Recognition of Arm Movements

Z. Duric F. Li H. Wechsler
Department of Computer Science
George Mason University
Fairfax, VA 22030
{zduric,fli,wechsler}@cs.gmu.edu

Abstract

This paper describes a method for detection, tracking and recognition of lower arm and hand movements from color video sequences using a linguistic approach driven by motion analysis and clustering techniques. The novelty of our method comes from (i) automatic arm detection, without any manual initialization, foreground or background modeling, (ii) gesture representation at different levels of abstraction using a linguistic approach based upon signal to symbolic mappings, and (iii) robust matching for gesture recognition using the weighted largest common sequence (of symbols). Learning Vector Quantization abstracts the affine motion parameters as morphological primitive units, i.e., "letters", clustering techniques derive sequence of letters as "words" for both subactivities and the transition occurring between them, and finally the arm activities are recognized in terms of sequences of some subactivities. Using activity cycles from six kinds of arm movements, e.g., (slow and fast pounding), striking, swing, swirl, and stirring, which were not available during training, the performance achieved is perfect - 100% - if one allows, as it should be the case for invariance purposes, to recognize slow and fast pounding video sequences as one and the same type of activity.

1. Introduction

Gestures, characteristic of nonverbal interactions, use motions of the limb or body as a means of both communication and expression. What is exactly a gesture ? According to Knapp and Hall (1997) "Gestures are movements of the body (or some part of it) used to communicate an idea, intention, or feeling. Many of these activities are made with the arms/hands. Gestures perform many functions. They may replace speech, regulate the flow and rhythm of interaction, maintain attention, add emphasis and/or clarity to the speech...".

Learning has long been a central issue in the understanding of intelligence as it plays a fundamental role in regulating the balance between internal representations and external regularities. As "versatility, generalization, and scalability are desirable attributes in most vision systems the only solution is to incorporate learning capabilities within the vision system" [9]. Gesture recognition involves both motion analysis and pattern recognition, often referred to as the where and what problems. Our approach to recognize arm movements incorporates clustering techniques, which connect the motion analysis and gesture recognition streams of computation for generalization and robustness purposes.

Gestures have to be parsed and interpreted by the computer in order to iteratively construct and refine a model of the human's affective and cognitive states. The availability of such users' models can be then used in an adaptive fashion to enhance human-computer interactions and to make them appear intelligent, i.e., causal, to an outside observer. Towards that end, this paper describes a novel method for the detection, tracking and recognition of arm movements from color video sequences. The novelty of our method comes from (i) automatic arm detection, without any manual initialization, foreground or background modeling, (ii) gesture representation at different abstraction levels using a linguistic approach based upon signal to symbolic mappings, and (iii) robust matching for gesture recognition using the largest common sequence (of symbols).

2. From Signals to Symbols

Recent reviews on machine analysis of human motions by Gavrila, Aggarwal and Cai [3, 1], and Moeslund [8] provide excellent coverage of research on the detection, tracking, and recognition of human motion. It is widely accepted that automatic detection of moving objects, their accurate tracking, and the interpretation and recognition of long image sequences remains very challenging. Bobick [2] has

Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition (FGR'02)
0-7695-1602-5/02 $17.00 © 2002 IEEE

IEEE COMPUTER SOCIETY

proposed a taxonomy of movement, activity, and action. In his taxonomy, movements are primitives, requiring no contextual or sequence knowledge in order to be recognized. Activities are sequences of movements or states, where the only knowledge required to recognize them involves statistics of the sequence. According to Bobick, most of the recent work in gesture understanding falls within this category. Actions are larger-scale events which typically include interactions with the environment and causal relationships. An important distinction between these levels is the degree to which time must be explicitly represented and manipulated, ranging from simple linear scaling of speed to constraint-based reasoning about temporal intervals. The taxonomy proposed by Bobick motivated our linguistic approach and the search for appropriate symbolic representations.

