for example, 3 pixels and discard those CCs whose area is smaller than A_{min}. A number of remaining CCs not greater than N_{cc} implies the absence of any thin connection, thus does not provide any occlusion separation. Otherwise, we use each one of these CCs as the seed of a different segment and expand it to cover S_2. For this we propose a *competitive reconstruction opening*, see Figure 3(e), described by the following iterative algorithm: In every iteration 1) each evolving segment expands using its conditional dilation by the 3×3 cross, relative to S_2, 2) pixels belonging to more than one segment are excluded from all segments. This means that segments are expanded inside S_2 but their expansion stops wherever they meet other segments. The above two steps are repeated until all segments remain unchanged.

3.4. Body-part Label Assignment

This algorithm yields 1) an assignment of one or multiple body-part labels, *head, left* and *right hand*, to all the segments and 2) an estimation of ellipses at segments with multiple labels (occluded). Note that these ellipses yield a rough estimate of the shapes of the occluded regions and contribute to the correct assignment of labels after each occlusion. A detailed presentation of this algorithm falls beyond the scope of this article. A brief description follows. *Non-occlusions*: For the hands' labels, given their values in the previous frames, we employ a prediction of the centroid position of each hand region taking into account three preceding frames and using a constant acceleration model. Then, we assign the labels based on minimum distances between the predicted positions and the segments' centroids. We also fit one ellipse on each segment since an ellipse can coarsely approximate the hand or head contour. *Occlusions*: Using the

parameters of the body-part ellipses already computed from the three preceding frames, we employ similarly forward prediction for all ellipses parameters, assuming constant acceleration. We face non-disambiguated cases by obtaining an auxiliary centroid estimation of each body-part via template matching of the corresponding image region between consecutive frames. Then, we repeat the estimations backwards in time. Forward and backward predictions, are fused yielding a final estimation of the ellipses' parameters for the signer's head and hands. Figure 1 depicts the output of the initial tracking in sequences of frames with non-occlusion and occlusion cases. We observe that the system yields accurate skin extraction and labels assignment.

4. Affine Shape-Appearance Modeling

In this section, we describe the proposed framework of dynamic affine-invariant shape-appearance model which offers a descriptive representation of the hand configurations as well as a simultaneous hand tracking and feature extraction process.

4.1. Representation by Shape-Appearance images

We aim to model all possible configurations of the dominant hand during signing, using directly the 2D hand images. These images exhibit a high diversity due to the variations on the configuration and 3D hand pose. Further, the set of the visible points of the hand is significantly varying. Therefore, it is more effective to represent the 2D handshape without using any landmarks. We thus represent the handshape by implicitly using its binary mask M, while incorporating also the *appearance* of the hand, that is, the color values inside this mask. These values depend on the hand texture and shading, and offer crucial 3D information.

If $I(x)$ is a cropped part of the current color frame around the hand mask M, then the hand is represented by the following *Shape-Appearance (SA) image* (see Figure 4):

$$f(x) = \begin{cases} g(I(x)), & \text{if } x \in M \\ -c_b, & \text{otherwise} \end{cases},$$

where $g : \mathbb{R}^3 \to \mathbb{R}$ maps the color values of the skin pixels to a color parameter that is appropriate for the hand appearance representation. This mapping is more descriptive for hand representation than a common color-to-gray transform. In addition, g is normalized so that the mapped values $g(I)$ of skin colors I have zero mean and unit variance. $c_b > 1$ is a background constant that controls the balance between shape and appearance. As c_b gets larger, the appearance variation gets relatively less weighted and more emphasis is given to the shape part. In the experiments, we have used $c_b = 3$ (that is three times the standard deviation of the foreground values $g(I)$).

The mapping $g(I)$ is constructed as follows. First we transform each color value I to the *CIE-Lab* color space, then keep only the chromaticity components a^*,b^*. Finally, we output the normalized weight of the first principal eigendirection of the PCA on the skin samples, that is the major axis of the Gaussian $p_s(a^*,b^*)$, see Section 3.1 and Figure 2(c). The output $g(I)$ is the most descriptive value for the skin pixels' chromaticity.

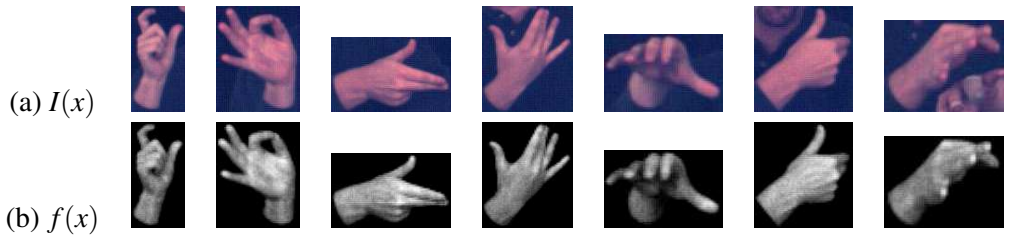


Figure 4: Construction of Shape-Appearance images. (a) Cropped hand images $I(x)$. (b) Corresponding Shape-Appearance images $f(x)$. For the foreground of $f(x)$ we use the most descriptive feature of the skin chromaticity. The background has been replaced by a constant value that is out of the range of the foreground values.

Furthermore, if considered together with the training of $p_s(a^*,b^*)$, the mapping $g(I)$ is invariant to global similarity transforms of the values (a^*,b^*). Therefore, the SA images are invariant not only to changes of the luminance component L but also to a wide set of global transforms of the chromaticity pair (a^*,b^*). As it will be described in Section 5, this facilitates the signer adaptation.

4.2. Modeling the Variation of Hand Shape-Appearance Images

Following Matthews and Baker (2004), the SA images of the hand, $f(x)$, are modeled by a linear combination of predefined variation images followed by an affine transformation:

$$f(W_p(x)) \approx A_0(x) + \sum_{i=1}^{N_c} \lambda_i A_i(x), x \in \Omega_M \,. \tag{1}$$

$A_0(x)$ is the mean image, $A_i(x)$ are N_c eigenimages that model the linear variation. These images can be considered as affine-transformation-free images. In addition, $\lambda = (\lambda_1 \cdots \lambda_{N_c})$ are the weights of the linear combination and W_p is an affine transformation with parameters $p = (p_1 \cdots p_6)$ that is defined as follows:

$$W_p(x,y) = \begin{pmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} .$$

The affine transformation models similarity transforms of the image as well as a significant range of changes in the 3D hand pose. It has a non-linear impact on the SA images and reduces the variation that is to be explained by the linear combination part, as compared to other appearance-based approaches that use linear models directly in the domain of the original images, (e.g., Cui and Weng, 2000). The linear combination of (1) models the changes in the configuration of the hand and the changes in the 3D orientation that cannot be modeled by the affine transform.

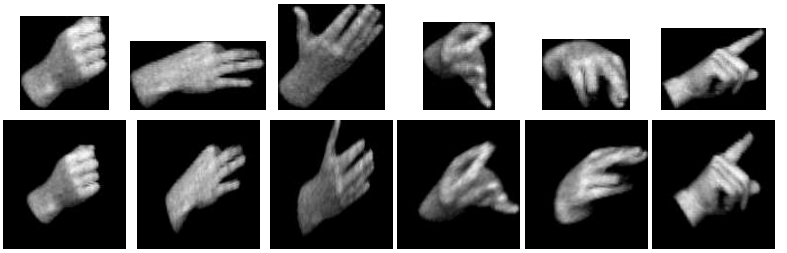


Figure 5: Semi-automatic affine alignment of a training set of Shape-Appearance images. (*Top row*) 6 out of 500 SA images of the training set. (*Bottom row*) Corresponding transformed images, after affine alignment of the training set. A video that demonstrates this affine alignment is available online (see text).

We will hereafter refer to the proposed model as *Shape-Appearance Model (SAM)*. A specific model of hand SA images is defined from the base image $A_0(x)$ and the eigenimages $A_i(x)$, which are statistically learned from training data. The vectors p and λ are the model parameters that fit the model to the hand SA image of every frame. These parameters are considered as features of hand pose and shape respectively.

4.3. Training of the SAM Linear Combination

In order to train the hand SA images model, we employ a representative set of hand-shape images from frames where the modeled hand is fully visible and non-occluded. Currently, this set is constructed by a random selection of approximately 500 such images. To exclude the variation that can be explained by the affine transformations of the model, we apply a semi-automatic affine alignment of the training SA images. For this, we use the framework of *procrustes analysis* (Cootes and Taylor, 2004; Dryden and Mardia, 1998), which is an iterative process that is repeatedly applying 1-1 alignments between pairs of training samples. In our case, the 1-1 alignments are affine alignments, implemented by applying the inverse-compositional (IC) algorithm (Gross et al., 2005) on pairs of SA images.

The IC algorithm result depends on the initialization of the affine warp, since the algorithm converges to a local optimum. Therefore, in each 1-1 alignment we test two different initializations: Using the binary masks M of foreground pixels of the two SA images, these initializations correspond to the two similarity transforms that make the two masks have the same centroid, area and orientation.[1] Among the two alignment results, the plausible one is kept, according to manual feedback from a user.

It must be stressed that the manual annotation of plausible alignment results is needed only during the training of the SA model, not during the fitting of the model. Also, compared to methods that use landmarks to model the shape (e.g., Cootes and Taylor, 2004; Matthews and Baker, 2004; Ahmad et al., 1997; Bowden and Sarhadi,

1. The existence of two such transforms is due to the modulo-π ambiguity of the orientation.

2002), the amount of manual annotation during training is substantially decreased: The user here is not required to annotate points but just make a binary decision by choosing the plausible result of 1-1 alignments. Other related methods for aligning sets of images are described by Learned-Miller (2005) and Peng et al. (2010). However, the adopted Procrustes analysis framework facilitates the incorporation of the manual annotation in the alignment procedure. Figure 5 shows some results from the affine alignment of the training set. For more details, please refer to the following URL that contains a video demonstration of the training set alignment: http://cvsp.cs.ntua.gr/research/sign/aff_SAM. We observe that the alignment produces satisfactory results, despite the large variability of the images of the training set. Note that the resolution of the aligned images is 127×133 pixels.

Then, the images A_i of the linear combination of the SA model are statistically learned using principal component analysis (PCA) on the aligned training SA images. The number N_c of eigenimages kept is a basic parameter of the SA model. Using a larger N_c, the model can better discriminate different hand configurations. On the other hand, if N_c gets too large, the model may not generalize well, in the sense that it will be consumed on explaining variation due to noise or indifferent information. In the setup of our experiments, we have practically concluded that the value $N_c = 35$ is quite effective. With this choice, the eigenimages kept explain 78% of the total variance of the aligned images.

Figure 6 demonstrates results of the application of PCA. Even though the modes of principal variation do not correspond to real handshapes, there is some intuition behind the influence of each eigenimage at the modeled hand SA image. For example, the first eigenimage A_1 has mainly to do with the foreground appearance: as its weight gets larger, the foreground intensities get darker and vice-versa. As another example, we see that by increasing the weight of the second eigenimage A_2, the thumb is extended. Note also that when we decrease the weight of A_4 all fingers extend and start detaching from each other.

4.4. Regularized SAM Fitting with Static and Dynamic Priors

After having built the shape-appearance model, we fit it in the frames of an input sign language video, in order to track the hand and extract handshape features. Precisely, we aim to find in every frame n the parameters $\lambda = \lambda[n]$ and $p = p[n]$ that generate a model-based synthesized image that is sufficiently "close" to the current input SA image $f(x)$. In parallel, to achieve robustness against occlusions, we exploit prior information about the handshape and its dynamics. Therefore, we minimize the following energy:

$$E(\lambda, p) = E_{rec}(\lambda, p) + w_S E_S(\lambda, p) + w_D E_D(\lambda, p) \,, \tag{2}$$

where E_{rec} is a reconstruction error term. The terms $E_S(\lambda, p)$ and $E_D(\lambda, p)$ correspond to static and dynamic priors on the SAM parameters λ and p. The values w_S,w_D are positive weights that control the balance between the 3 terms.

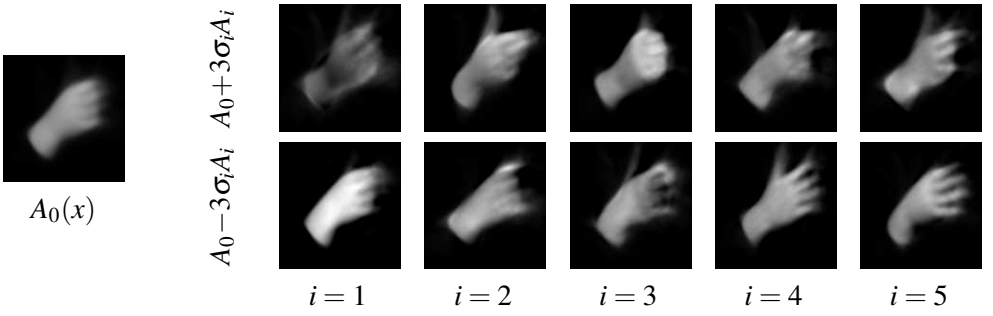


Figure 6: Result of the PCA-based learning of the linear variation images of Equation (1): Mean image $A_0(x)$ and principal modes of variation that demonstrate the first 5 eigenimages. The top (bottom) row corresponds to deviating from A_0 in the direction of the corresponding eigenimage, with a weight of $3\sigma_i$ (-$3\sigma_i$), where σ_i is the standard deviation of the corresponding component.

The reconstruction error term E_{rec} is a mean square difference defined by:

$$E_{rec}(\lambda, p) = \frac{1}{N_M} \sum_x \left\{ A_0(x) + \sum_{i=1}^{N_c} \lambda_i A_i(x) - f(W_p(x)) \right\}^2 ,$$

where the above summation is done over all the N_M pixels x of the domain of the images $A_i(x)$.

The static priors term $E_S(\lambda, p)$ ensures that the solution stays relatively close to the parameters mean values λ_0,p_0 :

$$E_S(\lambda, p) = \frac{1}{N_c} \|\lambda - \lambda_0\|^2_{\Sigma_\lambda} + \frac{1}{N_p} \|p - p_0\|^2_{\Sigma_p} ,$$

where N_c and N_p are the dimensions of λ and p respectively (since we model affine transforms, N_p=6). These numbers act as normalization constants, since they correspond to the expected values of the quadratic terms that they divide. Also, Σ_λ and Σ_p are the covariance matrices of λ and p respectively,[2] which are estimated during the training of the priors (Section 4.4.2). We denote by $\|y\|_A$, with A being a $N \times N$ symmetric positive-definite matrix and $y \in \mathbb{R}^N$, the following Mahalanobis distance from y to 0:

$$\|y\|_A \triangleq \sqrt{y^T A^{-1} y} .$$

Using such a distance, the term $E_S(\lambda, p)$ penalizes the deviation from the mean values but in a weighted way, according to the appropriate covariance matrices.

The dynamic priors term $E_D(\lambda, p)$ makes the solution stay close to the parameters estimations $\lambda^e = \lambda^e[n]$, $p^e = p^e[n]$ based on already fitted values on adjacent frames

2. We have assumed that the parameters λ and p are statistically independent.

(for how these estimations are derived, see Section 4.4.1):

$$E_D(\lambda, p) = \frac{1}{N_c} \|\lambda - \lambda^e\|^2_{\Sigma_{\varepsilon_\lambda}} + \frac{1}{N_p} \|p - p^e\|^2_{\Sigma_{\varepsilon_p}} \ , \tag{3}$$

where $\Sigma_{\varepsilon_\lambda}$ and Σ_{ε_p} are the covariance matrices of the estimation errors of λ and p respectively, see Section 4.4.2 for the training of these quantities too. The numbers N_c and N_p act again as normalization constants. Similarly to $E_S(\lambda, p)$, the term $E_D(\lambda, p)$ penalizes the deviation from the predicted values in a weighted way, by taking into account the corresponding covariance matrices. Since the parameters λ are the weights of the eigenimages $A_i(x)$ derived from PCA, we assume that their mean $\lambda_0 = 0$ and their covariance matrix Σ_λ is diagonal, which means that each component of λ is independent from all the rest.

It is worth mentioning that the energy-balancing weights w_S,w_D are not constant through time, but depend on whether the modeled hand in the current frame is occluded or not (this information is provided by the initial tracking preprocessing step of Section 3). In the occlusion cases, we are less confident than in the non-occlusion cases about the input SA image $f(x)$, which is involved in the term $E_{rec}(\lambda, p)$. Therefore, in these cases we obtain more robustness by increasing the weights w_S,w_D. In parallel, we decrease the relative weight of the dynamic priors term $\frac{w_D}{w_S+w_D}$, in order to prevent error accumulation that could be propagated in long occlusions via the predictions λ^e, p^e. After parameters tuning, we have concluded that the following choices are effective for the setting of our experiments: 1) w_S=0.07, w_D=0.07 for the non-occluded cases and 2) w_S=0.98, w_D=0.42 for the occluded cases.

An input video is split into much smaller temporal segments, so that the SAM fitting is *sequential* inside every segment as well *independent* from the fittings in all the rest segments: All the video segments of consecutive non-occluded and occluded frames are found and the middle frame of each segment is specified. For each non-occluded segment, we start from its middle frame and we get 1) a segment with forward direction by ending to the middle frame of the next occluded segment and 2) a segment with backward direction by ending after the middle frame of the previous occluded segment. With this splitting, we increase the confidence of the beginning of each sequential fitting, since in a non-occluded frame the fitting can be accurate even without dynamic priors. In the same time, we also get the most out of the dynamic priors, which are mainly useful in the occluded frames. Finally, this splitting strategy allows a high degree of parallelization.

4.4.1. DYNAMICAL MODELS FOR PARAMETER PREDICTION

In order to extract the parameter estimations λ^e, p^e that are used in the dynamic prior term E_D (3), we use linear prediction models (Rabiner and Schafer, 2007). At each frame n, a varying number $K = K(n)$ of already fitted frames is used for the parameter prediction. If the frame is far enough from the beginning of the current sequential fitting, K takes its maximum value, K_{max}. This maximum length of a prediction window is a parameter of our system (in our experiments, we used $K_{max} = 8$ frames). If on the

other hand, the frame is close to the beginning of the corresponding segment, then K varies from 0 to K_{max}, depending on the number of frames of the segment that have been already fitted.

If $K = 0$, we are at the starting frame of the sequential fitting, therefore no prediction from other available frames can be made. In this case, which is degenerate for the linear prediction, we consider that the estimations are derived from the prior means $\lambda^e = \lambda_0$, $p^e = p_0$ and also that $\Sigma_{\varepsilon_\lambda} = \Sigma_\varepsilon$, $\Sigma_{\varepsilon_p} = \Sigma_p$, which results to $E_D(\lambda, p) = E_S(\lambda, p)$. In all the rest cases, we apply the framework that is described next.

Given the prediction window value K, the parameters λ are predicted using the following autoregressive model:

$$\lambda^e[n] = \sum_{\nu=1}^{K} \mathrm{A}_\nu \lambda[n \mp \nu] ,$$

where the $-$ sign ($+$ sign) corresponds to the case of forward (backward) prediction. Also, A_ν are $N_c \times N_c$ weight matrices that are learned during training (see Section 4.4.2). Note that for every prediction direction and for every K, we use a different set of weight matrices A_ν that is derived from a separate training. This is done to optimize the prediction accuracy for the specific case of every prediction window. Since the components of λ are assumed independent to each other, it is reasonable to consider that all weight matrices A_ν are diagonal, which means that each component has an independent prediction model.

