Tracking, Analysis, and Recognition of Human Gestures in Video

Stan Sclaroff, Margrit Betke, George Kollios,
Jonathan Alon, Vassilis Athitsos, Rui Li, John Magee, Tai-peng Tian
Dept. of Computer Science, Boston University, Boston, MA, 02215, U.S.A

Abstract

An overview of research in automated gesture spotting, tracking and recognition by the Image and Video Computing Group at Boston University is given. Approaches for localization and tracking human hands in video, estimation of hand shape and upper body pose, tracking head and facial motion, as well as efficient spotting and recognition of specific gestures in video streams are summarized. Methods for efficient dimensionality reduction of gesture time series, boosting of classifiers for nearest neighbor search in pose space, and model-based pruning of gesture alignment hypotheses are described. Algorithms are demonstrated in three domains: American Sign Language, hand signals like those employed by flight-directors on airport runways, and gesture-based interfaces for severely disabled users. The methods described are general and can be applied in other domains that require efficient detection and analysis of patterns in time-series, images or video.

1 Introduction

Recognition of human gestural communication in video involves analysis of spatiotemporal patterns and time series. While some of the first reported progress in recognizing hand gestures was inspired by work with Hidden Markov Models for online handwriting recognition, human gestural communication presents additional challenges. For instance, the gestures can be produced in three-dimensional space, but measurements are obtained from the 2D image projections of gestures as observed in video images. Moreover, gestural communication can involve a combination of simultaneous channels: facial and head motion, hand motion, hand shape, and upper body motion; thus, algorithms for analysis each of these channels are important.

The following is an overview of research in automated gesture spotting, tracking and recognition by the Image and Video Computing Group at Boston University. Algorithms are demonstrated in three application domains: American Sign Language, hand signals like those employed by flight-directors on airport runways, and gesture-based interfaces

for disabled users. The methods described are general and applicable in other domains that require efficient detection and analysis of patterns in time-series, images, or video.

2 Manual Gesture

Most hand gesture recognition systems assume the gesturing hand is reliably located in every video frame. However, reliable hand location cannot be assumed in practice. Color-based detection can yield multiple hand candidates, and the top candidate is often incorrect. Motion-based detection may fail to uniquely locate the hand if the face, non-gesturing hand or other objects are moving.

Fortunately, these detection methods can be used to produce a relatively short list of candidate hand locations, and then instead of assuming perfect hand detection, we can make the milder assumption that a list of candidate hand locations is available for each video frame. The Dynamic Space-Time Warping (DSTW) algorithm [2] is an extension of DTW that embodies this philosophy. In the DSTW algorithm, a pair of gestures is aligned in time, and simultaneously the best hand location out of the multiple hypotheses available at each frame is identified.

2.1 Dynamic Space Time Warping (DSTW)

Let $M = (M_1, \ldots, M_m)$ be a model sequence in which each M_i is a feature vector. Let $Q = (Q_1, \ldots, Q_n)$ be a query sequence. In the DTW framework, each Q_j would be a feature vector, of the same form as each M_i. However, in DSTW, we have multiple candidate feature vectors in each frame of the query. For example, the feature vector could consist of the hand's position and velocity in each frame. Each hypothesis defines a different feature vector; therefore, in our algorithm, Q_j is a *set* of feature vectors: $Q_j = \{Q_{j1}, \ldots, Q_{jK}\}$, where each Q_{jk}, $k \in \{1, \ldots, K\}$, is a candidate feature vector. Here we assume that K, the number of feature vectors extracted from each query frame is fixed, but in principle K may vary from frame to frame.

A warping path W defines an alignment between M and Q. Formally, $W = w_1, \ldots, w_T$, where $\max(m, n) \leq T \leq m + n - 1$. Each $w_t = (i, j, k)$ is a triple, which specifies