3. Motion Analysis

We describe here the components responsible for moving arm detection, motion estimation, and tracking. Normal flow is used to detect a moving arm automatically. Expectation Maximization (EM), uniform sampling, and a shortest path algorithm are used to estimate the boundary of the arm. An affine motion model is fit to the arm region using residual analysis and outlier rejection for robust parameter estimation. The estimated parameters are used for both predicting the location of the moving arm and encoding its motion.

3.1. Normal Flow Estimation

We first consider normal flow in gray level images. Let $\vec{\imath}$ and $\vec{\jmath}$ be the unit vectors in the x and y directions, respectively; $\delta\vec{r} = \vec{\imath}\delta x + \vec{\jmath}\delta y$ is the projected displacement field at the point $\vec{r} = x\vec{\imath} + y\vec{\jmath}$. If we choose a unit direction vector $\vec{n}_r = n_x\vec{\imath} + n_y\vec{\jmath}$ at the image point $\vec{r}$ and call it the normal direction, then the *normal displacement field* at $\vec{r}$ is $\delta\vec{r}_n = (\delta\vec{r}\cdot\vec{n}_r)\vec{n}_r = (n_x\delta x + n_y\delta y)\vec{n}_r$. $\vec{n}_r$ can be chosen in various ways; the usual choice (and the one that we use) is the direction of the image intensity gradient $\vec{n}_r = \nabla I/\|\nabla I\|$.

Note that the normal displacement field along an edge is orthogonal to the edge direction. Thus, if at time t we observe an edge element at position $\vec{r}$, the apparent position of that edge element at time $t + \Delta t$ will be $\vec{r} + \Delta t\delta\vec{r}_n$. This is a consequence of the well-known *aperture problem*. We base our method of estimating the normal displacement field on this observation.

For an image frame (say collected at time t) we find edges using an implementation of the Canny edge detector. For each edge element, say at $\vec{r}$, we resample the image locally to obtain a small window with its rows parallel to the image gradient direction $\vec{n}_r = \nabla I/\|\nabla I\|$. For the next image frame (collected at time $t_0 + \Delta t$) we create a larger window, typically twice as large as the maximum expected value of the magnitude of the normal displacement field. We then slide the first (smaller) window along the second (larger) window and compute the difference between the image intensities. The zero of the resulting function is at distance u_n from the origin of the second window; note that the image gradient in the second window at the positions close to u_n must be positive. Our estimate of the normal displacement field is then $-u_n$, and we call it the *normal flow*.

In color images (RGB) we apply the Canny edge detector to each color band to obtain partial derivatives r_x, r_y, g_x, g_y, b_x, b_y for the (r)ed, (g)reen, and (b)lue bands. Edges in color images can be computed using a standard technique used for processing multi-channel imagery [5]. One defines a matrix S,

$$S = \begin{pmatrix} r_x^2 + g_x^2 + b_x^2 & r_x r_y + g_x g_y + b_x b_y \\ r_x r_y + g_x g_y + b_x b_y & r_y^2 + g_y^2 + b_y^2 \end{pmatrix}.$$

The trace of S corresponds to the edge strength. If there is an edge at point (x, y), the larger eigenvalue of S, λ_1, corresponds to the edge strength. The corresponding eigenvector (n_x, n_y) represents the edge direction. Therefore we can treat color edges in the same manner as we have treated gray level edges. The only difference is that the edge strength and the edge direction correspond to the larger eigenvalue of S and its corresponding eigenvector.

For each edge element, say at $\vec{r}$, we resample the three image color bands locally to obtain three small windows with their rows parallel to the image gradient direction $\vec{n}_r = (n_x, n_y)$. For the next image frame (collected at time $t_0 + \Delta t$) we create a larger window, typically twice as large as the maximum expected value of the magnitude of the normal displacement field. We then slide the first (smaller) window along the second (larger) window and compute the difference between the image intensities in all three color bands. The result is a vector function $(\delta_r, \delta_g, \delta_b)$ of the color differences. The magnitude of this vector has a zero crossing at distance u_n from the origin of the second window; the difference vector changes sign around the zero crossing. We estimate the zero crossing by comparing the magnitudes of the two difference vectors pointing in opposite directions.Our estimate of the normal displacement field is then $-u_n$, and we call it the *normal flow*.