As far as the parameters p are concerned, they do not have zero mean and we cannot consider them as independent since, in contrast to λ, they are not derived from a PCA. Therefore, in order to apply the same framework as above, we consider the following re-parametrization:

$$\widetilde{p} = U_p^T (p - p_0) \Leftrightarrow p = p_0 + U_p \widetilde{p} ,$$

where the matrix U_p contains column-wise the eigenvectors of Σ_p. The new parameters $\widetilde{p}$ have zero mean and diagonal covariance matrix. Similarly to λ, the normalized parameters $\widetilde{p}$ are predicted using the following model:

$$\widetilde{p}^e[n] = \sum_{\nu=1}^{K} \mathrm{B}_\nu \widetilde{p}[n \mp \nu] ,$$

where B_ν are the corresponding weight matrices which again are all considered diagonal.

4.4.2. AUTOMATIC TRAINING OF THE STATIC AND DYNAMIC PRIORS

In order to apply the regularized SAM fitting, we first learn the priors on the parameters λ and p and their dynamics. This is done by training subsequences of frames where the modeled hand is not occluded. This training does not require any manual annotation. We first apply a random selection of such subsequences from videos of the same signer. Currently, the randomly selected subsequences used in the experiments are 120 containing totally 2882 non-occluded frames and coming from 3 videos. In all the training

subsequences, we fit the SAM in each frame independently by minimizing the energy in Equation (2) with w_S=w_D=0 (that is without prior terms). In this way, we extract fitted parameters λ, p for all the training frames. These are used to train the static and dynamic priors.

4.4.3. STATIC PRIORS

In this case, for both cases of λ and p, the extracted parameters from all the frames are used as samples of the same multivariate distribution, without any consideration of their successiveness in the training subsequences. In this way, we form the training sets T_λ and T_p that correspond to λ and p respectively. Concerning the parameter vector λ, we have assumed that its mean $\lambda_0 = 0$ and its covariance matrix Σ_λ is diagonal. Therefore, only the diagonal elements of Σ_λ, that is the variances $\sigma^2_{\lambda_i}$ of the components of λ, are to be specified. This could be done using the result of the PCA (Section 4.2), but we employ the training parameters of T_λ that come from the direct SAM fitting, since they are derived from a process that is closer to the regularized SAM fitting. Therefore, we estimate each $\sigma^2_{\lambda_i}$ from the empirical variance of the corresponding component λ_i in the training set T_λ. Concerning the parameters p, we estimate p_0 and Σ_p from the empirical mean and covariance matrix of the training set T_p.

4.4.4. DYNAMIC PRIORS

As already mentioned, for each prediction direction (forward, backward) and for each length K of the prediction window, we consider a different prediction model. The $(K+1)$-plets[3] of samples for each one of these models are derived by sliding the appropriate window in the training sequences. In order to have as good accuracy as possible, we do not make any zero (or other) padding in unknown parameter values. Therefore, the samples are picked only when the window fits entirely inside the training sequence. Similarly to linear predictive analysis (Rabiner and Schafer, 2007) and other tracking methods that use dynamics (e.g., Blake and Isard, 1998) we learn the weight matrices A_v, B_v by minimizing the mean square estimation error over all the prediction-testing frames. Since we have assumed that A_v and B_v are diagonal, this optimization is done independently for each component of λ and $\widetilde{p}$, which is treated as 1D signal. The predictive weights for each component are thus derived from the solution of an ordinary least squares problem. The optimum values of the mean squared errors yield the diagonal elements of the prediction errors' covariance matrices $\Sigma_{\varepsilon_\lambda}$ and $\Sigma_{\varepsilon_{\widetilde{p}}}$, which are diagonal.

4.4.5. IMPLEMENTATION AND RESULTS OF SAM FITTING

The energy $E(\lambda, p)$ (2) of the proposed regularized SAM fitting is a special case of the general objective function that is minimized by the *simultaneous inverse compositional with a prior* (SICP) algorithm of Baker et al. (2004). Therefore, in order to minimize $E(\lambda, p)$, we specialize this algorithm for the specific types of our prior terms. Details

3. The $(K+1)$-plets follow from the fact that we need K neighbouring samples + the current sample.

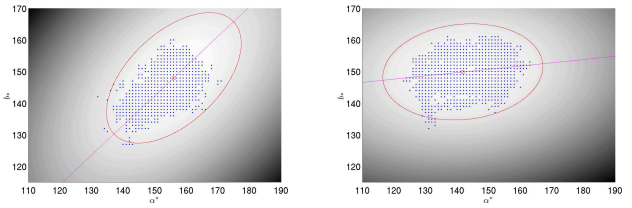
Figure 7: Regularized Shape-Appearance Model fitting in a sign language video. In every input frame, we superimpose the model-based reconstruction of the hand in the frame domain, $A_0(W_p^{-1}(x)) + \sum \lambda_i A_i(W_p^{-1}(x))$. In the upper-right corner, we display the reconstruction in the model domain, $A_0(x) + \sum \lambda_i A_i(x)$, which determines the optimum weights λ. A demo video is available online (see text).

are given in the Appendix A. At each frame n of a video segment, the fitting algorithm is initialized as follows. If the current frame is not the starting frame of the sequential fitting (that is $K(n) \neq 0$), then the parameters λ, p are initialized from the predictions λ^e, p^e. Otherwise, if $K(n) = 0$, we test as initializations the two similarity transforms that, when applied to the SAM mean image A_0, make its mask have the same centroid, area and orientation as the mask of the current frame's SA image. We twice apply the SICP algorithm using these two initializations, and finally choose the initialization that yields the smallest regularized energy $E(\lambda, p)$.

Figure 7 demonstrates indicative results of the regularized fitting of the dominant hand's SAM in a sign language video. For more details, please refer to the following URL that contains a video of these results: http://cvsp.cs.ntua.gr/research/sign/aff_SAM. We observe that in non-occlusion cases, this regularized method is effective and accurately tracks the handshape. Further, in occlusion cases, even after a lot of occluded frames, the result is especially robust. Nevertheless, the accuracy of the extracted handshape is smaller in cases of occlusions, compared to the non-occlusion cases, since the prior terms keep the result closer to the SAM mean image A_0. In addition, extensive handshape classification experiments were performed in order to evaluate the extracted handshape features employing the proposed Aff-SAM method (see Section 7).

5. Signer Adaptation

We develop a method for adapting a trained Aff-SAM model to a new signer. This adaptation is facilitated by the characteristics of the Aff-SAM framework. Let us consider an Aff-SAM model trained to a signer, using the procedure described in Section 4.3.



Signer A Signer B

Figure 8: Skin color modeling for the two signers of the GSL lemmas corpus, where we test the signer adaptation. Training samples in the a^*-b^* chromaticity space and fitted pdf's $p_s(a^*, b^*)$. In each case, the straight line defines the normalized mapping $g(I)$ used in the Shape-Appearance images formation.

We aim to reliably adapt and fit the existing Aff-SAM model on videos from a *new signer*.

5.1. Skin Color and Normalization

The employed skin color modeling adapts on the characteristics of the skin color of a new signer. Figure 8 illustrates the skin color modeling for the two signers of the GSL lemmas corpus, where we test the adaptation. For each new signer, the color model is built from skin samples of a face tracker (Section 3.1, Section 4.1). Even though there is an intersection, the domain of colors classified as skin is different between the two. In addition, the mapping $g(I)$ of skin color values, used to create the SA images, is normalized according to the skin color distribution of each signer. The differences in the lines of projection reveal that the normalized mapping $g(I)$ is different in these two cases. This skin color adaptation makes the body-parts label extraction of the visual front-end preprocessing to behave robustly over different signers. In addition, the extracted SA images have the same range of values and are directly comparable across signers.

5.2. Hand Shape and Affine Transforms

Affine transforms can reliably compensate for the anatomical differences of the hands of different signers. Figure 9 demonstrates some examples. In each case, the right hands of the signers are in a similar configuration and viewpoint. We observe that there exist pairs of affine transformations that successfully align the handshapes of both signers to the common model domain. For instance, the affine transforms have the ability to stretch or shrink the hand images over the major hand axis. They thus automatically compensate for the fact that the second signer has thinner hands and longer fingers. In general, the class of affine transforms can effectively approximate the transformation needed to align the 2D hand shapes of different signers.

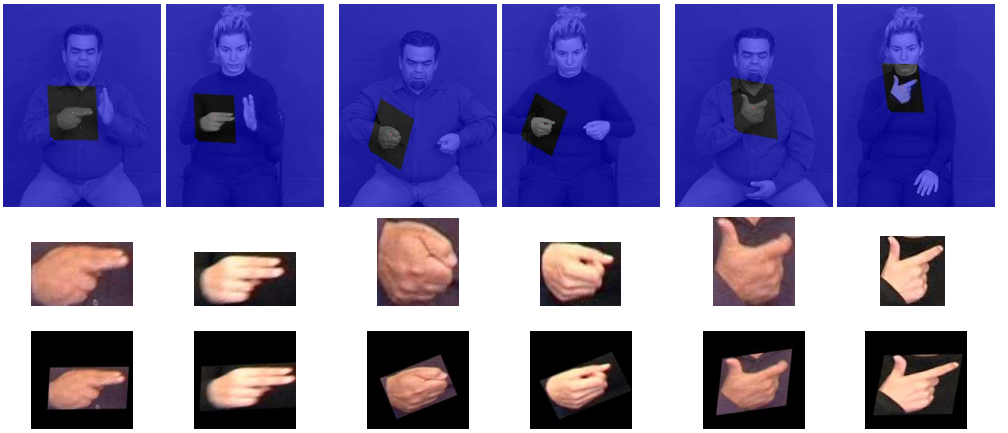
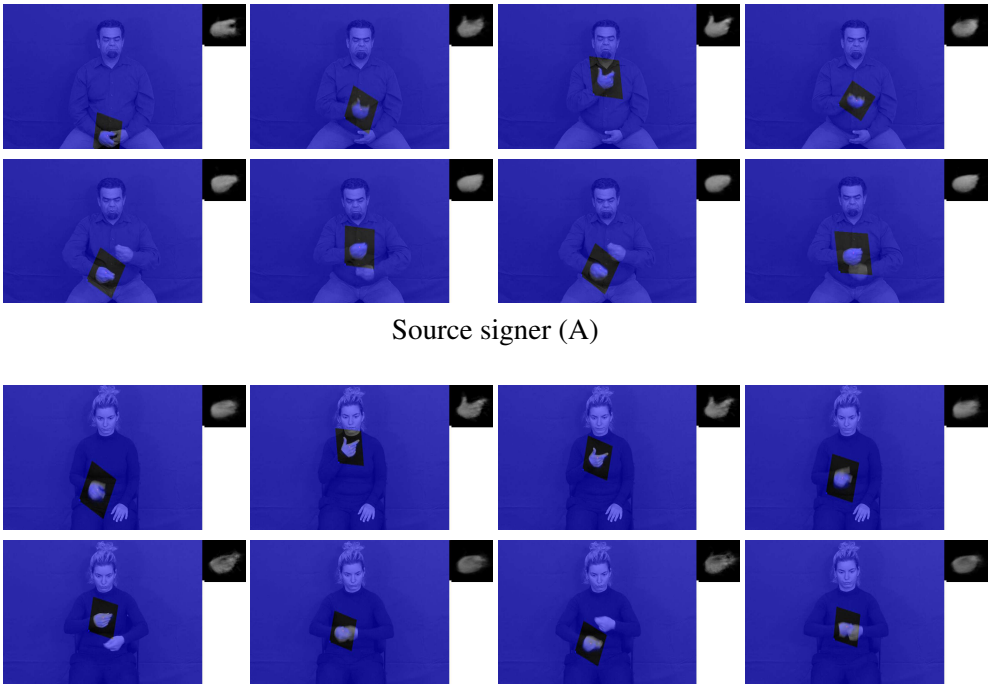


Figure 9: Alignment of the hands of two different signers, using affine transformations. *First row*: Input frames with superimposed rectangles that visualize the affine transformations. *Second row*: Cropped images around the hand. *Third row*: Alignment of the cropped images in a common model domain, using the affine transformations.

5.3. New Signer Fitting

To process a new signer the visual front-end is applied as in Section 3. Then, we only need to re-train the static and dynamic priors on the new signer. For this, we randomly select frames where the hand is not occluded. Then, for the purposes of this training, the existing SAM is fitted on them by minimizing the energy in Equation (2) with w_S=w_D=0, namely the reconstruction error term without prior terms. Since the SAM is trained on another signer, this fitting is not always successful, at this step. At that point, the user annotates the frames where this fitting has succeeded. This feedback is binary and is only needed during training and for a relatively small number of frames. For example, in the case of the GSL lemmas corpus, we sampled frames from approximately 1.2% of all corpus videos of this signer. In 15% of the sampled frames, this fitting with no priors was annotated as successful. Using the samples from these frames, we learn the static and dynamic priors of λ and p, as described in Section 4.4.2 for the new signer. The regularized SAM fitting is implemented as in Section 4.4.5.

Figure 10 demonstrates results of the SAM fitting, in the case of signer adaptation. The SAM eigenimages are learned using solely Signer A. The SAM is then fitted on the signer B, as above. For comparison, we also visualize the result of the SAM fitting to the signer A, for the same sign. Demo videos for these fittings also are included in the following URL: http://cvsp.cs.ntua.gr/research/sign/aff_SAM. We observe that, despite the anatomical differences of the two signers, the performance of the SAM fitting is satisfactory after the adaptation. In both signers, the fitting yields accurate shape estimation in non-occlusion cases.



Source signer (A)

New signer (B)

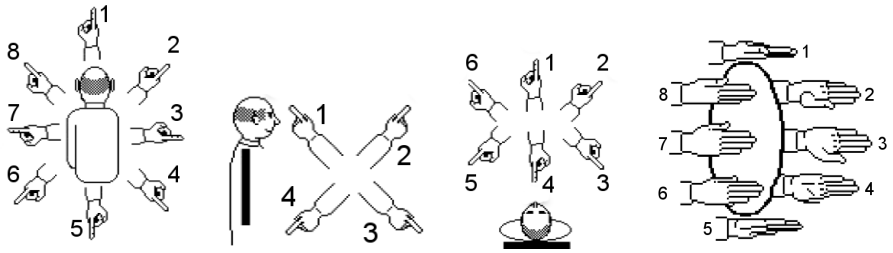
Figure 10: Regularized Shape-Appearance Model fitting on 2 signers. The SA model was trained on Signer A and adapted for Signer B. Demo videos are available online (see text).

6. Data Set and Handshape Annotation for Handshape Classification

The *SL Corpus BU400* (Neidle and Vogler, 2012) is a continuous American sign language database. The background is uniform and the images have a resolution of 648x484 pixels, recorded at 60 frames per second. In the classification experiments we employ the front camera video, data from a single signer, and the story 'Accident'. We next describe the annotation parameters required to produce the ground-truth labels. These concern the pose and handshape configurations and are essential for the supervised classification experiments.

6.1. Handshape Parameters and Annotation

The parameters that need to be specified for the annotation of the data are the (pose-independent) handshape configuration and the 3D hand pose, that is the orientation of the hand in the 3D space. For the annotation of the handshape configurations we followed the *SignStream annotation conventions* (Neidle, 2007). For the 3D hand pose we parametrized the 3D hand orientations inspired by the HamNoSys description (Hanke, 2004). The adopted annotation parameters are as follows: 1) *Handshape identity (HSId)*



(*a*) Front (F) (*b*) Side (S) (*c*) Bird's (B) (*d*) Palm (P)

Figure 11: 3D Hand Orientation parameters: (*a-c*) Extended Finger Direction Parameters: (*a*) Signer's front view (F), (*b*) Side view (S), (*c*) Birds' view (B); (*d*) Palm orientation (P). Note that we have modified the corresponding figures of Hanke (2004) with numerical parameters.

which defines the handshape configuration, that is, ('*A*', '*B*', '*1*', '*C*' etc.), see Table 1 for examples. 2) *3D Hand Orientation* (hand pose) consisting of the following parameters (see Figure 11): i) *Extended Finger Direction* parameters that define the orientation of the hand axis. These correspond to the hand orientation relatively to the three planes that are defined relatively to: the Signer's Front view (referred to as F), the Bird's view (B) and the Side view (S). ii) *Palm Orientation* parameter (referred to as P) for a given extended finger direction. This parameter is defined w.r.t. the bird's view, as shown in Figure 11(d).

6.2. Data Selection and Classes

We select and annotate a set of occluded and non-occluded handshapes so that 1) they cover substantial handshape and pose variation as they are observed in the data and 2) they are quite frequent. More specifically we have employed three different data sets (DS): 1) *DS-1*: 1430 non-occluded handshape instances with 18 different HSIds. 2) *DS-1-extend*: 3000 non-occluded handshape instances with 24 different HSIds. 3) *DS-2*: 4962 occluded and non-occluded handshape instances with 42 different HSIds. Table 1 presents an indicative list of annotated handshape configurations and 3D hand orientation parameters.

7. Handshape Classification Experiments

In this section we present the experimental framework consisting of the statistical system for handshape classification. This is based 1) on the handshape features extracted as described in Section 4; 2) on the annotations as described in Section 6.1 as well as 3) on the data selection and classes (Section 6.2). Next, we describe the experimental protocol containing the main experimental variations of the data sets, of the class dependency, and of the feature extraction method.

HSId		*1*	*1*	*4*	*4*	*5C*	*5*	*5*	*5*	*A*	*A*	*BL*	*BL*	*BL*	*BL*
3D hand pose	F	8	1	7	6	1	7	8	1	8	8	8	7	8	8
	S	0	0	0	3	1	0	2	2	0	2	0	0	0	0
	B	0	0	0	6	4	0	1	1	0	6	0	0	0	0
	P	1	8	3	1	3	3	1	5	3	2	2	3	3	4
# insts.		14	24	10	12	27	38	14	19	14	31	10	15	23	30
exmpls.															

HSId		*BLCUL*	*F*	*F*	*U*	*ULV*	*Y*	*b1*	*c5*	*c5*	*cS*	*cSfO2*			
3D hand pose	F	8	7	7	1	7	7	8	8	7	8	8	7	8	8
	S	2	0	0	2	0	0	0	0	0	0	0	0	2	0
	B	6	0	0	1	0	0	0	0	0	0	6	6	6	0
	P	4	3	3	3	2	3	2	2	3	3	1	3	3	1
# insts.		20	13	23	13	10	60	16	16	10	17	18	10	34	12
exmpls.															

Table 1: Samples of annotated handshape identities (HSId) and corresponding 3D hand orientation (pose) parameters for the D-HFSBP class dependency and the corresponding experiment; in this case each model is fully dependent on all of the orientation parameters. '# insts.' corresponds to the number of instances in the dataset. In each case, we show an example handshape image that is randomly selected among the corresponding handshape instances of the same class.

7.1. Experimental Protocol and Other Approaches

The experiments are conducted by employing cross-validation by selecting five different random partitions of the dataset into train-test sets. We employ 60% of the data for training and 40% for testing. This partitioning samples data, among all realizations per handshape class in order to equalize class occurrence. The number of realizations per handshape class are on average 50, with a minimum and maximum number of realizations in the range of 10 to 300 depending on the experiment and the handshape class definition. We assign to each experiment's training set one GMM per handshape class; each has one mixture and diagonal covariance matrix. The GMMs are uniformly initialized and are afterwards trained employing Baum-Welch re-estimation (Young et al., 1999). Note that we are not employing other classifiers since we are interested in the evaluation of the handshape features and not the classifier. Moreover this framework fits with common hidden Markov model (HMM)-based SL recognition frameworks (Vogler and Metaxas, 1999), as in Section 8.

Class Dependency label	Annotation Parameters HSId(H)	Front(F)	Side(S)	Bird's(B)	Palm(P)
D-HFSBP	D	D	D	D	D
D-HSBP	D	*	D	D	D
D-HBP	D	*	*	D	D
D-HP	D	*	*	*	D
D-H	D	*	*	*	*

Table 2: Class dependency on orientation parameters. One row for each model dependency w.r.t. the annotation parameters. The dependency or non-dependency state to a particular parameter for the handshape trained models is noted as 'D' or '*' respectively. For instance the D-HBP model is dependent on the HSId and Bird's view and Palm orientation parameters.

