Motion Segmentation and Pose Recognition with Motion History Gradients

Gary R. Bradski

Intel Corporation, Microcomputer Research Labs
SC12-303, 2200 Mission College Blvd.
Santa Clara, CA 95052-8119 USA
gary.bradski@intel.com

James Davis

MIT Media Lab
E15-390, 20 Ames St.
Cambridge, MA 02139 USA
jdavis@media.mit.edu

Abstract

This paper uses a simple method for representing motion in successively layered silhouettes that directly encode system time termed the timed Motion History Image (tMHI). This representation can be used to both (a) determine the current pose of the object and to (b) segment and measure the motions induced by the object in a video scene. These segmented regions are not "motion blobs", but instead motion regions naturally connected to the moving parts of the object of interest. This method may be used as a very general gesture recognition "toolbox". We use it to recognize waving and overhead clapping motions to control a music synthesis program.

1. Introduction and Related Work

Three years ago, a PC cost about $2500 and a low end video camera and capture board cost about $300. Today the computer could be had for under $700 and an adequate USB camera costs about $50. It is not surprising then that there is an increasing interest in the recognition of human motion and action in real-time vision. For example, during these three years this topic has been addressed by [[5][6][7][8][9][10][11][12][15] [17][24]] among others. Several survey papers in this period have reviewed computer vision based motion recognition [25], human motion capture [[22][23]] and human motion analysis [1]. In particular, with the advent of inexpensive and powerful hardware, tracking/surveillance systems, human computer interfaces, and entertainment domains have a heightened interest in understanding and recognizing human movements. For example, monitoring applications may wish to signal only when a person is seen moving in a particular area (perhaps within a dangerous or secure area), interface systems may require the understanding of gesture as a means of input or control, and entertainment applications may want to analyze the actions of the person to better aid in the immersion or reactivity of the experience.

One possible motion representation is found by collecting optical flow over the image or region of interest throughout the sequence, but this is computationally expensive and many times not robust. For example, hierarchical [2] and/or robust estimation [4] is often needed, and optical flow frequently signals unwanted motion in regions such as loose and textured clothing. Moreover, in the absence of some type of grouping, optical flow happens frame to frame whereas human gestures may span seconds. Despite these difficulties, optical flow signals have been grouped into regional blobs and used successfully for gesture recognition [9].

An alternative approach was proposed in [13] where successive layering of image silhouettes is used to represent patterns of motions. Every time a new frame arrives, the existing silhouettes are decreased in value subject to some threshold and the new silhouette (if any) is overlaid at maximal brightness. This layered motion image is termed a Motion History Image (MHI). MHI representations have the advantage that a range of times from frame to frame to several seconds may be encoded in a single image. Thus MHIs span the time scales of human gestures.

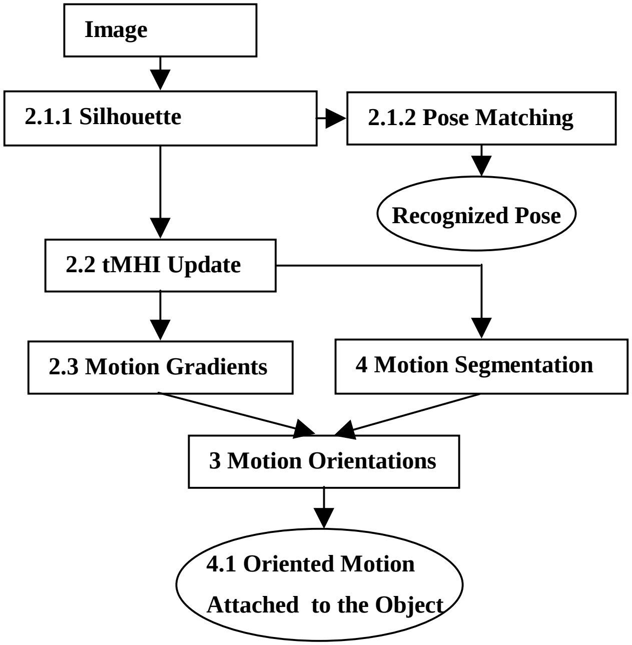

Figure 1 Process flow chart with section numbers.