3.2. Moving Arm Detection and Delineation

In a typical image the background is larger than the foreground and most background edges do not move; note that

IEEE COMPUTER SOCIETY

the shadows move. However, due to various factors including camera noise, shadows, and lightning variations between frames [1] the normal flow in the background is nonzero. In addition, the foreground edges usually have larger motion than the background edges, but due to interaction of the background and the foreground ("T-junctions") we cannot compute flow everywhere and at some points computed values are not reliable. To separate the foreground and the background edges, we assume that the the normal flow values (projections on the image gradients) in the background have a Gaussian distribution. We use the Expectation Maximization (EM) to fit a Gaussian to the histogram of normal flow values. We assume that the normal flow values $< 4\sigma$ belong to the background and that the normal flow values $\geq 4\sigma$ belong to the foreground; we use 4σ threshold to reduce noise in the foreground detection.

We compute a bounding box for the parts of the image that have normal flow values higher than the 4σ threshold. We scan the inside of the box to find the outside layer of the points within the box. We select points which have large values of the gradient as well as large normal flow values. We choose sample points from the selected points based on similarity of their gradient and flow values with their neighbors. The sample points are linked to their neighbors using a variation of the Dijkstra's shortest path algorithm to create a connected contour. The cost of the path is obtained by subtracting the gradient magnitude value from the maximum gradient magnitude computed for the image.

3.3. Estimating Affine Motion Parameters

Let (x, y) be the image coordinates of a pixel in an image $I(x, y)$ and let the image be centered at $(0, 0)$. Consider an affine transformation where (x', y') are image coordinates in the transformed image $I'(x', y')$ and $a - f$ are the transform parameters. We use all image points with high gradient values within or on the detected contour to estimate the affine flow parameters for that frame. If one subtracts the vector $(x \quad y)^T$ from both sides of affine equation one obtains the following expression for the displacement $(\delta x, \delta y)$ of the point (x, y):

$$\begin{pmatrix} \delta x \\ \delta y \end{pmatrix} = \begin{pmatrix} a-1 & b \\ c & d-1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \equiv \begin{pmatrix} a_1 & b \\ c & d_1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}. \quad (1)$$

Using Equation (1) we obtain the normal displacement field at (x, y) as

$$\begin{aligned} \delta\vec{r}_n \cdot \vec{n}_r &= n_x\delta x + n_y\delta y \\ &= a_1 n_x x + b n_x y + e n_x + c n_y x + d_1 n_x y + f n_y \\ &\equiv \mathbf{a} \cdot \mathbf{u} \end{aligned} \quad (2)$$

where $\vec{n}_r = n_x\vec{\imath} + n_y\vec{\jmath}$ is the gradient direction at (x, y), $\mathbf{a} = (n_x x \;\; n_x y \;\; n_x \;\; n_y x \;\; n_y y \;\; n_y)^T$, and $\mathbf{u} = (a_1 \; b \; e \; c \; d_1 \; f)^T$ is the vector of affine parameters.

We use the method described in Section 3.1 to compute normal flow. For each edge point $\vec{r}_i$ we have one normal flow value $u_{n,i}$ which we use as the estimate of the normal displacement at the point. This gives us one approximate equation $\mathbf{a}_i \cdot \mathbf{u} \approx u_{n,i}$. Let the number of edge points be $N \geq 6$. We then have a system

$$\mathbf{A}\mathbf{u} - \mathbf{b} = E$$

where $\mathbf{u}$ is an N-element array with elements $u_{n,i}$, $\mathbf{A}$ is an $N \times 6$ matrix with rows $\mathbf{a}_i$, and E is an N-element error vector. We seek $\mathbf{u}$ that minimizes $\|E\| = \|\mathbf{b} - \mathbf{A}\mathbf{u}\|$; the solution satisfies the system $\mathbf{A}^T\mathbf{A}\mathbf{u} = \mathbf{A}^T\mathbf{b}$ and corresponds to the linear least squares solution.