7.1.1. EXPERIMENTAL PARAMETERS

The experiments are characterized by the dataset employed, the class dependency and the feature extraction method as follows:

Data Set (DS): We have experimented employing three different data sets DS-1, DS-1-extend and DS-2 (Section 6.2 for details).

Class dependency (CD): The class dependency defines the orientation parameters in which our trained models are dependent to (Table 2). Take for instance the orientation parameter 'Front' (F). There are two choices, either 1) construct handshape models independent to this parameter or 2) construct different handshape models for each value of the parameter. In other words, at one extent CD restricts the models generalization by making each handshape model specific to the annotation parameters, thus highly discriminable, see for instance in Table 2 the experiment corresponding to D-HFSBP. At the other extent CD extends the handshape models generalization w.r.t. to the annotation parameters, by letting the handshape models account for pose variability (that is depend only on the HSId; same HSId's with different pose parameters are tied), see for instance experiment corresponding to the case D-H (Table 2). The CD field takes the values shown in Table 2.

7.1.2. FEATURE EXTRACTION METHOD

Apart from the proposed Aff-SAM method, the methods employed for handshape feature extraction are the following:

Direct Similarity Shape-Appearance Modeling (DS-SAM): Main differences of this method with Aff-SAM are as follows: *1)* we replace the affine transformations that are incorporated in the SA model (1) by simpler *similarity* transforms and *2)* we replace the regularized model fitting by direct estimation (*without* optimization) of the similarity transform parameters using the centroid, area and major axis orientation of the hand region followed by projection into the PCA subspace to find the eigenimage weights. Note that in the occlusion cases, this simplified fitting is done directly on the SA image

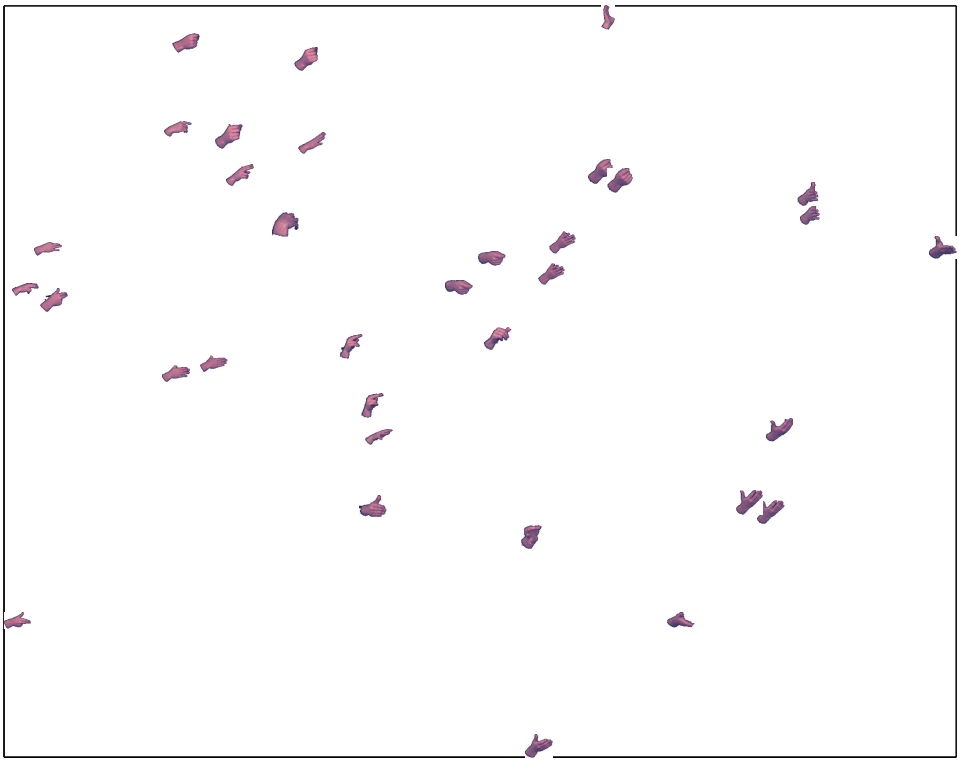
of the region that contains the modeled hand as well as the other occluded body-part(s) (that is the other hand and/or the head), without using any static or dynamic priors as those of Section 4.4. This approach is similar to Birk et al. (1997) and is adapted to fit our framework.

Direct Translation Scale Shape-Appearance Modeling (DTS-SAM): The main differences of this method with Aff-SAM are the following: *1)* we replace the affine transformations that are incorporated in the Shape-Appearance model (1) by simpler *translation-scale* transforms and *2)* we replace the regularized model fitting by direct estimation of the translation and scale parameters using the square that tightly surrounds the hand mask, followed again by projection into the PCA subspace to find the eigenimage weights. In this simplified version too, the hand occlusion cases are treated by simply fitting the model to the Shape-Appearance image that contains the occlusion, without static or dynamic priors. This approach is similar to Cui and Weng (2000), Wu and Huang (2000) and Du and Piater (2010) and is adapted so as to fit our proposed framework.

Other tested methods from the literature contain the *Fourier Descriptors* (FD): These are derived from the Fourier coefficients of the contour that surrounds the hand, after appropriate normalizations for scale and rotation invariance (Chen et al., 2003; Conseil et al., 2007). For dimensionality reduction, we keep the descriptors that correspond to the first N_{FD} frequencies. We tested different values for the parameter N_{FD} and finally kept $N_{FD} = 50$ that yield the best performance. *Moments* (M): These consist of the seven Hu moment invariants of the hand region (Hu, 1962). These depend only on the central moment of the binary shape of the hand region and are invariant to similarity transforms of the hand region. *Region Based* (RB): These consist of the area, eccentricity, compactness and minor and major axis lengths of the hand region (Agris et al., 2008). Compared to the proposed Aff-SAM features we consider the rest five sets of features belonging to either *baseline features* or *more advanced features*. First, the baseline features contain the FD, M and RB approaches. Second, the more advanced features contain the DS-SAM and DTS-SAM methods which we have implemented as simplified versions of the proposed Aff-SAM. As it will be revealed by the evaluations, the more advanced features are more competitive than the baseline features and the comparisons with them are more challenging.

7.2. Feature Space Evaluation Results

Herein we evaluate the feature space of the Aff-SAM method. In order to approximately visualize it, we employ the weights λ_1, λ_2 of the two principal eigenimages of Aff-SAM. Figure 12(*a*) provides a visualization of the trained models per class, for the experiment corresponding to D-HFSBP class dependency (that is each class is fully dependent on orientation parameters). It presents a single indicative cropped handshape image per class to add intuition on the presentation: these images correspond to the points in the feature space that are closest to the specific classes' centroids. We observe that similar handshape models share close positions in the space. The presented feature space is indicative and it seems clear when compared to feature spaces of other methods.



(*a*)

Figure 12: Feature space for the Aff-SAM features and the D-HFSBP experiment case (see text). The trained models are visualized via projections on the $\lambda_1 - \lambda_2$ plane that is formed from the weights of the two principal Aff-SAM eigen-images. Cropped handshape images are placed at the models' centroids.

To support this we compare the feature spaces with the Davies-Boulding index (DBi), which quantifies their quality. In brief, the DBi is the average over all n clusters, of the ratio of intra-cluster distances σ_i versus the inter-cluster distance $d_{i,j}$ of i, j clusters, as a measure of their separation: $DBi = \frac{1}{n}\sum_{i=1}^{n} \max_{i \neq j}(\frac{\sigma_i + \sigma_j}{d_{i,j}})$ (Davies and Bouldin, 1979). Figure 13 presents the results. The reported indices are for varying CD field, that is the orientation parameters on which the handshape models are dependent or not (as discussed in Section 7.1) and are referred in Table 2. We observe that the DBi's for the Aff-SAM features are lower that is the classes are more compact and more separable, compared to the other cases. The closest DBi's are these of DS-SAM. In addition, the proposed features show stable performance over experiments w.r.t. class-dependency, indicating robustness to some amount of pose variation.

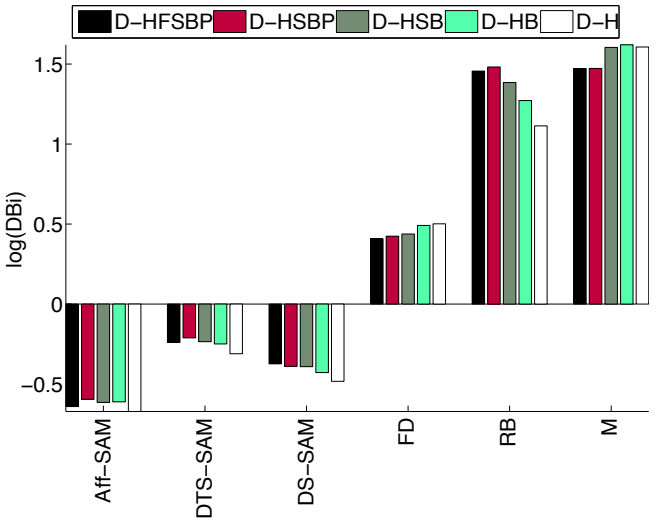


Figure 13: Davies-Bouldin index (DBi) in logarithmic scale (y-axis) for multiple feature spaces and varying models class dependency to the orientation parameters. Lower values of DBi indicate better compactness and separability of classes.

Data Set	# HSIds	CD	Occ.	Feat. Method	Avg.Acc.%	Std.
DS-1	18	Table. 2	✗	Aff-SAM	**93.7**	1.5
				DS-SAM	93.4	1.6
				DTS-SAM	89.2	1.9
DS-1-extend	24	'D-H'	✗	Aff-SAM	**77.2**	1.6
				DS-SAM	74	2.3
				DTS-SAM	67	1.4
DS-2	42	Table. 2	✓	Aff-SAM	**74.9**	0.9
				DS-SAM	66.1	1.1
				DTS-SAM	62.7	1.4

Table 3: Experiments overview with selected average overall results over different main feature extraction methods and experimental cases of DS and CD experiments, with occlusion or not (see Section 7.1). CD: class dependency. Occ.: indicates whether the dataset includes occlusion cases. # HSIds: the number of HSId employed, Avg.Acc.: average classification accuracy, Std.: standard deviation of the classification accuracy.

7.3. Results of Classification Experiments

We next show average classification accuracy results after 5-fold cross-validation for each experiment. together with the standard deviation of the accuracies. The experi-

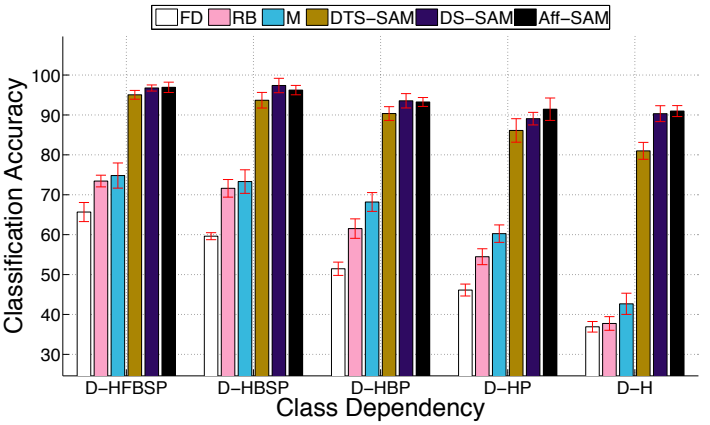


Figure 14: Classification experiments for non-occlusion cases, dataset DS-1. Classification Accuracy for varying experiments (x-axis) that is the dependency of each class w.r.t. the annotation parameters [H,F,B,S,P] and the feature employed (legend). For the numbers of classes per experiment see Table 4.

ments consist of 1) Class dependency and Feature variation for non-occlusion cases and 2) Class dependency and Feature variation for both occlusion and non-occlusion cases. Table 3 presents averages as well as comparisons with other features for the three main experimental data sets discussed. The averages are over all cross-validation cases, and over the multiple experiments w.r.t. class dependency, where applicable. For instance, in the first block for the case 'DS-1', that is non-occluded data from the dataset DS-1, the average is taken over all cases of class dependency experiments as described in Table 2. For the 'DS-1-extend' case, the average is taken over the D-H class dependency experiment, since we want to increase the variability within each class.

7.3.1. FEATURE COMPARISONS FOR NON-OCCLUDED CASES

Next, follow comparisons by employing the referred feature extraction approaches, for two cases of data sets, while accounting for non-occluded cases.

7.3.2. DATA SET DS-1

In Figure 14 we compare the employed methods, while varying the models' dependency w.r.t. the annotation parameters (x axis). We employ the DS-1 data set, consisting of 18 handshape types from non-occlusion cases. The number of classes are shown in Table 4. In Figure 14 we depict the performance over the different methods and models' dependency. At the one extent (that is 'D-HFBSP') we trained one GMM model for each different combination of the handshape configuration parameters (H,F,B,S,P). Thus, the trained models were dependent on the 3D handshape pose and so are the classes for the classification (34 different classes). In the other extent ('D-H') we trained one GMM model for each HSId thus the trained models were independent to

Class dependency Parameters	D-HFSBP	D-HSBP	D-HBP	D-HP	D-H
# Classes	34	33	33	31	18

Table 4: Number of classes for each type of class dependency (classification experiments for Non-Occlusion cases).

the 3D handshape pose and so are the classes for the classification (18 different classes). Furthermore we observe that the proposed method outperforms the baseline methods (FD, RB, M) and DTS-SAM. However the classification performance of Aff-SAM and DS-SAM methods is quite close in some cases. This is due to the easy classification task (small number of HSIds and 3D pose variability and non-occlusion cases). The classification performance of the proposed method is slightly affected from the decrease of the dependency on the annotation parameters. This strengthens our previous observation that the proposed method can handle small pose variations. For a results' overview see Table 3 (DS-1 block). The averages are across all pose-dependency cases.

7.3.3. DATA SET DS-1-EXTEND

This is an extension of DS-1 and consists of 24 different HSIds with much more 3D handshape pose variability. We trained models independent to the 3D handshape pose. Thus, these experiments refer to the D-H case. Table 3 (DS-1-extend block) shows average results for the three competitive methods. We observe that Aff-SAM outperforms both DS-SAM and DTS-SAM achieving average improvements of 3.2% and 10.2% respectively. This indicates the advancement of the Aff-SAM over the other two competitive methods (DS-SAM and DTS-SAM) in more difficult tasks. It also shows that, by incorporating more data with extended variability w.r.t. pose parameters, there is an increase in the average improvements.

Class dependency Parameters	D-HFSBP	D-HSBP	D-HBP	D-HP	D-H
# Classes	100	88	83	72	42

Table 5: Number of classes for each type of class dependency (classification experiments for Occlusion and Non-Occlusion cases).

7.3.4. FEATURE COMPARISONS FOR OCCLUDED AND NON-OCCLUDED CASES

In Figure 15 we vary the models' dependency w.r.t. the annotation parameters similar to Section 7.3.1. However, DS-2 data set consists of 42 handshape HSIds for *both* occlusion and non-occlusion cases. For the number of classes per experiment see Table 5.

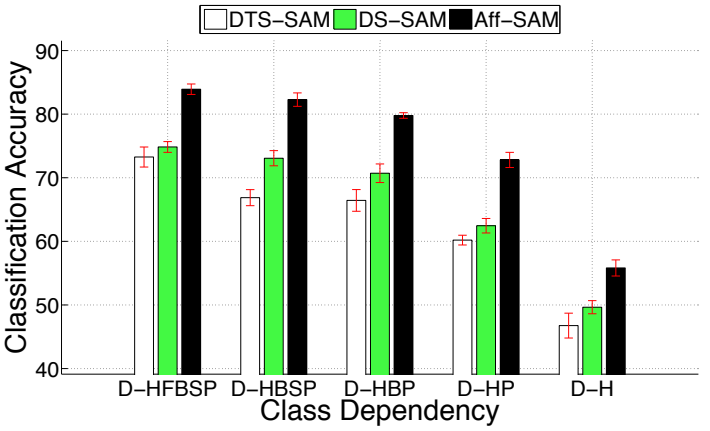


Figure 15: Classification experiments for both occluded and non-occluded cases. Classification Accuracy by varying the dependency of each class w.r.t. to the annotation parameters [H,F,B,S,P] (x-axis) and the feature employed (legend). For the numbers of classes per experiment see Table 5.

Aff-SAM outperforms both DS-SAM and DST-SAM obtaining on average 10% performance increase in all cases (Figure 15). This indicates that Aff-SAM handles handshape classification obtaining decent results even during occlusions. The performance for the other baseline methods is not shown since they cannot handle occlusions and the results are lower. The comparisons with the two more competitive methods show the differential gain due to the *claimed* contributions of the Aff-SAM. By making our models independent to 3D pose orientation, that is,-H, the classification performance decreases. This makes sense since by taking into consideration the occlusion cases the variability of the handshapes' 3D pose increases; as a consequence the classification task is more difficult. Moreover, the classification during occlusions may already include errors at the visual modeling level concerning the estimated occluded handshape. In this experiment, the range of 3D pose variations is larger than the amount handled by the affine transforms of the Aff-SAM.

8. Sign Recognition

Next, we evaluate the Aff-SAM approach, on automatic sign recognition experiments, while fusing with movement/position cues, as well as concerning its application on multiple signers. The experiments are applied on data from the GSL lexicon corpus (DictaSign, 2012). By employing the presented framework for tracking and feature extraction (Section 3) we extract the Aff-SAM features (Section 4). These are then employed to construct data-driven subunits as in Roussos et al. (2010b) and Theodorakis et al. (2012), which are further statistically trained. The lexicon corpus contains data from two different signers, A and B. Given the Aff-SAM based models from signer A

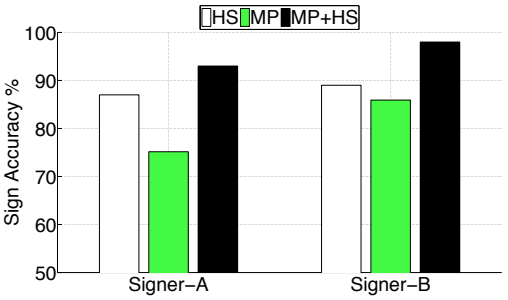


Figure 16: Sign recognition in GSL lemmas corpus employing 100 signs for each signer A and B, and multiple cues: Hanshape (HS), Movement-Position (MP) cue and MP+HS fusion between both via Parallel HMMs.

these are then adapted and fitted to another signer (B) as in Section 5 for which no Aff-SAM models have been trained. The features resulting as a product of the visual level adaptation, are employed next in the recognition experiment. For signer A, the features are extracted from the signer's own model. Note that, there are other aspects concerning signer adaptation during SL recognition, as for instance the manner of signing or the different pronunciations, which are not within the focus of this article.

GSL Lemmas: We employ 100 signs from the *GSL lemmas corpus*. These are articulated in isolation with five repetitions each, from two native signers (male and female). The videos have a uniform background and a resolution of 1440x1080 pixels, recorded at 25 fps.

8.1. Sub-unit Modeling and Sign Recognition

The SL recognition framework consists of the following: 1) First by employing the movement-position cue we construct dynamic/static SUs based on dynamic and static discrimination (Pitsikalis et al., 2010; Theodorakis et al., 2012). 2) Second we employ the handshape features and the sub-unit construction via clustering of the handshape features (Roussos et al., 2010b). 3) We then create one lexicon for each information cue, that is, movement-position and handshape. For the movement-position lexicon we recompose the constructed dynamic/static SUs, whereas for the Handshape lexicon we recompose the handshape subunits (HSU) to form each sign realization. 4) Next, for the training of the SUs we employ a GMM for the static and handshape subunits and an 5-state HMM for the dynamic subunits. Concerning the training, we employ four realizations for each sign for training and one for testing. 5) Finally, we fuse the movement-position and handshape cues via one possible late integration scheme, that is Parallel HMMs (PaHMMs) (Vogler and Metaxas, 1999).