In this paper, we generalize the Motion History Image to directly encode actual time in a floating point format which we call the timed Motion History Image (tMHI). We take Hu Moment shape descriptors [19] of the current silhouette to recognize pose. A gradient of the tMHI is used to determine normal optical flow (e.g. motion flow orthogonal to object boundaries). The motion is then segmented relative to object boundaries and the motion orientation and magnitude of each region is obtained. The processing flow is summarized in Figure 1 where numbers indicate which section that processing step is described in. The end result is recognized pose, and motion to that pose -- a general “tool” for use in object motion analysis or gesture recognition. Section 5 compares the computational advantages of our approach with the optical flow approaches such as used in [9]. We use our approach in section 6 to recognize walking, waving and clapping motions to control musical synthesis.

2. Pose and Motion Representation

2.1. Silhouettes and Pose Recognition

The algorithm as shown in Figure 1 depends on generating silhouettes of the object of interest. Almost any silhouette generation method can be used. Possible methods of silhouette generation include stereo disparity or stereo depth subtraction [3], infrared back-lighting [12], frame differencing [13], color histogram back-projection [6], texture blob segmentation, range imagery foreground segmentation etc. We chose a simple background subtraction method for the purposes of this paper as described below.

2.1. 1. Silhouette Generation

Although there is recent work on more sophisticated methods of background subtraction [[14][18][21]], we use a simplistic method here. We label as foreground those pixels that are a set number of standard deviations from the mean RGB background. Then a pixel dilation and region growing method is applied to remove noise and extract the silhouette. A limitation of using silhouettes is that no motion inside the body region can be seen. For example, a silhouette generated from a camera facing a person would not show the hands moving *in front of* the body. One possibility to help overcome this problem is to simultaneously use multiple camera views. Another approach would be to separately segment flesh-colored regions and overlay them when they cross the foreground silhouette.

2.1.2. Mahalanobis Match to Hu Moments of Silhouette Pose

For recognition of silhouette pose, seven higher-order Hu moments [19] provide shape descriptors that are invariant to translation and scale. Since these moments are of different orders, we must use the Mahalanobis distance metric [26] for matching based on a statistical measure of closeness to training examples.

$$mahal(x) = (x-m)^T K^{-1}(x-m) \qquad (1)$$

where x is the moment feature vector, m is the mean of the training moment vectors, and K^{-1} is the inverse covariance matrix for the training vectors. The discriminatory power of these moment features for the silhouette poses is indicated by a short example. For this example, the training set consisted of 5 people doing 5 repetitions of 3 gestural poses (“Y”, “T”, and “Left Arm”) shown in Figure 2 done by each of five people. A sixth person who had not practiced the gestures was brought in to perform the gestures.

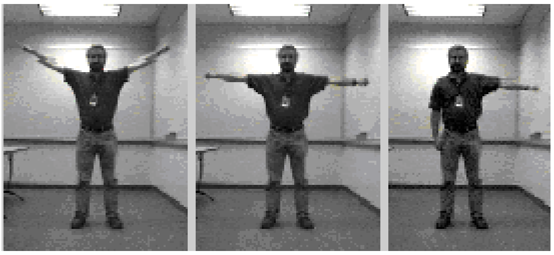

Figure 2 Test postures “Y”, “T” and “|-“.

Table 1 shows typical results for pose discrimination. We can see that even the confusable poses “Y” and “T” are separated by more than an order of magnitude making it easy to set thresholds to recognize test poses against trained model poses.

	Pose Y	Pose T	Pose |-
Test Y	**14**	204	2167
Test T	411	**11**	11085
Test |-	2807	257	**28**

Table 1 Discrimination results of posture recognition. Distance to correct pose model is much smaller than distances to incorrect poses.