3.4. Arm Tracking

We subtract the normal flow from the normal motion field given by the affine parameters to compute the *residual normal flow field*. We observe that the residual normal flow values should be small and that their distribution should be similar to the distribution of the background normal flow values. However, due to various factors including noise, shadows, lightning variations, and interactions of the background with the foreground we expect to have residual values corresponding to outliers. We use the same EM method as in Section 3.2 to estimate the Gaussian distribution and detect the outliers. After the outlier detection and rejection we reestimate the affine motion parameters by using the remaining normal flow vectors; the second estimate is usually much better than the first one. Fig.1 shows examples of the residual flow fields and the reestimated affine flows.

After outlier rejection and affine parameter reestimation we use the computed affine motion parameters to predict the position of the arm in the next frame. Note that we only use those points in the current frame that have small residuals; we assume that points with large flow residuals are erroneous. We vary the affine parameters in a small range to obtain multiple candidate points in the next frame; this is used to help account for errors in the model (e.g., the shirt moves nonrigidly) and in the parameter estimation. From the candidate points in the next frame we select those points that have the largest motion and gradient magnitude values in their neighborhoods. From these points we sample uniformly and link the neighbors by a shortest path algorithm; the neighbors are determined with respect to the points corresponding to them in the previous frame. Using this method we have been able to fully automate detection and tracking of moving arms in long image sequences.

[1] We rely on normal ambient lights that correspond to a mixture of neon light panels and outdoor lights.

IEEE COMPUTER SOCIETY

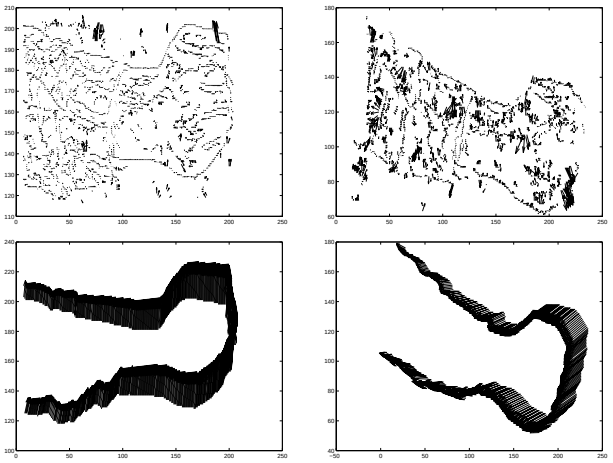

Figure 1. Residual and reestimated flows for the detected arms. Upper row: residual flow computed as the difference between the computed normal flow and the estimated affine normal motion field. Lower row: reestimated affine flow after outlier rejection.

4. Learning

Learning Vector Quantization (LVQ) [6] first abstracts the affine motion parameters as morphological primitive units, i.e., "letters", clustering driven by homogeneity then finds out the dictionary "word" entries corresponding to subactivities and the transitions holding between them, and finally the arm activities are recognized in terms of sequences of subactivities. Similar to a phonotopic map one can parse ("read") and interpret ("recognize") arm movement activities along the "linguistic" trajectories they trace. The learning component implements a linguistic approach and provides for computational efficiency, conceptual abstraction and hierarchical modeling, and robustness. The first few cycles of each moving arm sequence are used for training, while the remaining cycles are used for testing.

4.1. Self-Organization

Self-organization involves the unsupervised ability to learn and organize ('cluster') sensory information without the benefit of a teacher. Learning is driven here by homogeneity. One example of self-organizing algorithms is the Self-Organizing Feature Maps (SOFM) [7], driven by competitive learning. As a result of competition, a meaningful lattice-like coordinate system eventually emerges to represent the sensory input. Furthermore, the spatial locations of the neurons across the lattice induce a compressed and intrinsic feature representation of the sensory patterns. There are many situations where the clusters derived as a result of self-organization have to be appropriately labeled as it would be the case for information retrieval. Towards that end, one expands the SOFM using a supervised learning scheme and the result is Learning Vector Quantization (LVQ) [4]. The labeled collection of clusters corresponds to a (quantized) codebook of compressed sensory prototype patterns and it defines a Voronoi tesselation.