8.2. Sign Recognition Results

In Figure 16 we present the sign recognition performance on the GSL lemmas corpus employing 100 signs from two signers, A and B, while varying the cues employed: movement-position (MP), handshape (HS) recognition performance and the fusion of both MP+HS cues via PaHMMs. For both signers A and B, handshape-based recognition outperforms the one of movement-position cue. This is expected, and indicates that handshape cue is crucial for sign recognition. Nevertheless, the main result we focus is the following: The sign recognition performance in Signer-B is similar to Signer-A, where the Aff-SAM model has been trained. Thus by applying the affine adaptation procedure and employing only a small development set, as presented in Section 5 we can extract reliable handshape features for multiple signers. As a result, when both cues are employed, and for both signers, the recognition performance increases, leading to a 15% and 7.5% absolute improvement w.r.t. the single cues respectively.

9. Conclusions

In this paper, we propose a new framework that incorporates dynamic affine-invariant Shape - Appearance modeling and feature extraction for handshape classification. The proposed framework leads to the extraction of effective features for hand configurations. The main contributions of this work are the following: 1) We employ Shape-Appearance hand images for the representation of the hand configurations. These images are modeled with a linear combination of eigenimages followed by an affine transformation, which effectively accounts for some 3D hand pose variations. 2) In order to achieve robustness w.r.t. occlusions, we employ a regularized fitting of the SAM that exploits prior information on the handshape and its dynamics. This process outputs an accurate tracking of the hand as well as descriptive handshape features. 3) We introduce an affine-adaptation for different signers than the signer that was used to train the model. 4) All the above features are integrated in a statistical handshape classification GMM and a sign recognition HMM-based system.

The overall visual feature extraction and classification framework is evaluated on classification experiments as well as on sign recognition experiments. These explore multiple tasks of gradual difficulty in relation to the orientation parameters, as well as both occlusion and non-occlusion cases. We compare with existing baseline features as well as with more competitive features, which are implemented as simplifications of the proposed SAM method. We investigate the quality of the feature spaces and evaluate the compactness-separation of the different features in which the proposed features show superiority. The Aff-SAM features yield improvements in classification accuracy too. For the non-occlusion cases, these are on average 35% over the baseline methods (FD, RB, M) and 3% over the most competitive SAM methods (DS-SAM, DST-SAM). Furthermore, when we also consider the occlusion cases, the improvements in classification accuracy are on average 9.7% over the most competitive SAM methods (DS-SAM, DST-SAM). Although DS-SAM yields similar performance in some cases, it under-performs in the more difficult and extended data set classification tasks. On the

task of sign recognition for a 100-sign lexicon of GSL lemmas, the approach is evaluated via handshape subunits and also fused with movement-position cues, leading to promising results. Moreover, it is shown to have similar results, even if we do not train an explicit signer dependent Aff-SA model, given the introduction of the affine-signer adaptation component. In this way, the approach can be easily applicable to multiple signers.

To conclude with, given that handshape is among the main sign language phonetic parameters, we address issues that are indispensable for automatic sign language recognition. Even though the framework is applied on SL data, its application is extendable on other gesture-like data. The quantitative evaluation and the intuitive results presented show the perspective of the proposed framework for further research.

Acknowledgments

This research work was supported by the EU under the research program Dictasign with grant FP7-ICT-3-231135. A. Roussos was also supported by the ERC Starting Grant 204871-HUMANIS.

Appendix A. Details about the Regularized Fitting Algorithm

We provide here details about the algorithm of the regularized fitting of the shape-appearance model. The total energy $E(\lambda, p)$ that is to be minimized can be written as (after a multiplication with N_M that does not affect the optimum parameters):

$$\begin{aligned} J(\lambda,p) = & \sum_x \left\{ A_0(x) + \sum_{i=1}^{N_c} \lambda_i A_i(x) - f(W_p(x)) \right\}^2 + \\ & \frac{N_M}{N_c} \left(w_S \|\lambda - \lambda_0\|^2_{\Sigma_\lambda} + w_D \|\lambda - \lambda^e\|^2_{\Sigma_{\varepsilon_\lambda}} \right) + \\ & \frac{N_M}{N_p} \left(w_S \|p - p_0\|^2_{\Sigma_p} + w_D \|p - p^e\|^2_{\Sigma_{\varepsilon_p}} \right) . \end{aligned} \tag{4}$$

If σ_{λ_i}, $\sigma_{\tilde{p}_i}$ are the standard deviations of the components of the parameters λ, $\widetilde{p}$ respectively and $\sigma_{\varepsilon_{\lambda,i}}$, $\sigma_{\varepsilon_{\tilde{p},i}}$ are the standard deviations of the components of the parameters' prediction errors ε_λ, $\varepsilon_{\tilde{p}}$, then the corresponding covariance matrices Σ_λ, $\Sigma_{\tilde{p}}$, $\Sigma_{\varepsilon_\lambda}$, $\Sigma_{\varepsilon_{\tilde{p}}}$, which are diagonal, can be written as:

$$\begin{aligned} & \Sigma_\lambda = \mathrm{diag}(\sigma^2_{\lambda_1}, \ldots, \sigma^2_{\lambda_{N_c}}), \Sigma_{\tilde{p}} = \mathrm{diag}(\sigma^2_{\tilde{p}_1}, \ldots, \sigma^2_{\tilde{p}_{N_c}}), \\ & \Sigma_{\varepsilon_\lambda} = \mathrm{diag}(\sigma^2_{\varepsilon_{\lambda,1}}, \ldots, \sigma^2_{\varepsilon_{\lambda,N_c}}), \Sigma_{\varepsilon_{\tilde{p}}} = \mathrm{diag}(\sigma^2_{\varepsilon_{\tilde{p},1}}, \ldots, \sigma^2_{\varepsilon_{\tilde{p},N_p}}). \end{aligned}$$

The squared norms of the prior terms in Equation (4) are thus given by:

$$\|\lambda - \lambda_0\|^2_{\Sigma_\lambda} = \sum_{i=1}^{N_c} \left(\frac{\lambda_i}{\sigma_{\lambda_i}} \right)^2 ,$$

$$\|\lambda-\lambda^e\|^2_{\Sigma_{\varepsilon_\lambda}}=\sum_{i=1}^{N_c}\left(\frac{\lambda_i-\lambda_i^e}{\sigma_{\varepsilon_{\lambda,i}}}\right)^2,$$

$$\|p-p_0\|^2_{\Sigma_p}=(p-p_0)^T U_p\Sigma_{\tilde{p}}^{-1}U_p^T(p-p_0)=\|\tilde{p}\|^2_{\Sigma_{\tilde{p}}}=\sum_{i=1}^{N_p}\left(\frac{\tilde{p}_i}{\sigma_{\tilde{p}_i}}\right)^2,$$

$$\|p-p^e\|^2_{\Sigma_{\varepsilon_p}}=\|\tilde{p}-\tilde{p}^e\|^2_{\Sigma_{\varepsilon_{\tilde{p}}}}=\sum_{i=1}^{N_p}\left(\frac{\tilde{p}_i-\tilde{p}_i^e}{\sigma_{\varepsilon_{\tilde{p},i}}}\right)^2.$$

Therefore, if we set:

$$m_1=\sqrt{w_S N_M/N_c}\,,\ m_2=\sqrt{w_D N_M/N_c}\,,$$
$$m_3=\sqrt{w_S N_M/N_p}\,,\ m_4=\sqrt{w_D N_M/N_p}\,,$$

the energy in Equation (4) takes the form:

$$J(\lambda,p)=\sum_x\left\{A_0(x)+\sum_{i=1}^{N_c}\lambda_i A_i(x)-f(W_p(x))\right\}^2+\sum_{i=1}^{N_G}G_i^2(\lambda,p)\,, \tag{5}$$

with $G_i(\lambda,p)$ being $N_G=2N_c+2N_p$ prior functions defined by:

$$G_i(\lambda,p)=\begin{cases} m_1\frac{\lambda_i}{\sigma_{\lambda_i}}, & 1\le i\le N_c \\ m_2\frac{\lambda_j-\lambda_j^e}{\sigma_{\varepsilon_{\lambda,j}}},\ j=i-N_c, & N_c+1\le i\le 2N_c \\ m_3\frac{\tilde{p}_j}{\sigma_{\tilde{p}_j}},\ j=i-2N_c, & 2N_c+1\le i\le 2N_c+N_p \\ m_4\frac{\tilde{p}_j-\tilde{p}_j^e}{\sigma_{\varepsilon_{\tilde{p},j}}},\ j=i-2N_c-N_p, & 2N_c+N_p+1\le i\le 2N_c+2N_p \end{cases}. \tag{6}$$

Each component $\tilde{p}_j$, $j=1,\ldots,N_p$, of the re-parametrization of p can be written as:

$$\tilde{p}_j=v_{\tilde{p}_j}^T(p-p_0)\,, \tag{7}$$

where $v_{\tilde{p}_j}$ is the j-th column of U_p, that is the eigenvector of the covariance matrix Σ_p that corresponds to the j-th principal component $\tilde{p}_j$.

In fact, the energy $J(\lambda,p)$, Equation (5), for general prior functions $G_i(\lambda,p)$, has exactly the same form as the energy that is minimized by the algorithm of Baker et al. (2004). Next, we describe this algorithm and then we specialize it in the specific case of our framework.

A.1. Simultaneous Inverse Compositional Algorithm with a Prior

We briefly present here the algorithm *simultaneous inverse compositional with a prior* (SICP) (Baker et al., 2004). This is a *Gauss-Newton* algorithm that finds a local minimum of the energy $J(\lambda,p)$ (5) for general cases of prior functions $G_i(\lambda,p)$ and warps $W_p(x)$ that are controlled by some parameters p.

The algorithm starts from some initial estimates of λ and p. Afterwards, in every iteration, the previous estimates of λ and p are updated to λ' and p' as follows. It is considered that a vector $\Delta\lambda$ is added to λ:

$$\lambda' = \lambda + \Delta\lambda \tag{8}$$

and a warp with parameters Δp is applied to the synthesized image $A_0(x) + \sum \lambda_i A_i(x)$. As an approximation, the latter is taken as equivalent to updating the warp parameters from p to p' by composing $W_p(x)$ with the inverse of $W_{\Delta p}(x)$:

$$W_{p'} = W_p \circ W_{\Delta p}^{-1} \, . \tag{9}$$

From the above relation, given that p is constant, p' can be expressed as a $\mathbb{R}^{N_p} \rightarrow \mathbb{R}^{N_p}$ function of Δp, $p' = p'(\Delta p)$, with $p'(\Delta p = 0) = p$. Further, $p'(\Delta p)$ is approximated with a first order Taylor expansion around $\Delta p = 0$:

$$p'(\Delta p) = p + \frac{\partial p'}{\partial \Delta p} \Delta p \, . \tag{10}$$

where $\frac{\partial p'}{\partial \Delta p}$ is the Jacobian of the function $p'(\Delta p)$, which generally depends on Δp.

Based on the aforementioned type of updates of λ and p as well as the considered approximations, the values $\Delta\lambda$ and Δp are specified by minimizing the following energy:

$$\begin{aligned} F(\Delta\lambda, \Delta p) = \sum_x \Bigg\{ & A_0\big(W_{\Delta p}(x)\big) + \sum_{i=1}^{N_c} (\lambda_i + \Delta\lambda_i) A_i\big(W_{\Delta p}(x)\big) \\ & - f\big(W_p(x)\big) \Bigg\}^2 + \sum_{i=1}^{N_G} G_i^2 \left(\lambda + \Delta\lambda, p + \frac{\partial p'}{\partial \Delta p} \Delta p \right) \, , \end{aligned}$$

simultaneously with respect to $\Delta\lambda$ and Δp. By applying first order Taylor approximations on the two terms of the above energy $F(\lambda, p)$, one gets:

$$\begin{aligned} F(\Delta\lambda, \Delta p) \approx & \sum_x \left\{ E_{sim}(x) + SD_{sim}(x) \begin{pmatrix} \Delta\lambda \\ \Delta p \end{pmatrix} \right\}^2 + \\ & \sum_{i=1}^{N_G} \left\{ G_i(\lambda, p) + SD_{G_i} \begin{pmatrix} \Delta\lambda \\ \Delta p \end{pmatrix} \right\}^2 , \end{aligned} \tag{11}$$

where $E_{sim}(x)$ is the image of reconstruction error evaluated at the model domain:

$$E_{sim}(x) = A_0(x) + \sum_{i=1}^{N_c} \lambda_i A_i(x) - f\big(W_p(x)\big)$$

and $SD_{sim}(x)$ is a vector-valued "steepest descent" image with $N_c + N_p$ channels, each one of them corresponding to a specific component of the parameter vectors λ and p:

$$SD_{sim}(x) = \left[A_1(x), ..., A_{N_c}(x), \left(\nabla A_0(x) + \sum_{i=1}^{N_c} \lambda_i \nabla A_i(x) \right) \frac{\partial W_p(x)}{\partial p} \right] , \tag{12}$$

where the gradients $\nabla A_i(x) = \left[\frac{\partial A_i}{\partial x_1}, \frac{\partial A_i}{\partial x_2}\right]$ are considered as row vector functions. Also SD_{G_i}, for each $i = 1,..,N_G$, is a row vector with dimension $N_c + N_p$ that corresponds to the steepest descent direction of the prior term $G_i(\lambda, p)$:

$$SD_{G_i} = \left(\frac{\partial G_i}{\partial \lambda}, \frac{\partial G_i}{\partial p}\frac{\partial p'}{\partial \Delta p}\right) . \tag{13}$$

The approximated energy $F(\lambda, p)$ (11) is quadratic with respect to both $\Delta\lambda$ and Δp, therefore the minimization can be done analytically and leads to the following solution:

$$\begin{pmatrix} \Delta\lambda \\ \Delta p \end{pmatrix} = -H^{-1}\left[\sum_x SD^T_{sim}(x)E_{sim}(x) + \sum_{i=1}^{N_G} SD^T_{G_i} G_i(\lambda, p)\right], \tag{14}$$

where H is the matrix (which approximates the Hessian of F):

$$H = \sum_x SD^T_{sim}(x)SD_{sim}(x) + \sum_{i=1}^{N_G} SD^T_{G_i} SD_{G_i} .$$

In conclusion, in every iteration of the SICP algorithm, the Equation (14) is applied and the parameters λ and p are updated using Equations (8) and (10). This process terminates when a norm of the update vector $\begin{pmatrix} \Delta\lambda \\ \Delta p \end{pmatrix}$ falls below a relatively small threshold and then it is considered that the process has converged.

A.1.1. COMBINATION WITH LEVENBERG-MARQUARDT ALGORITHM

In the algorithm described above, there is no guarantee that the original energy (5), that is the objective function before any approximation, decreases in every iteration; it might increase if the involved approximations are not accurate. Therefore, following Baker and Matthews (2002), we use a modification of this algorithm by combining it with the *Levenberg-Marquardt* algorithm: In Equation (14) that specifies the updates, we replace the Hessian approximation H by $H + \delta\,\mathrm{diag}(H)$, where δ is a positive weight and $\mathrm{diag}(H)$ is the diagonal matrix that contains the diagonal elements of H. This corresponds to an interpolation between the updates given by the Gauss-Newton algorithm and weighted gradient descent. As δ increases, the algorithm has a behavior closer to gradient descent, which means that from the one hand is slower but from the other hand yields updates that are more reliable, in the sense that the energy will eventually decrease for sufficiently large δ.

In every iteration, we specify the appropriate weight δ as follows. Starting from setting δ to $1/10$ of its value in the previous iteration (or from $\delta = 0.01$ if this is the first iteration), we compute the updates $\Delta\lambda$ and Δp using the Hessian approximation $H + \delta\,\mathrm{diag}(H)$ and then evaluate the original energy (5). If the energy has decreased we keep the updates and finish the iteration. If the energy has increased, we set $\delta \to 10\,\delta$ and try again. We repeat that step until the energy decreases.

A.2. Specialization in the Current Framework

In this section, we derive the SICP algorithm for the special case that concerns our method. This case arises when 1) the general warps $W_p(x)$ are specialized to affine transforms and 2) the general prior functions $G_i(\lambda, p)$ are given by Equation (6).

A.2.1. The Case of Affine Transforms

In our framework, the general warps $W_p(x)$ of the SICP algorithm are specialized to affine transforms with parameters $p = (p_1 \cdots p_6)$ that are defined by:

$$W_p(x,y) = \begin{pmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

In this special case, which is analyzed also in Baker et al. (2004), the Jacobian $\frac{\partial W_p(x)}{\partial p}$ that is used in Equation (12) is given by:

$$\frac{\partial W_p(x)}{\partial p} = \begin{pmatrix} x_1 & 0 & x_2 & 0 & 1 & 0 \\ 0 & x_1 & 0 & x_2 & 0 & 1 \end{pmatrix}.$$

The restriction to affine transforms implies also a special form for the Jacobian $\frac{\partial p'}{\partial \Delta p}$ that is used in Equation (13). More precisely, as described in Baker et al. (2004), a first order Taylor approximation is first applied to the inverse warp $W_{\Delta p}^{-1}$ and yields $W_{\Delta p}^{-1} \approx W_{-\Delta p}$. Afterwards, based on Equation (9) and the fact that the parameters of a composition $W_r = W_p \circ W_q$ of two affine transforms are given by:

$$r = \begin{pmatrix} p_1 + q_1 + p_1 q_1 + p_3 q_2 \\ p_2 + q_2 + p_2 q_1 + p_4 q_2 \\ p_3 + q_3 + p_1 q_3 + p_3 q_4 \\ p_4 + q_4 + p_2 q_3 + p_4 q_4 \\ p_5 + q_5 + p_1 q_5 + p_3 q_6 \\ p_6 + q_6 + p_2 q_5 + p_4 q_6 \end{pmatrix},$$

the function $p'(\Delta p)$ (10) is approximated as:

$$p'(\Delta p) = \begin{pmatrix} p_1 - \Delta p_1 - p_1 \Delta p_1 - p_3 \Delta p_2 \\ p_2 - \Delta p_2 - p_2 \Delta p_1 - p_4 \Delta p_2 \\ p_3 - \Delta p_3 - p_1 \Delta p_3 - p_3 \Delta p_4 \\ p_4 - \Delta p_4 - p_2 \Delta p_3 - p_4 \Delta p_4 \\ p_5 - \Delta p_5 - p_1 \Delta p_5 - p_3 \Delta p_6 \\ p_6 - \Delta p_6 - p_2 \Delta p_5 - p_4 \Delta p_6 \end{pmatrix}.$$

Therefore, its Jacobian is given by:

$$\frac{\partial p'}{\partial \Delta p} = -\begin{pmatrix} 1+p_1 & p_3 & 0 & 0 & 0 & 0 \\ p_2 & 1+p_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1+p_1 & p_3 & 0 & 0 \\ 0 & 0 & p_2 & 1+p_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1+p_1 & p_3 \\ 0 & 0 & 0 & 0 & p_2 & 1+p_4 \end{pmatrix}.$$

A.2.2. SPECIFIC TYPE OF PRIOR FUNCTIONS

Apart from the restriction to affine transforms, in the proposed framework of the regularized shape-appearance model fitting, we have derived the specific formulas of Equation (6) for the prior functions $G_i(\lambda, p)$ of the energy $J(\lambda, p)$ in Equation (5). Therefore, in our case, their partial derivatives, which are involved in the above described SICP algorithm (see Equation (13)), are specialized as follows:

$$\frac{\partial G_i}{\partial p} \overset{(7)}{=} \begin{cases} 0, & 1 \le i \le 2N_c \\ \frac{m_3}{\sigma_{\tilde{p}_j}} v_{\tilde{p}_j}^T, j = i - 2N_c, & 2N_c + 1 \le i \le 2N_c + N_p \\ \frac{m_4}{\sigma_{\varepsilon_{\tilde{p},j}}} v_{\tilde{p}_j}^T, j = i - 2N_c - N_p, & 2N_c + N_p + 1 \le i \le 2N_c + 2N_p \end{cases},$$

$$\frac{\partial G_i}{\partial \lambda} = \begin{cases} \frac{m_1}{\sigma_{\lambda_i}} e_i^T, & 1 \le i \le N_c \\ \frac{m_2}{\sigma_{\varepsilon_{\lambda,j}}} e_j^T, j = i - N_c, & N_c + 1 \le i \le 2N_c \\ 0, & 2N_c + 1 \le i \le 2N_c + 2N_p \end{cases},$$

where e_i, $1 \le i \le N_c$, is the i-th column of the $N_c \times N_c$ identity matrix.