that feature vector M_i of the model is matched with feature vector Q_{jk}. We say that w_t has two *temporal* dimensions (denoted by i and j) and one *spatial* dimension (denoted by k). The warping path is subject to several constraints: (1) *boundary conditions*, namely that the warping path must match the first frame of the model with the first frame of the query, and match the last frame of the model with the last frame of the query, (2) *temporal continuity,* which restricts the allowable steps in the warping path to adjacent cells along the two temporal dimensions, and (3) *temporal monotonicity,* which forces the warping path sequence to increase monotonically in the two temporal dimensions.
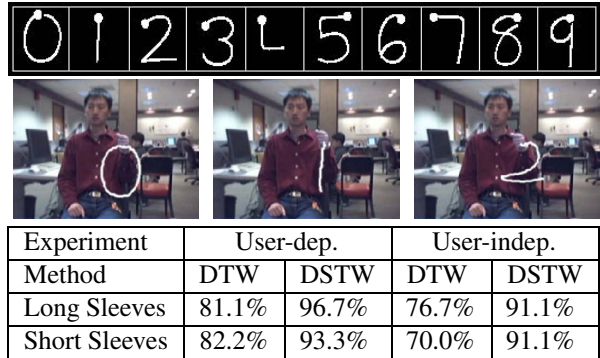
Given warping path element $w_t = (i, j, k)$, we define $N(i, j, k)$ to be the set of all possible values of w_{t-1} that satisfy these warping path constraints: $N(i, j, k) = \{(i-1, j), (i, j-1), (i-1, j-1)\} \times \{1, \ldots, K\}$. We assume that we have a cost measure $d(i, j, k) \equiv d(M_i, Q_{jk})$ between two feature vectors M_i and Q_{jk}. We also assume that we have a transition cost $\tau(w_{t-1}, w_t)$ between two successive warping path elements. DSTW finds the optimal path and the matching score as described in [2].

The algorithm has been tested in a system that recognizes users gesturing the ten digits in the style of Palm's Graffiti Alphabet (Fig. 1). Video was captured with a consumer-grade USB camera at image resolution 240×320. There were three test subjects, and a total of 270 digit exemplars were extracted from three different types of video clips depending on what the users wore: (1) *colored gloves* so the gesturing hand was uniquely detected and thus these clips could be used as gesture models, and (2) *long sleeves* and (3) *short sleeves*, both for use as testing recognition. A comparison between DTW and DSTW performance is shown in the table of Fig. 1. DSTW yields a $11.1\% - 21.1\%$ increase in classification accuracy over the simple DTW algorithm which only considers a single (best) candidate per frame.

2.2 Efficient Gesture Spotting

Gestures of interest typically appear in a continuous stream of motion. Gesture spotting is the challenging task of locating the start and end frames of the video stream that correspond to a gesture of interest, while at the same time rejecting non-gesture motion patterns.

A common algorithm for spotting is continuous dynamic programming (CDP). For every gesture model CDP maintains a table of cells, where each cell corresponds to an input frame and a model frame pair. At run time CDP visits every cell and computes the best alignment between the input video subsequence and the part of the model that correspond to that cell. We note, however, that not every cell has to be visited during the process: unlikely observations can be rejected and candidate alignment paths passing through those unlikely observations can be pruned from further consideration. In [1], pruning is determined by a set of model-

Experiment	User-dep.		User-indep.	
Method	DTW	DSTW	DTW	DSTW
Long Sleeves	81.1%	96.7%	76.7%	91.1%
Short Sleeves	82.2%	93.3%	70.0%	91.1%

Figure 1. Palm's Graffiti digits, three model digits obtained using a colored glove, and classification accuracy of DTW *vs.* the proposed algorithm, DSTW.

Method	CDP	CDPP	CDPPS
Detection Rate	78.3%	85.0%	96.7%
False Matches	13	9	2

Table 1. Gesture spotting accuracy: baseline (CDP), pruning (CDPP), and pruning with subgesture reasoning (CDPPS). Total number of gestures is 60.

dependent classifiers, that are learned from gesture training examples. Each threshold $\tau(i)$ defines a decision stump or a binary classifier for every model frame i.

Confusion may occur if multiple models are matched to the same subsequence. This can happen when some gestures may appear as subgestures of others. For example in the Palm Graffiti digit gestures, '5' may be falsely detected instead of the digit '8' because '5' is similar to a subgesture of the digit '8'. Other examples include '0' as part of a '9', or '7' as part of '2', '3', or '9', etc. A subgesture reasoning process can be added that handles the fact that some gesture models can falsely match parts of other longer gestures [1].

The gesture spotting algorithm has been tested on video clips of two users gesturing the Palm Graffiti digits 0-9 in sequence, three times each. The models are collected separately, as isolated utterances, using colored gloves. Fig. 1 summarizes the results. The proposed CDP with pruning is an order of magnitude faster than original CDP algorithm, and recognition accuracy improves by 7% because many hypotheses that could have led to false matches are eliminated at an early stage. Subgesture reasoning improves recognition accuracy by an additional 12%.

3 Hand Shape

Accurate estimation of 3D hand shape is particularly challenging due to the number of degrees of freedom in the hand, image clutter, and errors in segmentation/localization of the hand in each video frame.