An alternative approach to pose recognition uses gradient histograms of the segmented silhouette region [5].

2.2. timed Motion History Images (tMHI)

In this paper, we use a floating point Motion History Image [10] where new silhouette values are copied in with a floating point timestamp in the format: seconds.milliseconds. This MHI representation is updated as follows:

$$tMHI_{\delta}(x,y) = \begin{cases} \tau & \text{if current silhouette at (x,y)} \\ 0 & \text{else if } tMHI_{\delta}(x,y) < (\tau - \delta) \end{cases} \quad (2)$$

where τ is the current time-stamp, and δ is the maximum time duration constant (typically a few seconds) associated with the template. This method makes our representation independent of system speed or frame rate (within limits) so that a given gesture will cover the same MHI area at different capture rates. We call this representation the *timed Motion History Image* (tMHI). Figure 3 shows a schematic representation for a person doing an upward arm movement.

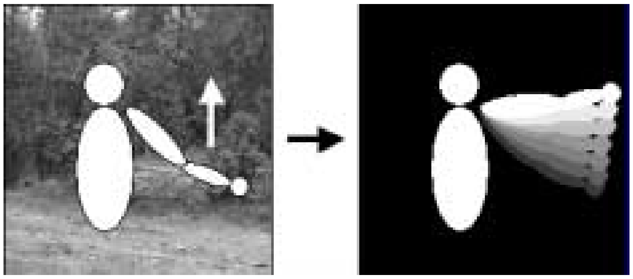

Figure 3 Successive silhouettes of an upward arm movement encoded in floating point timestamps yields the tMHI at right.

2.3. Motion History Gradients

Notice in the right image in Figure 3 (tMHI) that if we took the gradient of the tMHI, we would get direction vectors pointing in the direction of the movement of the arm. Note that these gradient vectors will point orthogonal to the moving object boundaries at each "step" in the tMHI giving us a normal optical flow representation (see middle left image, Figure 4). Gradients of the tMHI can be calculated efficiently by convolution with separable Sobel filters in the X and Y directions yielding the spatial derivatives: $F_x(x,y)$ and $F_y(x,y)$. Gradient orientation at each pixel is then:

$$\phi(x,y) = \arctan\frac{F_y(x,y)}{F_x(x,y)}. \quad (3)$$

We must be careful, though, when calculating the gradient information because it is only valid at locations within the tMHI. The surrounding boundary of the tMHI should not be used because non-silhouette (zero valued) pixels would be included in the gradient calculation, thus corrupting the result. Only tMHI *interior* silhouette pixels should be examined. Additionally, we must not use gradients of MHI pixels that have a contrast which is too low (inside a silhouette) or too high (large temporal disparity) in their local neighborhood. Figure 4 center, left shows raw tMHI gradients. Applying the above criteria to the raw gradients yeilds a masked region of valid gradients in Figure 4 center, right.

After calculating the motion gradients, we can then extract motion features to varying scales. For instance, we can generate a radial histogram of the motion orientations which then can be used directly for recognition as done in [10]. But, an even simpler measure is to find the *global* motion orientation as discussed next.

3. Global Gradient Orientation

Calculation of the global orientation should be weighted by normalized tMHI values to give more influence to the most current motion within the template. A simple calculation for the global weighted orientation is as follows:

$$\bar{\phi} = \phi_{ref} + \frac{\sum_{x,y} angDiff(\phi(x,y),\phi_{ref}) \times norm(\tau,\delta,tMHI_{\delta}(x,y))}{\sum_{x,y} norm(\tau,\delta,tMHI_{\delta}(x,y))}$$

where $\bar{\phi}$ is the global motion orientation, ϕ_{ref} is the base reference angle (peaked value in the histogram of orientations), $\phi(x,y)$ is the motion orientation map found from gradient convolutions, $norm(\tau,\delta,tMHI_{\delta}(x,y))$ is a normalized tMHI value (linearly normalizing the tMHI from 0-1 using the current time-stamp τ and duration δ), and $angDiff(\phi(x,y),\phi_{ref})$ is the minimum, signed angular difference of an orientation from the reference angle. A histogram-based reference angle (ϕ_{ref}) is required due to problems associated with averaging circular distance measurements. Figure 4 shows from left to right a tMHI, the raw gradients, the masked region of valid gradients and finally the orientation histogram with global direction vector calculated.