References

U. Agris, J. Zieren, U. Canzler, B. Bauer, and K. F. Kraiss. Recent developments in visual sign language recognition. *Universal Access in the Information Society*, 6: 323–362, 2008.

T. Ahmad, C.J. Taylor, and T.F. Lanitis, A. Cootes. Tracking and recognising hand gestures, using statistical shape models. *Image and Visual Computing*, 15(5):345–352, 1997.

A. Argyros and M. Lourakis. Real time tracking of multiple skin-colored objects with a possibly moving camera. In *Proceedings of the European Conference on Computer Vision*, 2004.

V. Athitsos and S. Sclaroff. An appearance-based framework for 3d hand shape classification and camera viewpoint estimation. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pages 45–52, 2002.

S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework: Part 1. Technical report, Carnegie Mellon University, 2002.

S. Baker, R. Gross, and I. Matthews. Lucas-kanade 20 years on: A unifying framework: Part 4. Technical report, Carnegie Mellon University, 2004.

B. Bauer and K. F. Kraiss. Towards an automatic sign language recognition system using subunits. In *Proceedings of the International Gesture Workshop*, volume 2298, pages 64–75, 2001.

H. Birk, T.B. Moeslund, and C.B. Madsen. Real-time recognition of hand alphabet gestures using principal component analysis. In *Proceedings of the Scandinavian Conference Image Analysis*, 1997.

A. Blake and M. Isard. *Active Contours*. Springer, 1998.

R. Bowden and M. Sarhadi. A nonlinear model of shape and motion for tracking fingerspelt american sign language. *Image and Visual Computing*, 20:597–607, 2002.

P. Buehler, M. Everingham, and A. Zisserman. Learning sign language by watching TV (using weakly aligned subtitles). In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2009.

J. Cai and A. Goshtasby. Detecting human faces in color images. *Image and Visual Computing*, 18:63–75, 1999.

F.-S. Chen, C.-M. Fu, and C.-L. Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and Visual Computing*, 21(8): 745–758, 2003.

S. Conseil, S. Bourennane, and L. Martin. Comparison of Fourier descriptors and Hu moments for hand posture recognition. In *Proceedings of the European Conference on Signal Processing*, 2007.

T.F. Cootes and C.J. Taylor. Statistical models of appearance for computer vision. Technical report, University of Manchester, 2004.

Y. Cui and J. Weng. Appearance-based hand sign recognition from intensity image sequences. *Computer Vision and Image Understanding*, 78(2):157–176, 2000.

L. Davies, David and W. Bouldin, Donald. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:224 – 227, April 1979.

J.-W. Deng and H.T. Tsui. A novel two-layer PCA/MDA scheme for hand posture recognition. In *Proceedings of the International Conference on Pattern Recognition*, volume 1, pages 283–286, 2002.

DictaSign. Greek sign language corpus. http://www.sign-lang.uni-hamburg.de/dictasign/portal, 2012.

L. Ding and A. M. Martinez. Modelling and recognition of the linguistic components in american sign language. *Image and Visual Computing*, 27(12):1826 – 1844, 2009.

P. Dreuw, J. Forster, T. Deselaers, and H. Ney. Efficient approximations to model-based joint tracking and recognition of continuous sign language. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, Sep. 2008.

I.L. Dryden and K.V. Mardia. *Statistical Shape Analysis*. John Wiley and Sons, 1998.

W. Du and J. Piater. Hand modeling and tracking for video-based sign language recognition by robust principal component analysis. In *Proceedings of the ECCV Workshop on Sign, Gesture and Activity*, September 2010.

A. Farhadi, D. Forsyth, and R. White. Transfer learning in sign language. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

H. Fillbrandt, S. Akyol, and K.-F. Kraiss. Extraction of 3D hand shape and posture from images sequences from sign language recognition. In *Proceedings of the International Workshop on Analysis and Modeling of Faces and Gestures*, pages 181–186, 2003.

R. Gross, I. Matthews, and S. Baker. Generic vs. person specific active appearance models. *Image and Visual Computing*, 23(12):1080–1093, 2005.

T. Hanke. HamNoSys Representing sign language data in language resources and language processing contexts. In *Proceedings of the International Conference on Language Resources and Evaluation*, 2004.

M.-K. Hu. Visual pattern recognition by moment invariants. *IEEE Transactions on Information Theory*, 8(2):179–187, February 1962.

C.-L. Huang and S.-H. Jeng. A model-based hand gesture recognition system. *Machine Vision and Application*, 12(5):243–258, 2001.

P. Kakumanu, S. Makrogiannis, and N. Bourbakis. A survey of skin-color modeling and detection methods. *Pattern Recognition*, 40(3):1106–1122, Mar. 2007.

E. Learned-Miller. Data driven image models through continuous joint alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):236–250, 2005.

S. Liwicki and M. Everingham. Automatic recognition of fingerspelled words in British sign language. In *Proceedings of the CVPR Workshop on Human Communicative Behavior Analysis*, 2009.

P. Maragos. *Morphological Filtering for Image Enhancement and Feature Detection*, chapter The Image and Video Processing Handbook. Elsevier, 2005.

I. Matthews and S. Baker. Active appearance models revisited. *International Journal of Computer Vision*, 60(2):135–164, 2004.

C. Neidle. Signstream annotation: Addendum to conventions used for the american sign language linguistic research project. Technical report, 2007.

C. Neidle and C. Vogler. A new web interface to facilitate access to corpora: development of the ASLLRP data access interface. In *Proceedings of the International Conference on Language Resources and Evaluation*, 2012.

E.J. Ong, H. Cooper, N. Pugeault, and R. Bowden. Sign language recognition using sequential pattern trees. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 2200–2207. IEEE, 2012.

Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma. RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2010.

V. Pitsikalis, S. Theodorakis, and P. Maragos. Data-driven sub-units and modeling structure for continuous sign language recognition with multiple cues. In *LREC Workshop Repr. & Proc. SL: Corpora and SL Technologies*, 2010.

L. R. Rabiner and R.W. Schafer. Introduction to digital speech processing. *Foundations and Trends in Signal Processing*, 1(1-2):1–194, 2007.

A. Roussos, S. Theodorakis, V. Pitsikalis, and P. Maragos. Affine-invariant modeling of shape-appearance images applied on sign language handshape classification. In *Proceedings of the International Conference on Image Processing*, Sep. 2010a.

A. Roussos, S. Theodorakis, V. Pitsikalis, and P. Maragos. Hand tracking and affine shape-appearance handshape sub-units in continuous sign language recognition. In *Proceedings of the ECCV Workshop on Sign, Gesture and Activity*, September 2010b.

J. Sherrah and S. Gong. Resolving visual uncertainty and occlusion through probabilistic reasoning. In *Proceedings of the British Machine Vision Conference*, pages 252–261, 2000.

P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer, 2004.

T. Starner, J. Weaver, and A. Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, Dec. 1998.

B. Stenger, A. Thayananthan, P.H.S Torr, and R. Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1372–1384, Sep. 2006.

G.J. Sweeney and A.C. Downton. Towards appearance-based multi-channel gesture recognition. In *Proceedings of the International Gesture Workshop*, pages 7–16, 1996.

N. Tanibata, N. Shimada, and Y. Shirai. Extraction of hand features for recognition of sign language words. In *Proceedings of the International Conference on Vision Interface*, pages 391–398, 2002.

J. Terrillon, M. Shirazi, H. Fukamachi, and S. Akamatsu. Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pages 54–61, 2000.

A. Thangali, J.P. Nash, S. Sclaroff, and C. Neidle. Exploiting phonological constraints for handshape inference in asl video. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 521–528. IEEE, 2011.

S. Theodorakis, V. Pitsikalis, and P. Maragos. Advances in dynamic-static integration of movement and handshape cues for sign language recognition. In *Proceedings of the International Gesture Workshop*, 2011.

S. Theodorakis, V. Pitsikalis, I. Rodomagoulakis, and P. Maragos. Recognition with raw canonical phonetic movement and handshape subunits on videos of continuous sign language. In *Proceedings of the International Conference on Image Processing*, 2012.

M. Viola and M. J. Jones. Fast multi-view face detection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2003.

C. Vogler and D. Metaxas. Parallel hidden markov models for american sign language recognition. In *Proceedings of the International Conference on Computer Vision*, volume 1, pages 116–122, 1999.

Y. Wu and T.S. Huang. View-independent recognition of hand postures. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, volume 2, pages 88–94, 2000.

M.-H. Yang, N. Ahuja, and M. Tabb. Extraction of 2d motion trajectories and its application to hand gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1061–1074, Aug. 2002.

S. Young, D. Kershaw, J. Odell, D. Ollason, V. Woodland, and P. Valtchevand. *The HTK Book*. Entropic Ltd., 1999.

J. Zieren, N. Unger, and S. Akyol. Hands tracking from frontal view for vision-based gesture recognition. In *Pattern Recognition*, LNCS, pages 531–539, 2002.

JMLR: Workshop and Conference Proceedings 13:2655–2682, 2012 Published 10/12

Discriminative Hierarchical Part-based Models for Human Parsing and Action Recognition

Yang Wang YWANG@CS.UMANITOBA.CA
Department of Computer Science
University of Manitoba
Winnipeg, MB, R3T 2N2, Canada
Duan Tran DDTRAN2@UIUC.EDU
Zicheng Liao LIAO17@UIUC.EDU
David Forsyth DAF@UIUC.EDU
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA

Editor: Isabelle Guyon and Vassilis Athitsos

Abstract

We consider the problem of parsing human poses and recognizing their actions in static images with part-based models. Most previous work in part-based models only considers rigid parts (e.g., torso, head, half limbs) guided by human anatomy. We argue that this representation of parts is not necessarily appropriate. In this paper, we introduce hierarchical poselets—a new representation for modeling the pose configuration of human bodies. Hierarchical poselets can be rigid parts, but they can also be parts that cover large portions of human bodies (e.g., torso + left arm). In the extreme case, they can be the whole bodies. The hierarchical poselets are organized in a hierarchical way via a structured model. Human parsing can be achieved by inferring the optimal labeling of this hierarchical model. The pose information captured by this hierarchical model can also be used as a intermediate representation for other high-level tasks. We demonstrate it in action recognition from static images.

Keywords: human parsing, action recognition, part-based models, hierarchical poselets, max-margin structured learning

1. Introduction

Modeling human bodies (or articulated objects in general) in images is a long-lasting problem in computer vision. Compared with rigid objects (e.g., faces and cars) which can be reasonably modeled using several prototypical templates, human bodies are much more difficult to model due to the wide variety of possible pose configurations.

A promising solution for dealing with the pose variations is to use part-based models. Part-based representations, such as cardboard people (Ju et al., 1996) or pictorial structure (Felzenszwalb and Huttenlocher, 2005), provide an elegant framework for modeling articulated objects, such as human bodies. A part-based model represents

the human body as a constellation of a set of rigid parts (e.g., torso, head, half limbs) constrained in some fashion. The typical constraints used are tree-structured kinematic constraints between adjacent body parts, for example, torso-upper half-limb connection, or upper-lower half-limb connection. Part-based models consist of two important components: (1) part appearances specifying what each body part should look like in the image; (2) configuration priors specifying how parts should be arranged relative to each other. Part-based models have been used extensively in various computer vision applications involving humans, such as human parsing (Felzenszwalb and Huttenlocher, 2005; Ramanan, 2006), kinematic tracking (Ramanan et al., 2005), action recognition (Yang et al., 2010) and human-object interaction (Yao and Fei-Fei, 2010).

Considerable progress has been made to improve part-based models. For example, there has been a line of work on using better appearance models in part-based models. A representative example is the work by Ramanan (2006), who learns color histograms of parts from an initial edge-based model. Ferrari et al. (2008) and Eichner and Ferrari (2009) further improve the part appearance models by reducing the search space using various tricks, for example, the relative locations of part locations with respect to a person detection and the relationship between different part appearances (e.g., upper-arm and torso tend to have the same color), Andriluka et al. (2009) build better edge-based appearance models using the HOG descriptors (Dalal and Triggs, 2005). Sapp et al. (2010b) develop efficient inference algorithm to allow the use of more expensive features. There is also work (Johnson and Everingham, 2009; Mori et al., 2004; Mori, 2005; Srinivasan and Shi, 2007) on using segmentation as a pre-processing step to provide better spatial support for computing part appearances.

Another line of work is on improving configuration priors in part-based models. Most of them focus on developing representations and fast inference algorithms that by-pass the limitations of kinematic tree-structured spatial priors in standard pictorial structure models. Examples include common-factor models (Lan and Huttenlocher, 2005), loopy graphs (Jiang and Martin, 2008; Ren et al., 2005; Tian and Sclaroff, 2010; Tran and Forsyth, 2010), mixtures of trees (Wang and Mori, 2008). There is also work on building spatial priors that adapt to testing examples (Sapp et al., 2010a).

Most of the previous work on part-based models use rigid parts that are anatomically meaningful, for example, torso, head, half limbs. Those rigid parts are usually represented as rectangles (e.g., Andriluka et al. 2009; Felzenszwalb and Huttenlocher 2005; Ramanan 2006; Ren et al. 2005; Sigal and Black 2006; Wang and Mori 2008) or parallel lines (e.g., Ren et al. 2005). However, as pointed out by some recent work (Bourdev and Malik, 2009; Bourdev et al., 2010), rigid parts are not necessarily the best representation since rectangles and parallel lines are inherently difficult to detect in natural images.

In this paper, we introduce a presentation of parts inspired by the early work of Marr (1982). The work in Marr (1982) recursively represents objects as generalized cylinders in a coarse-to-fine hierarchical fashion. In this paper, we extend Marr's idea for two problems in the general area of "looking at people". The first problem is human parsing, also known as human pose estimation. The goal is to find the location of

each body part (torso, head, limbs) of a person in a static image. We use a part-based approach for human parsing. The novelty of our work is that our notion of "parts" can range from basic rigid parts (e.g., torso, head, half-limb), to large pieces of bodies covering more than one rigid part (e.g., torso + left arm). In the extreme case, we have "parts" corresponding to the whole body. We propose a new representation called "hierarchical poselets" to capture this hierarchy of parts. We infer the human pose using this hierarchical representation.

The hierarchical poselet also provides rich information about body poses that can be used in other applications. To demonstrate this, we apply it to recognize human action in static images. In this application, we use hierarchical poselets to capture various pose information of the human body, this information is further used as some intermediate representation to infer the action of the person.

A preliminary version of this work appeared in Wang et al. (2011). We organize the rest of the paper as follows. Section 2 reviews previous work in human parsing and action recognition. Section 3 introduces hierarchical poselet, a new representation for modeling human body configurations. Section 4 describes how to use hierarchical poselets for human parsing. Section 5 develops variants of hierarchical poselets for recognizing human action in static images. We present experimental results on human parsing and action recognition in Section 6 and conclude in Section 7.

2. Previous Work

Finding and understanding people from images is a very active area in computer vision. In this section, we briefly review previous work in human parsing and action recognition that is most related to our work.

Human parsing: Early work related to finding people from images is in the setting of detecting and tracking people with kinematic models in both 2D and 3D. Forsyth et al. (2006) provide an extensive survey of this line of work.

Recent work has examined the problem in static images. Some of these approaches are exemplar-based. For example, Toyama and Blake (2001) track people using 2D exemplars. Mori and Malik (2002) and Sullivan and Carlsson (2002) estimate human poses by matching pre-stored 2D templates with marked ground-truth 2D joint locations. Shakhnarovich et al. (2003) use local sensitive hashing to allow efficient matching when the number of exemplars is large.

Part-based models are becoming increasingly popular in human parsing. Early work includes the cardboard people (Ju et al., 1996) and the pictorial structure (Felzenszwalb and Huttenlocher, 2005). Tree-structured models are commonly used due to its efficiency. But there are also methods that try to alleviate the limitation of tree-structured models, include common-factor models (Lan and Huttenlocher, 2005), loopy graphs (Jiang and Martin, 2008; Ren et al., 2005; Tian and Sclaroff, 2010; Tran and Forsyth, 2010), mixtures of trees (Wang and Mori, 2008).

Many part-based models use discriminative learning to train the model parameters. Examples include the conditional random fields (Ramanan and Sminchisescu, 2006;

Ramanan, 2006), max-margin learning (Kumar et al., 2009; Wang et al., 2011; Yang and Ramanan, 2011) and boosting (Andriluka et al., 2009; Sapp et al., 2010b; Singh et al., 2010). Previous approaches have also explored various features, including image segments (superpixels) (Johnson and Everingham, 2009; Mori et al., 2004; Mori, 2005; Sapp et al., 2010a,b; Srinivasan and Shi, 2007), color features (Ramanan, 2006; Ferrari et al., 2008), gradient features (Andriluka et al., 2009; Johnson and Everingham, 2010; Wang et al., 2011; Yang and Ramanan, 2011).

Human action recognition: Most of the previous work on human action recognition focuses on videos. Some work (Efros et al., 2003) uses global template for action recognition. A lot of recent work (Dollár et al., 2005; Laptev et al., 2008; Niebles et al., 2006) uses bag-of-words models. There is also work (Ke et al., 2007; Niebles and Fei-Fei, 2007) using part-based models.

Compared with videos, human action recognition from static images is a relatively less-studied area. Wang et al. (2006) provide one of the earliest examples of action recognition in static images. Recently, template models (Ikizler-Cinbis et al., 2009), bag-of-words models (Delaitre et al., 2010), part-based models (Delaitre et al., 2010; Yang et al., 2010) have all been proposed for static-image action recognition. There is also a line of work on using contexts for action recognition in static images, including human-object context (Desai et al., 2010; Gupta et al., 2009; Yao and Fei-Fei, 2010) and group context (Lan et al., 2010; Maji et al., 2011).

3. Hierarchical Poselets

Our pose representation is based on the concept of "poselet" introduced in Bourdev and Malik (2009). In a nutshell, poselets refer to pieces of human poses that are tightly clustered in both appearance and configuration spaces. Poselets have been shown to be effective at person detection (Bourdev and Malik, 2009; Bourdev et al., 2010).

In this paper, we propose a new representation called *hierarchical poselets*. Hierarchical poselets extend the original poselets in several important directions to make them more appropriate for human parsing. We start by highlighting the important properties of our representation.

Beyond rigid "parts": Most of the previous work in part-based human modeling are based on the notion that the human body can be modeled as a set of rigid parts connected in some way. Almost all of them use a natural definition of parts (e.g., torso, head, upper/lower limbs) corresponding to body segments, and model those parts as rectangles, parallel lines, or other primitive shapes.