In [4], a system is proposed that can provide estimates of 3D hand pose from a cluttered image. Pose estimation is formulated as an image database indexing problem. The closest matches for an input image are retrieved from a large database of synthetic hand images. Ground truth labels of the retrieved matches are used as hand pose estimates for the input. Fig. 2 shows some hand shapes used in the database, as well as example input images and poses retrieved.

A simple solution to this retrieval problem is brute-force search: compare the input to all database objects. However, many useful distance measures are computationally-expensive, e.g., the chamfer distance used in our hand image retrieval application. Spatial indexing methods that improve the computational complexity of search are inappropriate, since they typically rely on Euclidean or metric properties, and cannot be applied to arbitrary non-metric spaces. An alternative approach is to utilize an embedding, as proposed in the BoostMap framework [3].

3.1 Defining Embeddings via BoostMap

Let X be a set of objects, and $D_X(x_1, x_2)$ be a distance measure between objects $x_1, x_2 \in X$. D_X that can be metric or non-metric. A Euclidean embedding $F : X \to \mathbb{R}^d$ is a function that maps objects from X into the d-dimensional Euclidean space $\mathbb{R}^d$, where distances are typically measured using an L_p or weighted L_p measure, denoted as $D_{\mathbb{R}^d}$.

A 1D Euclidean embedding of space X is simply a function $F : X \to \mathbb{R}$. For instance, given an object $r \in X$, a simple 1D Euclidean embedding F^r can be defined $F^r(x) = D_X(x, r)$. The object r that is used to define F^r is typically called a *reference object* or a *vantage object*. If objects $a, b \in X$ are very similar to each other, we expect that their distances to r, i.e. $D_X(a, r)$ and $D_X(b, r)$ will also be very similar. Therefore, F^r tends to map nearby objects to nearby points on the line.

We want the embedding to preserve as much of the nearest neighbor structure as possible. Ideally, we want an embedding F that has the following property: for every query q, and every integer $k > 0$, if a is the k-nearest neighbor of q in the database, then $F(a)$ should be the k-nearest neighbor of $F(q)$ among the embeddings of all database objects. In other words, for any query q and any objects a and b in the database, we want $F(q)$ to be closer to $F(a)$ than to $F(b)$ if and only if q is closer to a than to b.

Finding such a perfect embedding is usually impossible. However, what we can try to do is construct an embedding in a way that maximizes the fraction of triples (q, a, b) for which the embedding indeed preserves the relative ranking of a and b with respect to q. To construct such an embedding, it is useful to think of embeddings as binary classifiers: given three objects (q, a, b), decide if q is closer to a or to b. Every embedding F defines a classifier $\tilde{F}$, which decides if q is closer to a or to b by simply checking if $F(a)$
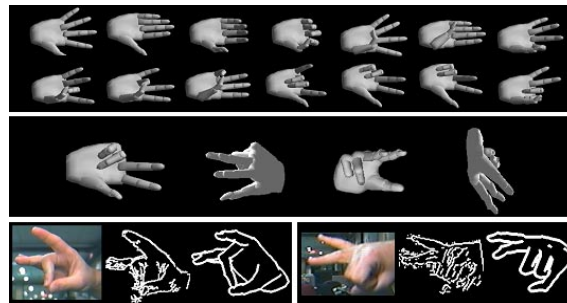
Figure 2. Top: some hand shapes in database. Middle: four of the 4128 3D hand shape orientations. Bottom: two test images, extracted edge images, matches retrieved from the database.

is closer to $F(b)$ or to $F(c)$. Overall, we want the classification error of $\tilde{F}$ to be as low as possible.

Simple 1D embeddings, like the one defined above, are expected to behave as *weak classifiers*, i.e., classifiers that may have a high error rate, but at least give answers that are better than random guesses. The problem of combining many 1D embeddings into a high-dimensional embedding then becomes equivalent to the problem of combining many weak classifiers into a strong classifier, i.e., a classifier with a low error rate. As described in [3], AdaBoost can be used in constructing this embedding.

BoostMap has been tested in various applications, including hand pose estimation. The database for hand pose experiments contains 107,328 images, generated via computer graphics (26 hand shapes, 4128 3D orientations). 703 real images of hands are used in testing. Similarity is evaluated via the symmetric chamfer distance applied to edge images. A *filter-and-refine* strategy is employed: a small number p of top matches is first retrieved using the approximate distance, and then the expensive measure is used to re-rank the p retrieved items. The experiments indicate that the BoostMap approach can exactly match brute-force accuracy but is an order of magnitude faster. Two orders of magnitude speed-up is achieved with 95% accuracy.