4.2. Parsing and Interpretation

Arm activities include *striking*, (*slow* or *fast*)*pounding*, *swirling*, *swing*, and *stirring* (augmented by tool using). Video sequences are represented using three hierarchical layers of abstraction in order to facilitate parsing and interpretation of arm movements. LVQ self-organizes and compresses the original input space spanned by affine parameters using 14 (fourteen) symbols, i.e., "letters", each of them corresponding to an individual tile in the Voronoi tesselation. One can show using the inner product of the prototype vectors that the letters are quite dissimilar. Clustering derives the next layer, which consists of 14 (fourteen) "words", corresponding to specific subactivities and transitions ("edges") as segmentation ("punctuation") points between subactivities. Examples of words include ("UP" - AAAAAAAAA), ("LEFT" - QQQQQQQQQLQ), while examples of transitions include (FKHGJ) and (FJHFJKQ). Please note that some conceptual subactivity can map to more than one cluster. One can again show that the dictionary "word" entries are quite dissimilar. The last representational layer encodes arm activities in terms of simple word sequences using clustering, e.g., *pounding* consists of repeating cycles of *up* and *down* segments, *swirling* consists of repeating *circle* segments, *swing* consists of repeating *left* and *right* segments and so on. Finally, one can recognize the arm activity as follows : (a) derive the letter sequence using the prototypes derived using LVQ; (b) derive the corresponding subactivity (and transition) sequence via flexible and robust matching between the sequence of letters derived so far and the subactivity clusters using as distance a properly weighted largest common letter subsequence and the nearest neighbor as the classification rule. The identity of the sequence is determined by matching the sequence of subactivities, punctuated by transitions, against activities prototypes learned earlier. The arm activities are again quite dissimilar to each other.

5. Experimental Results

We show here the feasibility of our method using the six kinds of arm movements referred to in the previous section. Each sequence consists of several hundred (100 - 500)

IEEE COMPUTER SOCIETY

frames. These images were collected using a Sony DFW-VL500 progressive color scan camera; the frame rate was thirty frames per second and the resolution was 320×240 pixels per frame. The moving arm was detected and tracked automatically using the methods described in Section 3. Figure 2 shows 4 frames and the corresponding tracked contours for each kind of arm movement. The input to LVQ learning program consists of the affine motion parameters derived for one cycle (approx. 60 frames) for each of the six kinds of arm movements.

The output of LVQ yields fourteen prototypes; the corresponding affine flows are shown in Figure 3. After the letters are derived by LVQ, each video frame is represented by a letter corresponding to the nearest motion prototype. Next clustering compresses the letter sequences into word sequences. As an example, slow and fast pounding cycles are compressed from 123 and 70 letters, respectively, to $< 01 >$, where 0 and 1 stand for two subactivities – note that the resulting activity representation for pounding is invariant to speed; swing is compressed from 44 letters to $< 6(12) >$, where 6 and (12) stand for two subactivities; stirring is compressed from 115 letters to $< 13965047(12)(13) >$, where 1, 3, 9, 6, 5, 0, 4, 7, (12) and (13) stands for 10 subactivities. The feasibility and the robustness of the proposed method is shown using long and inherently noisy image sequences corresponding to different arm activities, some of them possibly recorded at different speeds. Testing performed on those activity cycles not used during training yields perfect accuracy - 100% – if one allows that slow and fast pounding are recognized as one activity.

6. Conclusions

This paper describes a method for detection, tracking and recognition of lower arm and hand movements from color video sequences using a linguistic approach driven by motion analysis and clustering techniques. The novelty of our method comes from (i) automatic arm detection, without any manual initialization, foreground or background modeling, (ii) gesture representation at different levels of abstraction using a linguistic approach based upon signal to symbolic mappings, and (iii) robust matching for gesture recognition using the largest common sequence (of symbols). The feasibility and the robustness of the proposed method has been shown using long and inherently noisy image sequences corresponding to different arm activities, some of them possibly recorded at different speeds. Using activity cycles from six kinds of arm movements, which were not available during training, the performance achieved is perfect - 100% - if one allows, as it should be the case for invariance purposes, to recognize slow and fast pounding video sequences as one and the same type of activity.