As pointed out by Bourdev and Malik (2009), this natural definition of "parts" fails to acknowledge the fact that rigid parts are not necessarily the most salient features for visual recognition. For example, rectangles and parallel lines can be found as limbs, but they can also be easily confused with windows, buildings, and other objects in the background. So it is inherently difficult to build reliable detectors for those parts. On the other hand, certain visual patterns covering large portions of human bodies, for example, "a torso with the left arm raising up" or "legs in lateral pose", are much more visually distinctive and easier to identify. This phenomenon was observed even

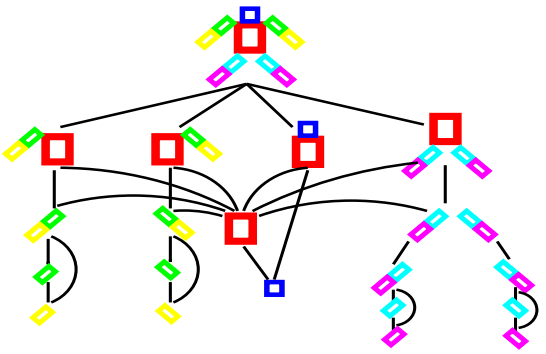


Figure 1: An illustration of the hierarchical pose representation. The black edges indicate the connectivity among different parts.

prior to the work of poselet and was exploited to detect stylized human poses and build appearance models for kinematic tracking (Ramanan et al., 2005).

Multiscale hierarchy of "parts": Another important property of our representation is that we define "parts" at different levels of hierarchy to cover pieces of human poses at various granularity, ranging from the configuration of the whole body, to small rigid parts. In particular, we define 20 parts to represent the human pose and organize them in a hierarchy shown in Figure 1. To avoid terminological confusion, we will use "part" to denote one of the 20 parts in Figure 1 and use "primitive part" to denote rigid body parts (i.e., torso, head, half limbs) from now on.

In this paper, we choose the 20 parts and the hierarchical structure in Figure 1 manually. Of course, it is possible to define parts corresponding to other combinations of body segments, for example, left part of the whole body. It may also be possible to learn the connectivity of parts automatically from data, for example, using structure learning methods similar to the Chow-Liu algorithm (Chow and Liu, 1968). We would like to leave these issues as future work.

We use a procedure similar to Yang et al. (2010) to select poselets for each part. First, we cluster the joints on each part into several clusters based on their relative x and y coordinates with respect to some reference joint of that part. For example, for the part "torso", we choose the middle-top joint as the reference and compute the relative coordinates of all the other joints on the torso with respect to this reference joint. The concatenation of all those coordinates will be the vector used for clustering. We run K-means clustering on the vectors collected from all training images and remove clusters that are too small. Similarly, we obtain the clusters for all the other parts. In the end, we obtain 5 to 20 clusters for each part. Based on the clustering, we crop the corresponding patches from the images and form a set of poselets for that part. Figure 2 shows examples of two different poselets for the part "legs".

Our focus is the new representation, so we use standard HOG descriptors (Dalal and Triggs, 2005) to keep the feature engineering to the minimum. For each poselet,

Figure 2: Examples of two poselets for the part "legs". Each row corresponds to a poselet. We show several patches from the poselet cluster. The last column shows the HOG template of the poselet.

we construct HOG features from patches in the corresponding cluster and from random negative patches. Inspired by the success of multiscale HOG features (Felzenszwalb et al., 2010), we use different cell sizes when computing HOG features for different parts. For example, we use cells of 12×12 pixel regions for poselets of the whole body, and cells of 2×2 for poselets of the upper/lower arm. This is motivated by the fact that large body parts (e.g., whole body) are typically well-represented by coarse shape information, while small body parts (e.g., half limb) are better represented by more detailed information. We then train a linear SVM classifier for detecting the presence of each poselet. The learned SVM weights can be thought as a template for the poselet. Examples of several HOG templates for the "legs" poselets are shown as the last columns of Figure 2. Examples of poselets and their corresponding HOG templates for other body parts are shown in Figure 3.

A poselet of a primitive part contains two endpoints. For example, for a poselet of upper-left leg, one endpoint corresponds to the joint between torso and upper-left leg, the other one corresponds to the joint between upper/lower left leg. We record the mean location (with respect to the center of the poselet image patch) of each endpoint. This information will be used in human parsing when we need to infer the endpoints of a primitive part for a test image.

4. Human Parsing

In this section, we describe how to use hierarchical poselets in human parsing. We first develop an undirected graphical model to represent the configuration of the human pose (Section 4.1). We then develop the inference algorithm for finding the best pose configuration in the model (Section 4.2) and the algorithm for learning model parameters (Section 4.3) from training data.

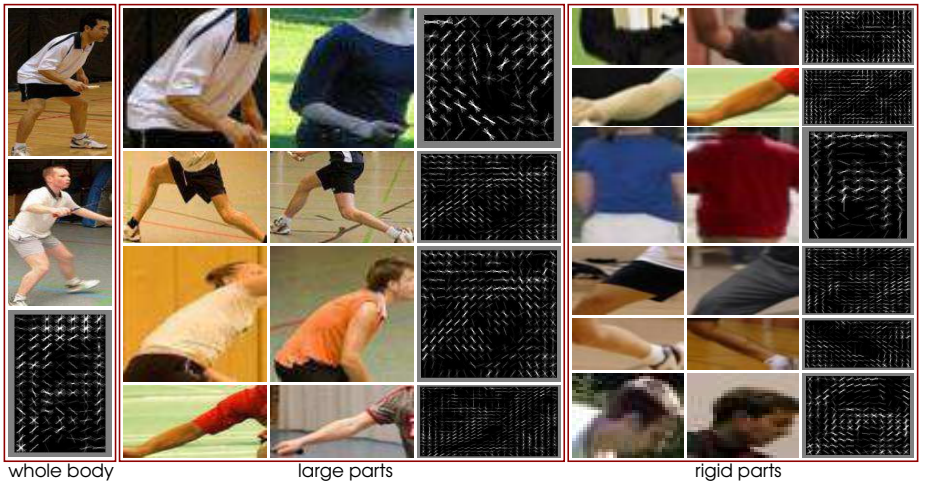


Figure 3: Visualization of some poselets learned from different body parts on the UIUC people data set, including whole body, large parts (top to bottom: torso+left arm, legs, torso+head, left arm), and rigid parts (top to bottom: upper/lower left arm, torso, upper/lower left leg, head). For each poselet, we show two image patches from the corresponding cluster and the learned SVM HOG template.

4.1. Model Formulation

We denote the complete configuration of a human pose as $L = \{l_i\}_{i=1}^{K}$, where K is the total number of parts (i.e., $K = 20$ in our case). The configuration of each part l_i is parametrized by $l_i = (x_i, y_i, z_i)$. Here (x_i, y_i) defines the image location, and z_i is the index of the corresponding poselet for this part, that is, $z_i \in \{1, 2, ..., \mathscr{P}_i\}$, where $\mathscr{P}_i$ is the number of poselets for the i-th part. In this paper, we assume the scale of the person is fixed and do not search over multiple scales. It is straightforward to augment l_i with other information, for example, scale and foreshortening.

The complete pose L can be represented by a graph $\mathscr{G} = \{\mathscr{V}, \mathscr{E}\}$, where a vertex $i \in \mathscr{V}$ denotes a part and an edge $(i, j) \in \mathscr{E}$ captures the constraint between parts i and j. The structure of $\mathscr{G}$ is shown in Figure 1. We define the score of labeling an image I with the pose L as:

$$F(L, I) = \sum_{i \in \mathscr{V}} \phi(l_i; I) + \sum_{(i,j) \in \mathscr{E}} \psi(l_i, l_j) \tag{1}$$

The details of the potential functions in Equation 1 are as follows.

Spatial prior $\psi(l_i, l_j)$: This potential function captures the compatibility of configurations of part i and part j. It is parametrized as:

$$\psi(l_i, l_j) = \alpha_{i;j;z_i;z_j}^{\top} \text{bin}(x_i - x_j, y_i - y_j)$$
$$= \sum_{a=1}^{\mathscr{P}_i} \sum_{b=1}^{\mathscr{P}_j} \mathbb{1}_a(z_i) \mathbb{1}_b(z_j) \alpha_{i;j;a;b}^{\top} \text{bin}(x_i - x_j, y_i - y_j)$$

Similar to Ramanan (2006), the function $\text{bin}(\cdot)$ is a vectorized count of spatial histogram bins. We use $\mathbb{1}_a(\cdot)$ to denote the function that takes 1 if its argument equals a, and 0 otherwise. Here $\alpha_{i;j;z_i;z_j}$ is a model parameter that favors certain relative spatial bins when poselets z_i and z_j are chosen for parts i and j, respectively. Overall, this potential function models the (relative) spatial arrangement and poselet assignment of a pair (i, j) of parts.

Local appearance $\phi(l_i; I)$: This potential function captures the compatibility of placing the poselet z_i at the location (x_i, y_i) of an image I. It is parametrized as:

$$\phi(l_i; I) = \beta_{i;z_i}^{\top} f(I(l_i)) = \sum_{a=1}^{\mathscr{P}_i} \beta_{i;a}^{\top} f(I(l_i)) \cdot \mathbb{1}_a(z_i)$$

where $\beta_{i;z_i}$ is a vector of model parameters corresponding to the poselet z_i and $f(I(l_i))$ is a feature vector corresponding to the image patch defined by l_i. We define $f(I(l_i))$ as a length $\mathscr{P}_i + 1$ vector as:

$$f(I(l_i)) = [f_1(I(l_i)), f_2(I(l_i)), ..., f_{\mathscr{P}_i}(I(l_i)), 1]$$

Each element $f_r(I(l_i))$ is the score of placing poselet z_r at image location (x_i, y_i). The constant 1 appended at the end of vector allows us to learn the model with a bias term. In other words, the score of placing the poselet z_i at image location (x_i, y_i) is a linear combination (with bias term) of the responses all the poselet templates at (x_i, y_i) for part i. We have found that this feature vector works better than the one used in Yang et al. (2010), which defines $f(I(l_i))$ as a scalar of a single poselet template response. This is because the poselet templates learned for a particular part are usually not independent of each other. So it helps to combine their responses as the local appearance model.

We summarize and highlight the important properties of our model and contextualize our research by comparing with related work.

Discriminative "parts": Our model is based on a new concept of "parts" which goes beyond the traditional rigid parts. Rigid parts are inherently difficult to detect. We instead consider parts covering a wide range of portions of human bodies. We use poselets to capture distinctive appearance patterns of various parts. These poselets have better discriminative powers than traditional rigid part detectors. For example, look at the examples in Figure 2 and Figure 3, the poselets capture various characteristic patterns for large parts, such as the "A"-shape for the legs in the first row of Figure 2.

Coarse-to-fine granularity: Different parts in our model are represented by features at varying levels of details (i.e., cell sizes in HOG descriptors). Conceptually, this multi-level granularity can be seen as providing an efficient coarse-to-fine search strategy.

However, it is very different from the coarse-to-fine cascade pruning in Sapp et al. (2010b). The method in Sapp et al. (2010b) prunes the search space of small parts (e.g., right lower arm) at the coarse level using simple features and apply more sophisticated features in the pruned search space. However, we would like to argue that at the coarse level, one should not even consider small parts, since they are inherently difficult to detect or prune at this level. Instead, we should focus on large body parts since they are easy to find at the coarse level. The configurations of large pieces of human bodies will guide the search of smaller parts. For example, an upright torso with arms raising up (coarse-level information) is a very good indicator of where the arms (fine-level details) might be.

Structured hierarchical model: A final important property of our model is that we combine information across different parts in a structured hierarchical way. The original work on poselets (Bourdev and Malik, 2009; Bourdev et al., 2010) uses a simple Hough voting scheme for person detection, that is, each poselet votes for the center of the person, and the votes are combined together. This Hough voting might be appropriate for person detection, but it is not enough for human parsing which involves highly complex and structured outputs. Instead, we develop a structured model that organize information about different parts in a hierarchical fashion. Another work that uses hierarchical models for human parsing is the AND-OR graph in Zhu et al. (2008). But there are two important differences. First, the appearance models used in Zhu et al. (2008) are only defined on sub-parts of body segments. Their hierarchical model is only used to put all the small pieces together. As mentioned earlier, appearance models based on body segments are inherently unreliable. In contrast, we use appearance models associated with parts of varying sizes. Second, the OR-nodes in Zhu et al. (2008) are conceptually similar to poselets in our case. But the OR-nodes in Zhu et al. (2008) are defined manually, while our poselets are learned.

Our work on human parsing can be seen as bridging the gap between two popular schools of approaches for human parsing: part-based methods, and exemplar-based methods. Part-based methods, as explained above, model the human body as a collection of rigid parts. They use local part appearances to search for those parts in an image, and use configuration priors to put these pieces together in some plausible way. But since the configuration priors in these methods are typically defined as pairwise constraints between parts, these methods usually lack any notion that captures what a person should look like as a whole. In contrast, exemplar-based methods (Mori and Malik, 2002; Shakhnarovich et al., 2003; Sullivan and Carlsson, 2002) search for images with similar whole body configurations, and transfer the poses of those well-matched training images to a new image. The limitation of exemplar-based approaches is that they require good matching of the entire body. They cannot handle test images of which the legs are similar to some training images, while the arms are similar to other training images. Our work combines the benefits of both schools. On one hand, we capture the large-scale information of human pose via large parts. On the other hand, we have the flexibility to compose new poses from different parts.

4.2. Inference

Given an image I, the inference problem is to find the optimal pose labeling L^* that maximize the score $F(L,I)$, that is, $L^* = \arg\max_L F(L,I)$. We use the max-product version of belief propagation to solve this problem. We pick the vertex corresponding to part "whole body" as the root and pass messages upwards towards this root. The message from part i to its parent j is computed as:

$$m_i(l_j) = \max_{l_i}(u(l_j) + \psi(l_i,l_j)) \tag{2}$$
$$u(l_j) = \phi(l_j) + \sum_{k \in \text{kids}_j} m_k(l_j)$$

Afterwards, we pass messages downward from the root to other vertices in a similar fashion. This message passing scheme is repeated several times until it converges. If we temporarily ignore the poselet indices z_i and z_j and think of $l_i = (x_i, y_i)$, we can represent the messages as 2D images and pass messages using techniques similar to those in Ramanan (2006). The image $u(l_j)$ is obtained by summing together response images from its child parts $m_k(l_j)$ and its local response image $\phi(l_j)$. $\phi(l_j)$ can be computed in linear time by convolving the HOG feature map with the template of z_j. The maximization in Equation 2 can also be calculated in time linear to the size of $u(l_j)$. In practice, we compute messages on each fixed (z_i, z_j) and enumerate all the possible assignments of (z_i, z_j) to obtain the final message. Note that since the graph structure is not a tree, this message passing scheme does not guarantee to find the globally optimal solution. But empirically, we have found this approximate inference scheme to be sufficient for our application.

The inference gives us the image locations and poselet indices of all the 20 parts (both primitive and non-primitive). To obtain the final parsing result, we need to compute the locations of the two endpoints for each primitive part. These can be obtained from the mean endpoint locations recorded for each primitive part poselet (see Sec. 3).

Figure 4 shows a graphical illustration of applying our model on a test image. For each part in the hierarchy, we show two sample patches and the SVM HOG template corresponding to the poselet chosen for that part.

4.3. Learning

In order to describe the learning algorithm, we first write Equation 1 as a linear function of a single parameter vector w which is a concatenation of all the model parameters, that is:

$$F(L,I) = w^\top \Phi(I,L), \quad \text{where}$$
$$w = [\alpha_{i;j;a;b}; \beta_{i;a}], \quad \forall i,j,a,b$$
$$\Phi(I,L) = [\mathbb{1}_a(z_i)\mathbb{1}_b(z_j)\text{bin}(x_i - x_j, y_i - y_j); f(I(l_i))\mathbb{1}_a(z_i)], \quad \forall i,j,a,b$$

The inference scheme in Section 4.2 solves $L^* = \arg\max_L w^\top \Phi(I,L)$. Given a set of training images in the form of $\{I^n, L^n\}_{n=1}^N$, we learn the model parameters w using a

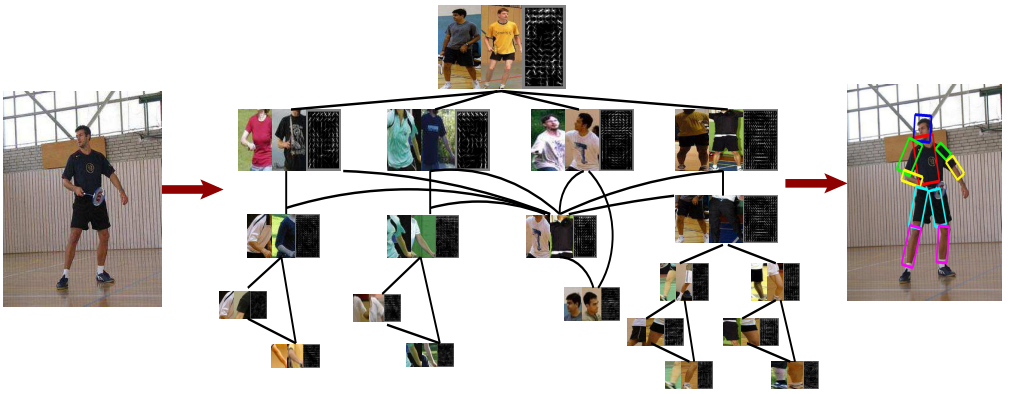


Figure 4: A graphical illustration of applying our model on a test image. For each part (please refer to Figure 1), we show the inferred poselet by visualizing two sample patches from the corresponding poselet cluster and the SVM HOG template.

form of structural SVM (Tsochantaridis et al., 2005) as follows:

$$\min_{w,\xi} \quad \frac{1}{2}||w||^2 + C\sum_n \xi^n, \quad \text{s.t. } \forall n, \ \forall L \tag{3}$$

$$w^\top \Phi(I^n, L^n) - w^\top \Phi(I^n, L) \geq \Delta(L, L^n) - \xi^n \tag{4}$$

Consider a training image I^n, the constraint in Equation 4 enforces the score of the true label L^n to be larger than the score of any other hypothesis label L by some margin. The loss function $\Delta(L, L^n)$ measures how incorrect L is compared with L^n. Similar to regular SVMs, ξ_n are slack variables used to handle soft margins. This formulation is often called margin-rescaling in the SVM-struct literature (Tsochantaridis et al., 2005).

We use a loss function that decomposes into a sum of local losses defined on each part $\Delta(L, L^n) = \sum_{i=1}^K \Delta_i(L_i, L_i^n)$. If the i-th part is a primitive part, we define the local loss $\Delta_i(L_i, L_i^n)$ as:

$$\Delta_i(L_i, L_i^n) = \lambda \cdot \mathbb{1}(z_i \neq z_i^n) + d((x_i, y_i), (x_i^n, y_i^n)) \tag{5}$$

where $\mathbb{1}(\cdot)$ is an indicator function that takes 1 if its argument is true, and 0 otherwise. The intuition of Equation 5 is as follows. If the hypothesized poselet z_i is the same as the ground-truth poselet z_i^n for the i-th part, the first term of Equation 5 will be zero. Otherwise it will incur a loss λ (we choose $\lambda = 10$ in our experiments). The second term in Equation 5, $d((x_i, y_i), (x_i^n, y_i^n))$, measures the distance (we use l_1 distance) between two image locations (x_i, y_i) and (x_i^n, y_i^n). If the hypothesized image location (x_i, y_i) is the same as the ground-truth image location (x_i^n, y_i^n) for the i-th part, no loss is added. Otherwise a loss proportional to the l_1 distance of these two locations will be incurred.

If the i-th part is not a primitive part, we simply set $\Delta(L_i, L_i^n)$ to be zero. This choice is based on the following observation. In our framework, non-primitive parts only serve as some intermediate representations that help us to search for and disambiguate small primitive parts. The final human parsing results are still obtained from configurations l_i of primitive parts. Even if a particular hypothesized L gets one of its non-primitive part labeling wrong, it should not be penalized as long as the labelings of primitive parts are correct.

The optimization problem in Equations (3,4) is convex and can be solved using the cutting plane method implemented in the SVM-struct package (Joachims et al., 2008). However we opt to use a simpler stochastic subgradient descent method to allow greater flexibility in terms of implementation.