3.2 ASL Gesture Retrieval

The BoostMap framework has also been tested on a database of 880 gray-scale American Sign Language (ASL) sequences. Each video sequence depicts a sign, as signed by one of three native ASL signers (Fig. 4). The test queries are 180 video sequences of ASL signs, signed by a signer not included in the database. Similarity between sequences is measured in terms of optical flow, and then DTW to compute the optimal time alignment and the overall matching cost between video sequences. Evaluating the exact distance between the query and the entire database takes about 6 min., whereas BoostMap takes approximately 1.5 min. at

Figure 3. Example frames from upper-body gesture sequence. Tracking is correct even through self-occlusions: right hand is raised, occludes face, then crosses the body, and the tracker does not diverge.

Figure 4. Four frames from ASL vide database.

95% accuracy, and 2.5 min. for 100%. Assuming the number of embedding dimensions remains approximately fixed, greater improvement is expected for larger databases.

4 Upper Body Gesture

Gestures of the upper body are often used in the signals of referees at sporting events, flight directors on the airport tarmac, police directing vehicular traffic, etc. Since the lexicon of gestures is finite, the range of valid, observable body poses should also be limited.

A machine learning framework in [6] exploits the observation that such motion is intrinsically low-dimensional. The mapping to this low-dimensional space is formulated in terms of a Gaussian Process Latent Variable Model (GPLVM). Given training poses $\{\mathbf{y}_i\}$ as inputs, the GPLVM is used to define a smooth continuous low-dimensional representation of the original data, which is called *latent space*. It is spanned by the values of latent space variables $\mathbf{x}_i$, which comprise the lower dimensional representation of corresponding $\mathbf{y}_i$. Unlike some other nonlinear dimensionality techniques (e.g., ISOMAP), the GPLVM provides mappings μ and μ^{-1} between the manifold of feasible 2D body poses and the lower-dimensional (latent) space.

The GPLVM learning takes place once, off-line. Kernel parameters (α and γ) and latent variables $\mathbf{x}_i$'s are estimated. Given a set of training data $\{\mathbf{y}_i\}$, each $\mathbf{y}_i$ is a M-dimension vector. For instance, upper body training poses $\mathbf{y}_i$ can be defined in terms of the positions of key points on the body

as they would appear projected in a 2D image, e.g., elbow, shoulder, eyes, etc. In our experiments there are 12 such key points, and therefore $M = 24$. We then maximize the posterior $p(\{\mathbf{x}_i\}, \alpha, \gamma | \{\mathbf{y}_i\})$, with respect to the α, γ and latent variables $\mathbf{x}_i'$s. More details are given in [6].

Once the relationship between $\mathbf{y}$ and $\mathbf{x}$ is obtained via GPLVM learning, the lower-dimensional latent space can be utilized in pose estimation and tracking algorithms [6]. Given features $\mathbf{z}$ extracted from an input image, the goal is to find the corresponding optimal $\mathbf{y}$ and $\mathbf{x}$. For instance, given $\mathbf{z}$, proposals of likely poses can be generated in the lower-dimensional space, where generally fewer points will be required to adequately sample the space of poses. The original space configuration (24-D body pose vector) is only used when evaluating the observation likelihood $p(\mathbf{z}|\mathbf{y} = \mu(\mathbf{x}))$, where the function $\mu(.)$ maps the low-dimensional representation back into the original configuration space.

This approach has been tested in a system that tracks upper body gestures taken from a lexicon of aircraft signals. 5000 training samples are generated via computer graphics. Latent space dimensions $d = 2, 3, 4$ have been tested, and it has been found that $d = 2$ is sufficient for this experiment, A modified CONDENSATION tracker (particle filtering method) is used to track body poses through time. Example frames from a test sequence are shown in Fig. 3. The GPLVM tracks well at speeds approaching video rate.

5 Head and Facial Gesture

People who are severely paralyzed and nonverbal from cerebral palsy, stroke, multiple sclerosis, amyotrophic lateral sclerosis, or brain injury often depend on the computer to facilitate or enrich communication with friends, family, and care givers. Alternative pointing devices allow users who have a limited range of voluntary motions, to control the computer with, for example, head or tongue movements.

The CAMERA MOUSE system [5] tracks a computer user's movements with a video camera and translates them into the movements of the mouse pointer on the screen. Body features such as the tip of the user's nose can be tracked. After the feature to track is initialized, the the normalized correlation coefficient is used to search for the best location of the small template in a local window. Mouse clicks can be generated by dwelling the pointer in the same

place for a short period of time. This system has been successfully used by many people with severe disabilities, and is available to schools and individuals throughout the world.