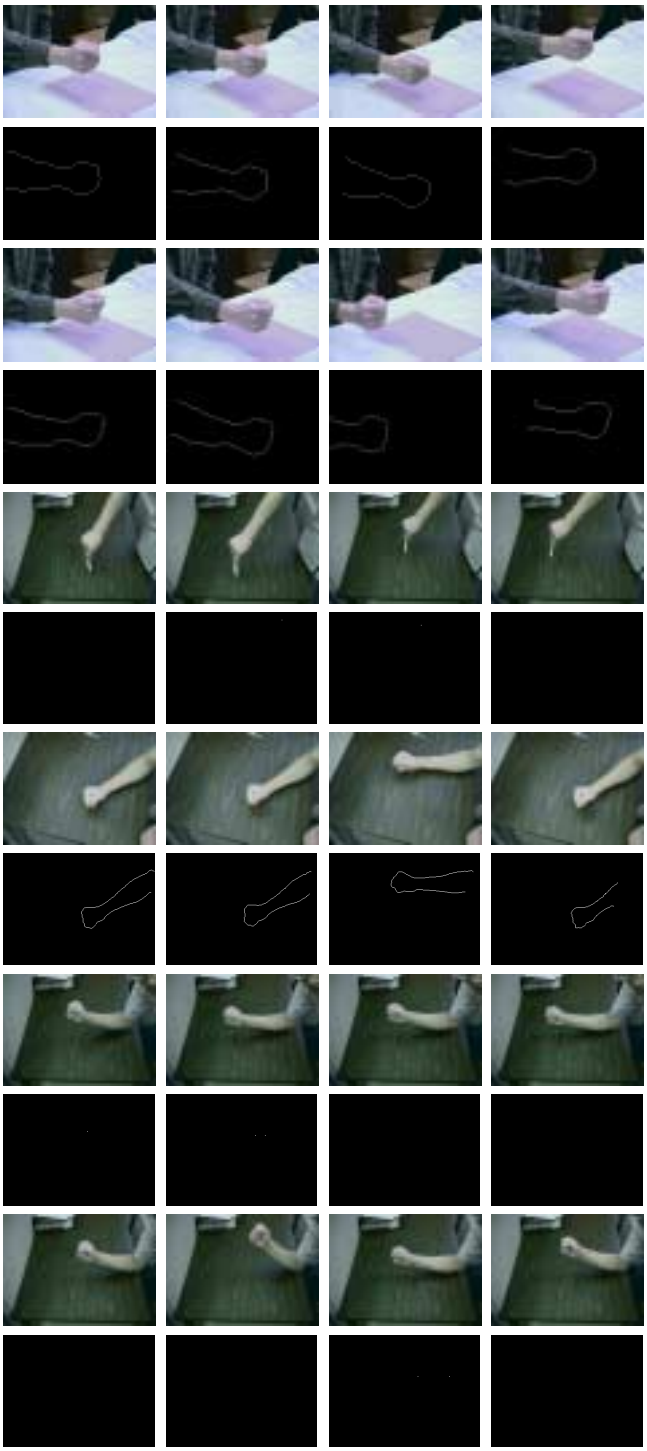

Figure 2. Frames from the "striking", " (slow and fast) pounding", "swirling", "swing", and "stirring"arm movement sequences. First row: color frames. Second row: tracked contours.