First, it is easy to show that Equations (3,4) can be equivalently written as:

$$\min_w \frac{1}{2}||w||^2 + C\sum_n \mathscr{R}^n(L), \text{where } \mathscr{R}^n(L) =$$
$$\max_L \left(\Delta(L, L^n) + w^\top \Phi(I^n, L) - w^\top \Phi(I^n, L^n)\right)$$

In order to do gradient descent, we need to calculate the subgradient $\partial_w \mathscr{R}^n(L)$ at a particular w. Let us define:

$$L^\star = \arg\max_L \left(\Delta(L, L^n) + w^\top \Phi(I^n, L)\right) \tag{6}$$

Equation 6 is called loss-augmented inference (Joachims et al., 2008). It can be shown that the subgradient $\partial_w \mathscr{R}^n(L)$ can be computed as $\partial_w \mathscr{R}(L) = \Phi(I^n, L^\star) - \Phi(I^n, L^n)$. Since the loss function $\Delta(L, L^n)$ can be decomposed into a sum over local losses on each individual part, the loss-augmented inference in Equation 6 can be solved in a similar way to the inference problem in Section 4.2. The only difference is that the local appearance model $\phi(l_i; I)$ needs to be augmented with the local loss function $\Delta(L_i, L_i^n)$. Interested readers are referred to Joachims et al. (2008) for more details.

5. Action Recognition

The hierarchical poselet is a representation general enough to be used in many applications. In this section, we demonstrate it in human action recognition from static images.

Look at the images depicted in Figure 5. We can easily perceive the actions of people in those images, even though only static images are given. So far most work in human action recognition has been focusing on recognition from videos. While videos certainly provide useful cues (e.g., motion) for action recognition, the examples in Figure 5 clearly show that the information conveyed by static images is also an important component of action recognition. In this paper, we consider the problem of inferring human actions from static images. In particular, we are interested in exploiting the human pose as a source of information for action recognition.

Several approaches have been proposed to address the problem of static image action recognition in the literature. The first is a standard pattern classification approach,

Figure 5: Human actions in static images. We show some sample images and their annotations on the two data sets used in our experiments (see Section 6). Each image is annotated with the action category and joints on the human body. It is clear from these examples that static images convey a lot of information about human actions.

that is, learning a classifier based on certain image feature representations. For example, Ikizler-Cinbis et al. (2009) learn SVM classifiers based on HOG descriptors. The limitation with this approach is that it completely ignores the pose of a person. Another limitation is that SVM classifiers implicitly assume that images from the same action category can be represented by a canonical prototype (which are captured by the weights of the SVM classifier). However, the examples in Figure 5 clearly show that humans can have very varied appearances when performing the same action, which are hard to characterize with a canonical prototype.

Another approach to static image action recognition is to explicitly recover the human pose, then use the pose as a feature representation for action recognition. For example, Ferrari et al. (2009) estimate the 2D human pose in TV shots. The estimated 2D poses can be used to extract features which in turn can be used to retrieve TV shots containing people with similar poses to a query. As point out in Yang et al. (2010), the problem with this approach is that 2D human pose estimation is still a very challenging problem. The output of the state-of-the-art pose estimation system is typically not reliable enough to be directly used for action recognition.

The work in Yang et al. (2010) is the closest to ours. It uses a representation based on human pose for action recognition. But instead of explicitly recovering the precise pose configuration, it represents the human pose as a set of latent variables in the model. Their method does not require the predicted human pose to be exactly correct. Instead, it learns which components of the pose are useful for differentiating various actions.

The pose representation in Yang et al. (2010) is limited to four parts: upper body, left/right arm, and legs. Learning and inference in their model amounts to infer the best configurations of these four parts for a particular action. A limitation of this represen-

tation is that it does not contain pose information about larger (e.g., whole body) or smaller (e.g., half-limbs) parts. We believe that pose information useful for discerning actions can vary depending on different action categories. Some actions (e.g., *running*) have distinctive pose characteristics in terms of both the upper and lower bodies, while other actions (e.g., *pointing*) are characterized by only one arm. The challenge is how to represent the pose information at various levels of details for action recognition.

In this section, we use hierarchical poselets to capture richer pose information for action recognition. While a richer pose representation may offer more pose information (less bias), it must also be harder to estimate accurately (more variance). In this paper, we demonstrate that our rich pose representation (even with higher variance) is useful for action recognition.

5.1. Action-Specific Hierarchical Poselets

Since our goal is action recognition, we choose to use an action-specific variant of the hierarchical poselets. This is similar to the action-specific poselets used in Yang et al. (2010). The difference is that the action-specific poselets in Yang et al. (2010) are only defined in terms of four parts—left/right arms, upper-body, and legs. These four parts are organized in a star-like graphical model. In contrast, our pose representation captures a much wider range of information across various pieces of the human body. So ours is a much richer representation than Yang et al. (2010).

The training images are labeled with ground-truth action categories and joints on the human body (Figure 5). We use the following procedure to select poselets for a specific part (e.g., *legs*) of a particular action category (e.g., *running*). We first collect training images of that action category (*running*). Then we cluster the joints on the part (*legs*) into several clusters based on their relative (x, y) coordinates with respect to some reference joint. Each cluster will correspond to a "running legs" poselet. We repeat this process for the part in other action categories. In the end, we obtain about 15 to 30 clusters for each part. Figures 6 and 7 show examples of poselets for "playing golf" and "running" actions, respectively.

Similarly, we train a classifier based on HOG features (Dalal and Triggs, 2005) to detect the presence of each poselet. Image patches in the corresponding poselet cluster are used as positive examples and random patches as negative examples for training the classifier. Similar to the model in Sec. 4, we use different cell sizes when constructing HOG features for different parts. Large cell sizes are used for poselets of large body parts (e.g., whole body and torso), while small cell sizes are used for small body parts (e.g., half limbs). Figure 6 and Figure 7 show some examples of the learned SVM weights for some poselets.

5.2. Our Model

Let I be an image containing a person, $Y \in \mathscr{Y}$ be its action label where $\mathscr{Y}$ is the action label alphabet, L be the pose configuration of the person. The complete pose configuration is denoted as $L = \{l_i\}_{i=1}^{K}$ ($K = 20$ in our case), where $l_i = (x_i, y_i, z_i)$ represents the

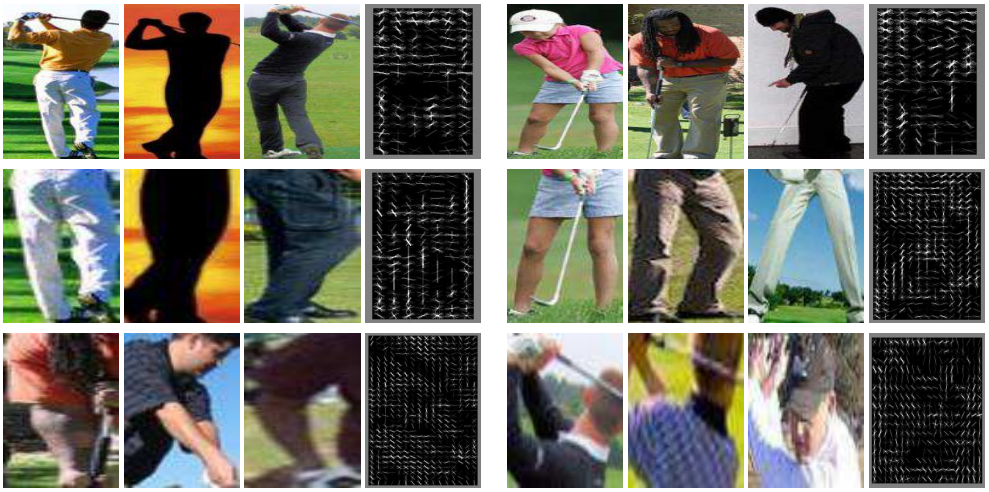


Figure 6: Examples of poselets for "playing golf". For each poselet, we visualize several patches from the corresponding cluster and the SVM HOG template. Notice the multi-scale nature of the poselets. These poselets cover various portions of the human bodies, including the whole body (1st row), both legs (2nd row), one arm (3nd row), respectively.

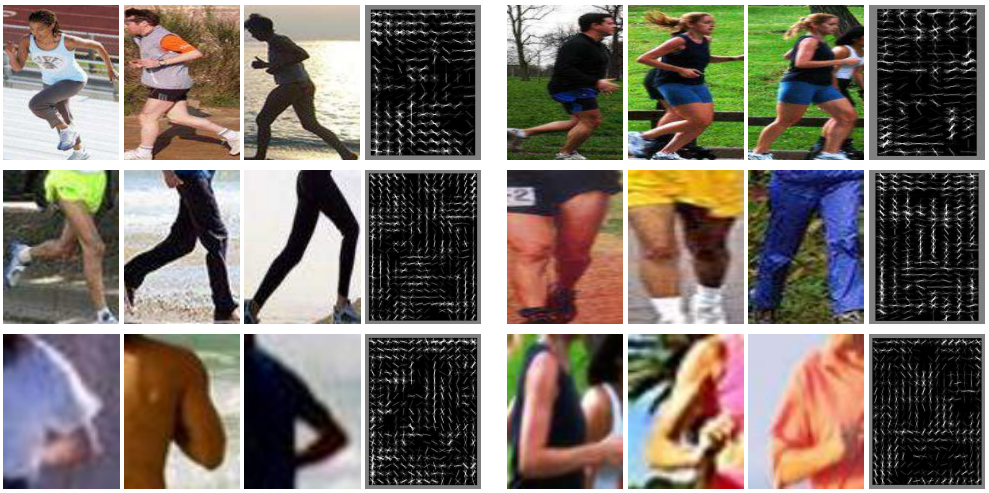


Figure 7: Examples of poselets for "running". For each poselet, we visualize several patches from the corresponding cluster and the SVM HOG template. Similar to Figure 6, these poselets cover various portions of the human bodies

2D image location and the index of the corresponding poselet cluster for the i-th part. The complete pose L can be represented by a graph $\mathscr{G} = \{\mathscr{V}, \mathscr{E}\}$ shown in Figure 1. A vertex $i \in \mathscr{V}$ denotes the i-th part and an edge $(i, j) \in \mathscr{E}$ represents the spatial constraint between the i-th and the j-th parts. We define the following scoring function to measure the compatibility of the triple (I, L, Y):

$$F(I, L, Y) = \omega_Y(I) + \sum_{i \in \mathscr{V}} \phi_Y(I, l_i) + \sum_{i,j \in \mathscr{E}} \psi_Y(l_i, l_j) \tag{7}$$

Here we use the subscript to explicitly emphasize that these functions are specific for a particular action label Y. The details of the potential functions in Equation 7 are as follows.

Root appearance $\omega_Y(I)$: This potential function models the compatibility of the action label Y and the global appearance of an image I. It is parametrized as:

$$\omega_Y(I) = \alpha_Y^\top \cdot f(I) \tag{8}$$

Here $f(I)$ is a feature vector extracted from the whole image I without considering the pose. In this paper, we use the HOG descriptor (Dalal and Triggs, 2005) of I as the feature vector $f(I)$. The parameters α_Y can be interpreted as a HOG template for the action category Y. Note that if we only consider this potential function, the parameters $\{\alpha_Y\}_{Y \in \mathscr{Y}}$ can be obtained from the weights of a multi-class linear SVM trained with HOG descriptors $f(I)$ alone without considering the pose information.

Part appearance $\phi_Y(I, l_i)$: This potential function models the compatibility of the configuration l_i of the i-th part and the local image patch defined by $l_i = (x_i, y_i, z_i)$, under the assumption that the action label is Y. Since our goal is action recognition, we also enforce that the poselet z_i should comes from the action Y. In other words, if we define $\mathscr{Z}_i^Y$ as the set of poselet indices for the i-th part corresponding to the action category Y, this potential function is parametrized as:

$$\phi_Y(I, l_i) = \begin{cases} \beta_{i,Y}^\top \cdot f(I, l_i) & \text{if } z_i \in \mathscr{Z}_i^Y \\ -\infty & \text{otherwise.} \end{cases} \tag{9}$$

Here $f(I, l_i)$ is the score of placing the SVM HOG template z_i at location (x_i, y_i) in the image I.

Pairwise part constraint $\psi(l_i, l_j)$: This potential function models the compatibility of the configurations between the i-th and the j-th parts, under the assumption that the action label is Y. We parametrize this potential function using a vectorized counts of spatial histogram bins, similar to Ramanan (2006); Yang et al. (2010). Again, we enforce poselets z_i and z_j to come from action Y as follows:

$$\psi_Y(l_i, l_j) = \begin{cases} \gamma_{i,Y}^\top \cdot \text{bin}(l_i - l_j) & \text{if } z_i \in \mathscr{Z}_i^Y, z_j \in \mathscr{Z}_j^Y \\ -\infty & \text{otherwise} \end{cases} \tag{10}$$

Here $\text{bin}(\cdot)$ is a vector all zeros with a single one for the occupied bin.

Note that if the potential functions and model parameters in Equations(7,8,9,10) do not depend on the action label Y, the part appearance $\phi(\cdot)$ and pairwise part constraint $\psi(\cdot)$ exactly recover the human parsing model in Section 4.

5.3. Learning and Inference

We define the score of labeling an image I with the action label Y as follows:

$$H(I,Y) = \max_{L} F(I,L,Y) \tag{11}$$

Given the model parameters $\Theta = \{\alpha, \beta, \gamma\}$, Equation 11 is a standard MAP inference problem in undirected graphical models. We can approximately solve it using message passing scheme similar to that in Section 4.2. The predicted action label Y^* is chosen as $Y^* = \arg\max_Y H(I,Y)$.

We adopt the latent SVM (Felzenszwalb et al., 2010) framework for learning the model parameters. First, it is easy to see that Equation 7 can be written as a linear function of model parameters as $F(I,L,Y) = \Theta^\top \Phi(I,L,Y)$, where Θ is the concatenation of all the model parameters (i.e., α, β and γ) and $\Phi(I,L,Y)$ is the concatenation of the corresponding feature vectors. Given a set of training examples in the form of $\{I^n, L^n, Y^n\}_{n=1}^N$, the model parameters are learned by solving the following optimization problem:

$$\min_{\Theta,\xi} \ \frac{1}{2}||\Theta||^2 + C\sum_n \xi^n, \quad \text{s.t. } \forall n, \ \forall Y \tag{12}$$

$$H(I^n,Y^n) - H(I^n,Y) \geq \Delta(Y,Y^n) - \xi^n \tag{13}$$

It is easy to show that Equations (12,13) can be equivalently written as:

$$\min_{\Theta} \frac{1}{2}||\Theta||^2 + C\sum_n \mathscr{R}^n, \quad \text{where} \tag{14}$$

$$\mathscr{R}^n = \max_{Y,L} \Big(\Delta(Y,Y^n) + \Theta^\top \cdot \Phi(I^n,Y)\Big) - \max_{L} \Theta^\top \cdot \Phi(I^n,L,Y^n)$$

The problem in Equation 14 is not convex, but we can use simple stochastic subgradient descent to find a local optimum. Let us define:

$$(Y^*,L^*) = \arg\max_{Y,L}(\Delta(Y,Y^n) + \Theta^\top \cdot \Phi(I^n,L,Y))$$

$$L' = \arg\max_{L}(\Theta^\top \cdot \Phi(I^n,L,Y^n))$$

Then the gradient of Equation 14 can be computed as:

$$\Theta + C\sum_n \Big(\Phi(I^n,L^*,Y^*) - \Phi(I^n,L',Y^n)\Big)$$

To initialize the parameter learning, we first learn a pose estimation model using the labeled (I^n,L^n) collected from training examples with class label Y. The parameters of these pose estimation models are used to initialize β_Y and γ_Y. The parameters α_Y are initialized from a linear SVM model based on HOG descriptors without considering the poses.

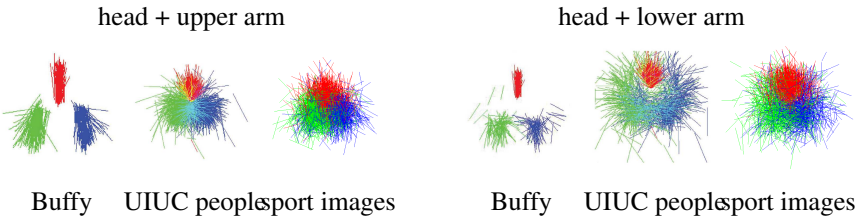


Figure 8: Scatter plots of heads (red) and upper/lower arms (blue and green) with respect to fixed upper body position on three data sets.

6. Experiments

In this section, we present our experimental results on human parsing (Section 6.1) and action recognition (Section 6.2).

6.1. Experiments on Human Parsing

There are several data sets popular in the human parsing community, for example, Buffy data set (Ferrari et al., 2008), PASCAL stickmen data set (Eichner and Ferrari, 2009). But these data sets are not suitable for us for several reasons. First of all, they only contain upper-bodies, but we are interested in full-body parsing. Second, as pointed out in Tran and Forsyth (2010), there are very few pose variations in those data sets. In fact, previous work has exploited this property of these data sets by pruning search spaces using upper-body detection and segmentation (Ferrari et al., 2008), or by building appearance model using location priors (Eichner and Ferrari, 2009). Third, the contrast of image frames of the Buffy data set is relatively low. This issue suggests that better performance can be achieved by engineering detectors to overcome the contrast difficulties. Please refer to the discussion in Tran and Forsyth (2010) for more details. In our work, we choose to use two data sets[1] containing very aggressive pose variations. The first one is the UIUC people data set introduced in Tran and Forsyth (2010). The second one is a new sport image data set we have collected from the Internet which has been used in Wang et al. (2011). Figure 8 shows scatter plots of different body parts of our data sets compared with the Buffy data set (Ferrari et al., 2008) using a visualization style similar to Tran and Forsyth (2010) . It is clear that the two data sets used in this paper have much more variations.

6.1.1. UIUC PEOPLE DATA SET

The UIUC people data set (Tran and Forsyth, 2010) contains 593 images (346 for training, 247 for testing). Most of them are images of people playing badminton. Some are images of people playing Frisbee, walking, jogging or standing. Sample images and their parsing results are shown in the first three rows of Figure 9. We compare with two

1. Both data sets can be downloaded from http://vision.cs.uiuc.edu/humanparse.

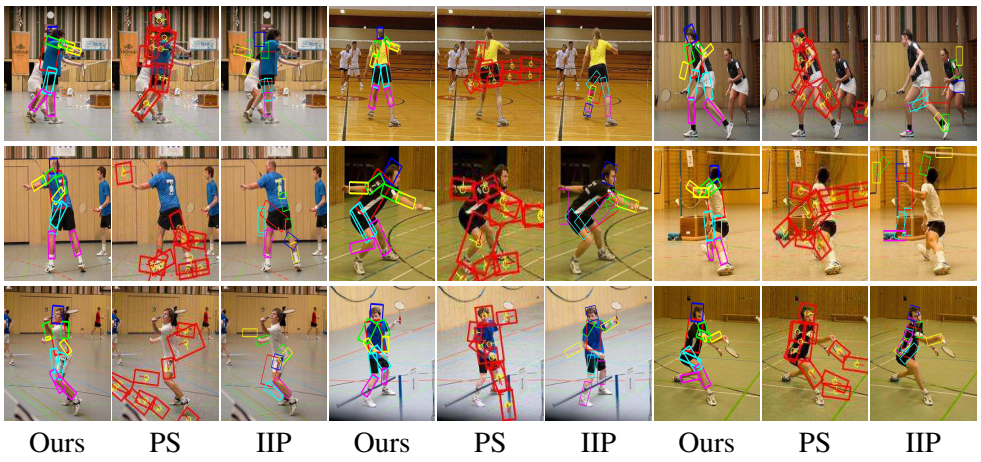


Figure 9: Examples of human body parsing on the UIUC people data set. We compare our method with the pictorial structure (PS) (Andriluka et al., 2009) and the iterative image parsing (IIP) (Ramanan, 2006). Notice the large pose variations, cluttered background, self-occlusions, and many other challenging aspects of the data set.

other state-of-the-art approaches that do full-body parsing (with published codes): the improved pictorial structure by Andriluka et al. (2009), and the iterative parsing method by Ramanan (2006). The results are also shown in Figure 9.