5.1 EyeKeys

There are some users whose physical conditions are even more limiting – they may only be able to move their eyes for communication. A mouse-substitution system EYE-KEYS [7] has been developed for these people. The system runs on a consumer-grade PC with video input from an inexpensive USB camera and works without special lighting.

The system detects and tracks the person's face using multi-scale template correlation. Symmetry between left and right eyes is exploited to detect if the person is looking at the camera, or to the left or right side. A motion mask is created to isolate eye pixels and exclude other pixels whenever eye motion occurs. To determine the direction of the eyes, the signed differences are projected onto the x-axis. The signed difference creates peaks in the projection because eye sclera pixels are lighter than pupil pixels. If the user is looking left, the signed difference operation creates a peak followed by a valley in the projection. A valley followed by a peak indicates that the user is looking right. Let I_ℓ and I_r be the $m \times n$ left and right eye images masked by motion information. The projection of the signed difference onto vector $\mathbf{a} = a_1, \ldots, a_m$ is computed by: $a_i = \sum_{j=1}^{n} (I_r(i, j) - I_\ell(m - i, j))$. This projection is then compared to thresholds to determine if the peak and valley points exceed certain values.

Eye direction can be used to control applications such as spelling programs, games, or web browsers. In one experiment, eight able-bodied people tested the system to determine how often it could detect an intentional look to the left or right. The system correctly detected 140 out of 160 intentional looks (87.5%). Experiments with EYEKEYS have shown that it is an easily-used computer input and control device for able-bodied people and has the potential to become a practical tool for people with severe paralysis.

Conclusion

Work on the above problems is ongoing, and many open problems remain. Since gestural communication involves a combination of channels, methods for simultaneous analysis of multiple gesture channels are being developed. Efforts are also underway to develop statiotemporal data mining methods for the automatic discovery of patterns (and anomalies) in human motion data.

Acknowledgments
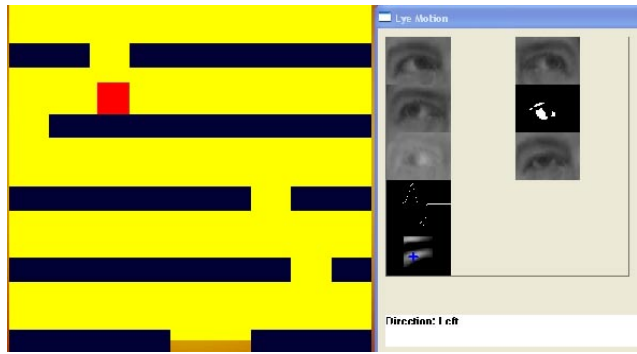
Figure 5. The EyeKeys System applied to the Block-Escape Game: The goal of the game is to maneuver a block (red) through openings of moving walls (black). The video-based system analyzes the eye motion of a computer user (on the right) and sends a command "move block left" to the game interface (on the left).

References

[1] J. Alon, V. Athitsos, and S. Sclaroff. Fast and accurate gesture spotting using subgesture reasoning and pruning of unlikely dynamic programming paths. *B.U. Comp. Sci. Tech. Report*, 2005.

[2] J. Alon, V. Athitsos, and S. Sclaroff. Simultaneous localization and recognition of dynamic hand gestures. *Proc. IEEE Motion Workshop*, 2005.

[3] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. Boostmap: A method for efficient approximate similarity rankings. *Proc.CVPR*, 2004.

[4] V. Athitsos and S. Sclaroff. Estimating 3D hand pose from a cluttered image. *Proc. CVPR*, 2003.

[5] M. Betke, J. Gips, and P. Fleming. The Camera Mouse: Visual tracking of body features to provide computer access for people with severe disabilities. *IEEE Trans Neural Sys. Rehab. Eng.*, 10(1):1–10, 2002.

[6] R. Li, T.P. Tian, and S. Sclaroff. Articualted pose estimation in a learned smooth space for feasible solutions. *Proc. IEEE Workshop on Learning in Computer Vision and Pattern Recognition*, 2005.

[7] J. J. Magee, M. Betke, M. R. Scott, and B. N. Waber. A real-time vision interface based on gaze detection – EyeKeys. In B. Kisacanin, V. Pavlovic, and T. S. Huang, ed., *Real-Time Vision for Human-Machine Interaction*. Springer-Verlag, 2005.