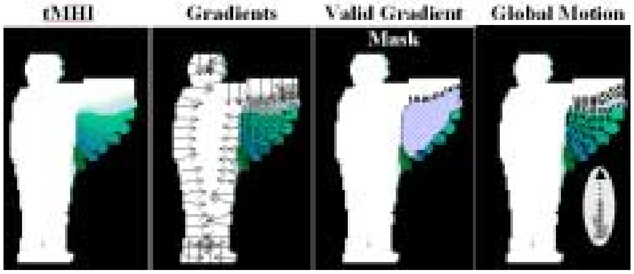

Figure 4 tMHI; Gradients; Mask; Global Orientation.

Figure 5 below shows global motion direction for the movements of kneeling, walking and lifting the arms.

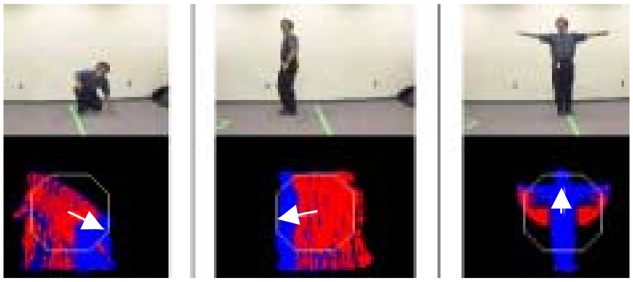

Figure 5 Global motion: kneeling, walking, arms up.

4. Motion Segmentation

Any segmentation scheme begs the question as to what is being segmented. Segmentation by collecting "blobs" of similar direction motion collected frame to frame from optical flow as done in [9] doesn't guarantee that the motion corresponds to the actual movement of objects in a scene. We want to group motion regions that were produced by the movement of parts or the whole of the object of interest. A novel modification to the tMHI gradient algorithm has an advantage in this regard – by labeling motion regions connected to the current silhouette using a downward stepping floodfill, we can identify areas of motion directly attached to parts of the object of interest.

4.1. Motion Attached to Object

By construction, the most recent silhouette has the maximal values (e.g. most recent timestamp) in the tMHI. We scan the image until we find this value, then "walk" along the most recent silhouette's contour to find attached areas of motion. Below, let ***dT*** be a time difference threshold, for example, the time difference between each video frame. The algorithm for creating masks to segment motion regions is as follows (with reference to Figure 6):

1. Scan the tMHI until we find a pixel of the current timestamp. This is a boundary pixel of the most recent silhouette (Figure 6b).
2. "Walk" around the boundary of the current silhouette region looking outside for recent (within ***dT***) unmarked motion history "steps". When a suitable step is found, mark it with a downward floodfill (Figure 6b). If the size of the fill isn't big enough, zero out the area.
3. Store the segmented motion masks that were found.
4. Continue the boundary "walk" until the silhouette has been circumnavigated.