To quantitatively evaluate different methods, we measure the percentage of correctly localized body parts. Following the convention proposed in Ferrari et al. (2008), a body part is considered correctly localized if the endpoints of its segment lies within 50% of the ground-truth segment length from their true locations. The comparative results are shown in Table 1(a). Our method outperforms other approaches in localizing most of body parts. We also show the result (3rd row, Table 1(a)) of using only the basic-level poselets corresponding to the rigid parts. It is clear that our full model using hierarchical poselets outperforms using rigid parts alone.

Detection and parsing: An interesting aspect of our approach is that it produces not only the configurations of primitive parts, but also the configurations of other larger body parts. These pieces of information can potentially be used for applications (e.g., gesture-based HCI) that do not require precise localizations of body segments. In Figure 10, we visualize the configurations of four larger parts on some examples. Interestingly, the configuration of the whole body directly gives us a person detector. So our model can be seen as a principled way of unifying human pose estimation, person detection, and many other areas related to understanding humans. In the first row of Table 2, we show the results of person detection on the UIUC people data set by running our human parsing model, then picking the bounding box corresponding to the part "whole body" as the detection. We compare with the state-of-the-art person detectors in Felzenszwalb et al. (2010) and Andriluka et al. (2009). Since most images

Method	Torso	Upper leg		Lower leg		Upper arm		Forearm		Head
Ramanan (2006)	44.1	11.7	7.3	25.5	25.1	11.3	10.9	**25.9**	**25**	30.8
Andriluka et al. (2009)	70.9	37.3	35.6	23.1	22.7	22.3	30.0	9.7	10.5	59.1
Our method (basic-level)	79.4	53.8	53.4	47.8	39.7	17.8	21.1	11.7	16.6	65.2
Our method (full model)	**86.6**	**58.3**	**54.3**	**53.8**	**46.6**	**28.3**	**33.2**	23.1	17.4	**68.8**

(a) UIUC people data set

Method	Torso	Upper leg		Lower leg		Upper arm		Forearm		Head
Ramanan (2006)	28.7	7.4	7.2	17.6	20.8	8.3	6.6	**20.2**	**21**	12.9
Andriluka et al. (2009)	71.5	44.2	43.1	30.7	31	**28**	**29.6**	17.3	15.3	**63.3**
Our method (basic-level)	73.3	45.0	47.6	40.4	39.9	19.4	27.0	13.3	9.9	47.5
Our method (full model)	**75.3**	**50.1**	**48.2**	**42.5**	**36.5**	23.3	27.1	12.2	10.2	47.5

(b) Sport image data set

Table 1: Human parsing results by our method and two comparison methods (Ramanan, 2006; Andriluka et al., 2009) on two data sets. The percentage of correctly localized parts is shown for each primitive part. If two numbers are shown in one cell, they indicate the left/right body parts. As a comparison, we also show the results of using only rigid parts (basic-level).

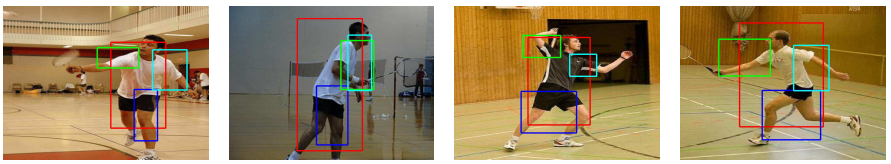


Figure 10: Examples of other information produced by our model. On each image, we show bounding boxes corresponding to the whole body, left arm, right arm and legs. The size of each bounding box is estimated from its corresponding poselet cluster.

	Our method	Felzenszwalb et al. (2010)	Andriluka et al. (2009)
UIUC people	**66.8**	48.58	50.61
Sport image	**63.94**	45.61	59.94

Table 2: Comparison of accuracies of person detection on both data sets. In our method, the configuration of the poselets corresponding to the whole body can be directly used for person detection.

contain one person, we only consider the detection with the best score on an image for all the methods. We use the metric defined in the PASCAL VOC challenge to measure the performance. A detection is considered correct if the intersection over union with respect to the ground truth bounding box is at least 50%. It is interesting to see that our method outperforms other approaches, even though it is not designed for person detection.

Figure 11: Examples of human body parsing on the sport image data set. We compare our method with the pictorial structure (PS) (Andriluka et al., 2009) and the iterative image parsing (IIP) (Ramanan, 2006).

6.1.2. SPORT IMAGE DATA SET

The UIUC people data set is attractive because it has very aggressive pose and spatial variations. But one limitation of that data set is that it mainly contains images of people playing badminton. One might ask what happens if the images are more diverse. To answer this question, we have collected a new sport image data set from more than 20 sport categories, including acrobatics, American football, croquet, cycling, hockey, figure skating, soccer, golf and horseback riding. There are in total 1299 images. We randomly choose 649 of them for training and the rest for testing. The last three rows of Figure 9 show examples of human parsing results, together with results of Andriluka et al. (2009) and Ramanan (2006) on this data set. The quantitative comparison is shown in Table 1(b). We can see that our approach outperforms the other two on the majority of body parts.

Similarly, we perform person detection using the poselet corresponding to the whole body. The results are shown in the second row of Table 2. Again, our method outperforms other approaches.

6.1.3. KINEMATIC TRACKING

To further illustrate our method, we apply the model learned from the UIUC people data set for kinematic tracking by independently parsing the human figure in each frame. In Figure 12, we show our results compared with applying the method in Ramanan (2006). It is clear from the results that kinematic tracking is still a very challenging problem. Both methods make mistakes. Interestingly, when our method makes mistakes (e.g., figures with blue arrows), the output still looks like a valid body configuration. But when the method in Ramanan (2006) makes mistakes (e.g., figures with red arrows),

Figure 12: Examples of kinematic tracking on the baseball and figure skating data sets. The 1st and 3rd rows are our results. The 2rd and 4th rows are results of Ramanan (2006). Notice how mistakes of our method (blue arrows) still look like valid human poses, while those of Ramanan (2006) (red arrows) can be wild.

the errors can be very wild. We believe this can be explained by the very different representations used in these two methods. In Ramanan (2006), a human body is represented by the set of primitive parts. Kinematic constraints are used to enforce the connectivity of those parts. But these kinematic constraints have no idea what a person looks like as a whole. In the incorrect results of Ramanan (2006), all the primitive parts are perfectly connected. The problem is their connectivity does not form a reasonable human pose as a whole.

In contrast, our model uses representations that capture a spectrum of both large and small body parts. Even in situations where the small primitive parts are hard to detect, our method can still reason about the plausible pose configuration by pulling information from large pieces of the human bodies.

6.2. Experiments on Action Recognition

We test our approach on two publicly available data sets: the still images data set (Ikizler et al., 2008) and the Leeds sport data set (Johnson and Everingham, 2010). Both data sets contain images of people with ground-truth pose annotations and action labels.

method	overall	avg per-class
Our approach	**65.15**	**70.77**
Yang et al. (2010)*	63.49	68.37
SVM mixtures	62.8	64.05
Linear SVM	60.32	61.5

Table 3: Performance on the still image data set. We report both overall and average per-class accuracies. *The results are based on our own implementation.

6.2.1. STILL IMAGE DATA SET

We first demonstrate our model on the still image data set collected in Ikizler et al. (2008). This data set contains more than 2000 static images from five action categories: dancing, playing golf, running, sitting, and walking. Sample images are shown in the first two rows of Figure 5. Yang et al. (2010) have annotated the pose with 14 joints on the human body on all the images in the data set. Following Yang et al. (2010), we choose $1/3$ of the images from each category to form the training data, and the remaining ones as the test data.[2]

We compare our approach with two baseline method. The first baseline is a multi-class SVM based on HOG features. For the second baseline, we use mixtures of SVM models similar to that in Felzenszwalb et al. (2010). We set the number of mixtures for each class to be the number of whole-body poselets. From Table 3, we can see that our approach outperforms the baseline by a large margin. Our performance is also better than the reported results in Yang et al. (2010). However, the accuracy numbers are not directly comparable since the training/testing data sets and features are not completely identical. In order to do a fair comparison, we re-implemented the method in Yang et al. (2010) by only keeping the parts used in Yang et al. (2010). Our full model performs better.

In Figure 13, we visualize several inferred poselets on some examples whose action categories are correctly classified. Each poselet is visualized by showing several patches from the corresponding poselet cluster.

6.2.2. LEEDS SPORT DATA SET

The Leeds sport data set (Johnson and Everingham, 2010) contains 2000 images from eight different sports: athletics, badminton, baseball, gymnastics, parkour, soccer, tennis, volleyball. Each image in the data set is labeled with 14 joints on the human body. Sample images and the labeled joints are shown in the last four rows of Figure 5. This data set is very challenging due to very aggressive pose variations.

We choose half of the images for training, and the other half for testing. The performance is shown in Table 4. Again, we compare with the HOG-based SVM and SVM

2. A small number of images/annotations we obtained from the authors of Yang et al. (2010) are somehow corrupted due to some file-system failure. We have removed those images from the data set.

Figure 13: Visualization of some inferred poselets on the still image data set. These test images have been correctly recognized by our model. For a test image, we show three poselets that have high responses. Each poselet is visualized by showing several patches from its cluster.

Figure 14: Visualization of some inferred poselets on the Leeds sport data set. These test images have been correctly recognized by our model. For a test image, we show three poselets that have high responses. Each poselet is visualized by showing several patches from its cluster.

method	overall	avg per-class
Our approach	**54.6**	**54.6**
SVM mixtures	52.7	49.13
Linear SVM	52.7	52.93

Table 4: Performance on the Leeds sport data set. We report both overall and average per-class accuracies.

American football→dancing croquet→playing golf field hockey→running

Figure 15: Visualization of inferred poses on unseen actions. Here the actions of the test images (*American football*, *croquet* and *field hockey*) are not available during training. Our model recognizes these examples as *dancing*, *playing golf*, *running*, respectively. Some of the results (e.g., *croquet*→ *golfing*) make intuitive sense. Others (e.g., *football*→*dancing*) might not be intuitive at first. But if we examine the poselets carefully, we can see that various pieces of the football player are very similar to those found in the dancing action.

mixtures as the baselines. We can see that our method still outperforms the baseline. Similarly, we visualize the inferred poselets on some examples in Figure 14.

6.2.3. UNSEEN ACTIONS

An interesting aspect of our model is that it outputs not only the predicted action label, but also some rich intermediate representation (i.e., action-specific hierarchical poselets) about the human pose. This information can potentially be exploited in various contexts. As an example, we apply the model learned from the still image data set to *describe* images from sports categories not available during training. In Figure 15, we show examples of applying the model learned from the still image data set to images with unseen action categories. The action categories (*American football*, *croquet* and *field hockey*) for the examples in Figure 15 are disjoint from the action categories of the still image data set. In this situation, our model obviously cannot correctly predict the action labels (since they are not available during training). Instead, it classifies those images using the action labels it has learned. For example, it classifies "American foot-

ball" as "dancing", "croquet" as "playing golf", "field hockey" as "running". More importantly, our model outputs poselets for various parts which support its prediction. From these information, we can say a lot about "American football" even though the predicted action label is wrong. For example, we can say it is closer to "dancing" than "playing golf" because the pose of the football player in the image is similar to certain type of dancing legs, and certain type of dancing arms.

7. Conclusion and Future Work

We have presented hierarchical poselets, a new representation for modeling human poses. Different poselets in our representation capture human poses at various levels of granularity. Some poselets correspond to the rigid parts typically used in previous work. Others can correspond to large pieces of the human bodies. Poselets corresponding to different parts are organized in a structured hierarchical model. The advantage of this representation is that it infers the human pose by pulling information across various levels of details, ranging from the coarse shape of the whole body, to the fine-detailed information of small rigid parts. We have demonstrate the applications of this representation in human parsing and human action recognition from static images. Recently, similar ideas (Sun and Savarese, 2011) have been applied in other applications, such as object detection.

As future work, we would like to explore how to automatically construct the parts and the hierarchy using data-driven methods. This will be important in order to extend hierarchical poselets to other objects (e.g., birds) that do not have obvious kinematic structures. We also like to apply the hierarchical poselet representation to other vision tasks, such as segmentation.

Acknowledgments

This work was supported in part by NSF under IIS-0803603 and IIS-1029035, and by ONR under N00014-01-1-0890 and N00014-10-1-0934 as part of the MURI program. Yang Wang was also supported in part by an NSERC postdoc fellowship when the work was done. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of NSF, ONR, or NSERC.

References

Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

Lubomir Bourdev and Jitendra Malik. Poselets: Body part detectors training using 3d human pose annotations. In *IEEE International Conference on Computer Vision*, 2009.

Lubomir Bourdev, Subhransu Maji, Thomas Brox, and Jitendra Malik. Detecting people using mutually consistent poselet activations. In *European Conference on Computer Vision*, 2010.

C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

Navneet Dalal and Bill Triggs. Histogram of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

Vincent Delaitre, Ivan Laptev, and Josef Sivic. Recognizing human actions in still images: a study of bag-of-features and part-based representations. In *British Machine Vision Conference*, 2010.

Chaitanya Desai, Deva Ramanan, and Charless Fowlkes. Discriminative models for static human-object interactions. In *Workshop on Structured Models in Computer Vision*, 2010.

Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *ICCV'05 Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005.

Alexei A. Efros, Alexander C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *IEEE International Conference on Computer Vision*, pages 726–733, 2003.

Marcin Eichner and Vittorio Ferrari. Better appearance models for pictorial structures. In *British Machine Vision Conference*, 2009.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, January 2005.

Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1672–1645, 2010.

Vittorio Ferrari, Manuel Marín-Jiménez, and Andrew Zisserman. Progressive search space reduction for human pose estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

Vittorio Ferrari, Manuel Marín-Jiménez, and Andrew Zisserman. Pose search: retrieving people using their pose. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

David A. Forsyth, Okan Arikan, Leslie Ikemoto, James O'Brien, and Deva Ramanan. Computational studies of human motion: Part 1, tracking and motion synthesis. *Foundations and Trends in Computer Graphics and Vision*, 1(2/3):77–254, July 2006.

Abhinav Gupta, Aniruddha Kembhavi, and Larry S. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1775–1789, 2009.

Nazli Ikizler, R. Gokberk Cinbis, Selen Pehlivan, and Pinar Duygulu. Recognizing actions from still images. In *International Conference on Pattern Recognition*, 2008.

Nazli Ikizler-Cinbis, R. Gokberk Cinbis, and Stan Sclaroff. Learning actions from the web. In *IEEE International Conference on Computer Vision*, 2009.

Hao Jiang and David R. Martin. Globel pose estimation using non-tree models. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

Thorsten Joachims, Thomas Finley, and Chun-Nam Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 2008.

Sam Johnson and Mark Everingham. Combining discriminative appearance and segmentation cues for articulated human pose estimation. In *International Workshop on Machine Learning for Vision-based Motion Analysis*, 2009.

Sam Johnson and Mark Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *British Machine Vision Conference*, 2010.

Shanon X. Ju, Michael J. Black, and Yaser Yaccob. Cardboard people: A parameterized model of articulated image motion. In *International Conference on Automatic Face and Gesture Recognition*, pages 38–44, 1996.

Yan Ke, Rahul Sukthankar, and Martial Hebert. Event detection in crowded videos. In *IEEE International Conference on Computer Vision*, 2007.

M. Pawan Kumar, Andrew Zisserman, and Philip H. S. Torr. Efficient discriminative learning of parts-based models. In *IEEE International Conference on Computer Vision*, 2009.

Tian Lan, Yang Wang, Weilong Yang, and Greg Mori. Beyond actions: Discriminative models for contextual group activities. In *Advances in Neural Information Processing Systems*. MIT Press, 2010.

Xiangyang Lan and Daniel P. Huttenlocher. Beyond trees: Common-factor models for 2d human pose recovery. In *IEEE International Conference on Computer Vision*, volume 1, pages 470–477, 2005.

Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

Subhransu Maji, Lubomir Bourdev, and Jitendra Malik. Action recognition from a distributed representation of pose and appearance. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2011.

David Marr. *A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman, 1982.

Greg Mori. Guiding model search using segmentation. In *IEEE International Conference on Computer Vision*, volume 2, pages 1417–1423, 2005.

Greg Mori and Jitendra Malik. Estimating human body configurations using shape context matching. In *European Conference on Computer Vision*, volume 3, pages 666–680, 2002.

Greg Mori, Xiaofeng Ren, Alyosha Efros, and Jitendra Malik. Recovering human body configuration: Combining segmentation and recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 326–333, 2004.

Juan Carlos Niebles and Li Fei-Fei. A hierarchical model of shape and appearance for human action classification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

Juan Carlos Niebles, Hongcheng Wang, and Li Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. In *British Machine Vision Conference*, volume 3, pages 1249–1258, 2006.

Deva Ramanan. Learning to parse images of articulated bodies. In *Advances in Neural Information Processing Systems*, volume 19, pages 1129–1136, 2006.

Deva Ramanan and Cristian Sminchisescu. Training deformable models for localization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 206–213, 2006.

Deva Ramanan, David A. Forsyth, and Andrew Zisserman. Strike a pose: Tracking people by finding stylized poses. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 271–278, 2005.

Xiaofeng Ren, Alexander Berg, and Jitendra Malik. Recovering human body configurations using pairwise constraints between parts. In *IEEE International Conference on Computer Vision*, volume 1, pages 824–831, 2005.

Benjamin Sapp, Chris Jordan, and Ben Taskar. Adaptive pose priors for pictorial structures. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010a.

Benjamin Sapp, Alexander Toshev, and Ben Taskar. Cascaded models for articulated pose estimation. In *European Conference on Computer Vision*, 2010b.

G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter sensitive hashing. In *IEEE International Conference on Computer Vision*, volume 2, pages 750–757, 2003.

Leonid Sigal and Michael J. Black. Measure locally, reason globally: Occlusion-sensitive articulated pose estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2041–2048, 2006.

Vivek Kumar Singh, Ram Nevatia, and Chang Huang. Efficient inference with multiple heterogenous part detectors for human pose estimation. In *European Conference on Computer Vision*, 2010.

Praveen Srinivasan and Jianbo Shi. Bottom-up recognition and parsing of the human body. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

Josephine Sullivan and Stefan Carlsson. Recognizing and tracking human action. In *European Conference on Computer Vision LNCS 2352*, volume 1, pages 629–644, 2002.

Min Sun and Silvio Savarese. Articulated part-base model for joint object detection and pose estimation. In *IEEE International Conference on Computer Vision*, 2011.

Tai-Peng Tian and Stan Sclaroff. Fast globally optimal 2d human detection with loopy graph models. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

K. Toyama and A. Blake. Probabilistic exemplar-based tracking in a metric space. In *IEEE International Conference on Computer Vision*, volume 2, pages 50–57, 2001.

Duan Tran and David Forsyth. Improved human parsing with a full relational model. In *European Conference on Computer Vision*, 2010.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

Yang Wang and Greg Mori. Multiple tree models for occlusion and spatial constraints in human pose estimation. In *European Conference on Computer Vision*, 2008.

Yang Wang, Hao Jiang, Mark S. Drew, Ze-Nian Li, and Greg Mori. Unsupervised discovery of action classes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

Yang Wang, Duan Tran, and Zicheng Liao. Learning hierarchical poselets for human parsing. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2011.

Weilong Yang, Yang Wang, and Greg Mori. Recognizing human actions from still images with latent poses. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

Yi Yang and Deva Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2011.

Bangpeng Yao and Li Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

Long Zhu, Yuanhao Chen, Yifei Lu, Chenxi Lin, and Alan Yuille. Max margin AND/OR graph learning for parsing the human body. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.