IEEE COMPUTER SOCIETY

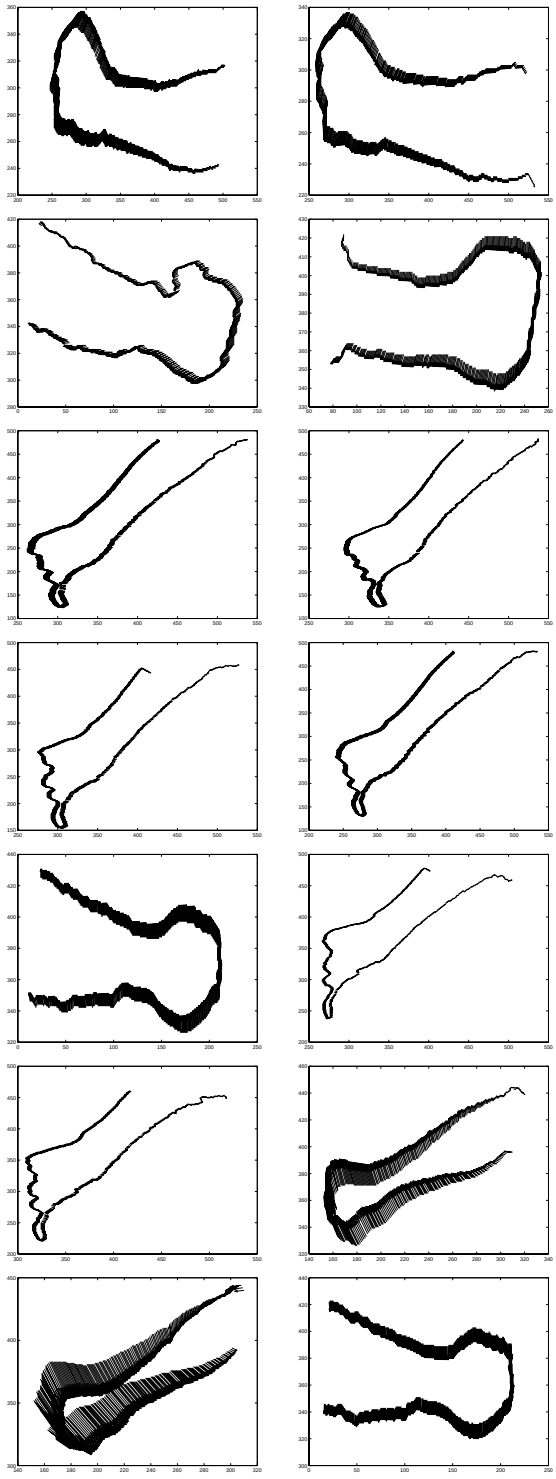

Figure 3. Affine flows corresponding to the 14 (fourteen) prototypes: A, C, F, G, H, I, J, K, L, M, O, Q, R and T.

The linguistic approach described in this paper can be easily expanded in terms of both alphabets and dictionary entries. Furthermore, one can add yet another level of abstraction, and define actions as sequences of activities. Another possible extension could include analysis of motions of wielded tools. The stirring activity considered earlier on as part of our experimental data involved the use of some tool. Our preliminary experiments suggest that one can actually enhance recognition of arm activities by coupling the parsing and interpretation processes for the motions traced by both the lower arm and the wielded tool. Such coupling would be similar in spirit to using Graphical Models in general, and Coupled HMM [10] in particular but different in implementation.

References

[1] J. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding*, 73:428–440, 1999.

[2] A.F. Bobick. Movement, activity, and action: The role of knowledge in the perception of motion. *In Proc. Royal Society Workshop in Knowledge-based Vision in Man and Machine, London, England*, 1997.

[3] D.M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73:82–98, 1999.

[4] S. Haykin. *Neural Networks* (2nd ed). Prentice Hall, 1999.

[5] B. Jähne. *Digital Image Processing*. Springer-Verlag, Berlin, Germany, 1997.

[6] T. Kohonen. The "neural" phonetic typewriter. *Computer*, 21:11–24, 1988.

[7] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, 1995.

[8] T.B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81:231–268, 2001.

[9] S. Nayar and T. Poggio Early visual learning. In *S. Nayar and T. Poggio (Eds.), Early Visual Learning*, Oxford University Press, 1996.

[10] N.M. Oliver, B. Rosario, and A.P. Pentland. A Bayesian computer vision system for modeling human interactions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22:831–843, 2000.

[11] M.L. Knapp and J. A. Hall Nonverbal Communication in Human Interaction, 4th Ed., Harcourt Brace Publishers, Ft. Worth, Texas, 1997.

IEEE COMPUTER SOCIETY