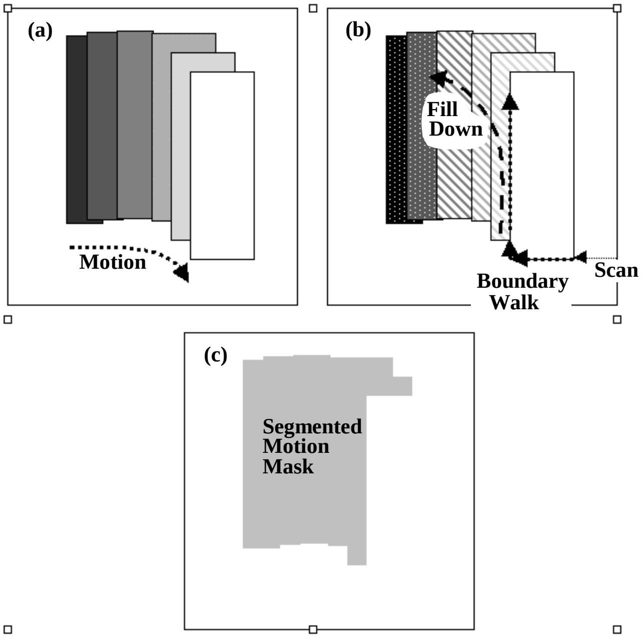

Figure 6 (a) tMHI from a moving "block". (b) Find current silhouette region; "walk" the boundary and fill downwards wherever a step found; (c) Store the found motion masks.

In the algorithm above, "downfill" refers to floodfills that will fill (replace with a labeled value) pixels with the same value, OR pixels of a value one step (within ***dT***) lower than the current pixel being filled. The segmentation algorithm then relies on 2 parameters: (1) The maximum allowable downward step distance ***dT*** (e.g. how far back in time can a past motion be considered to be connected to the current silhouette); (2) The minimum acceptable size of the downward flood fill (else zero it out because the region is too small -- a motion "noise" region).

The algorithm above produces segmentation masks that are used to select portions of the valid motion history gradient described in Section 2.3. These segmented regions may then be labeled with their weighted regional orientation as described in Section 3. Since these segmentation masks derive directly from past motion that "spilled" from the current silhouette boundary of the object, the motion regions are directly connected to the object itself. We give segmentation examples in the section below.

4.2. Motion Segmentation Examples

Figure **8** shows a hand opening and closing in front of a camera. Note that the small arrows correctly catch the finger motion while the global motion is ambiguous.

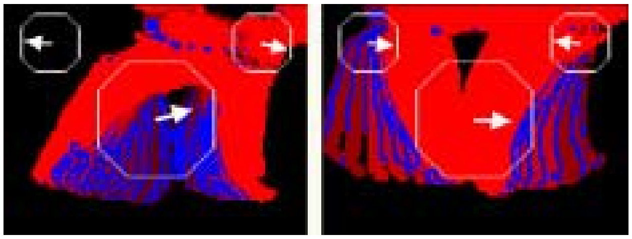

Figure 8 Segmented and global finger motion for hand open and close.

Figure 9 shows a kicking motion from left to right. At left, hands had just been brought down as indicated by the large global motion arrow. The small segmentation arrow is already catching the leftward lean of the body at right. In the center, left image the left leg lean and right leg motion are detected. At center right, the left hand motion and right leg are indicated. At right, the downward leg motion and rightward lean of the body are found.

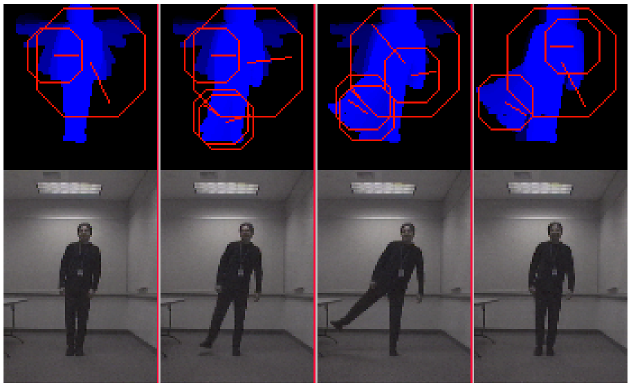

Figure 9 Kicking motion.

Figure 10 shows segmented motion and recognized pose for lifting the arms into a "T" position and then dropping the arms back down. The large arrow indicates global motion over a few seconds, the smaller arrows show segmented motion as long as the corresponding silhouette region moved less than 0.2 seconds ago.

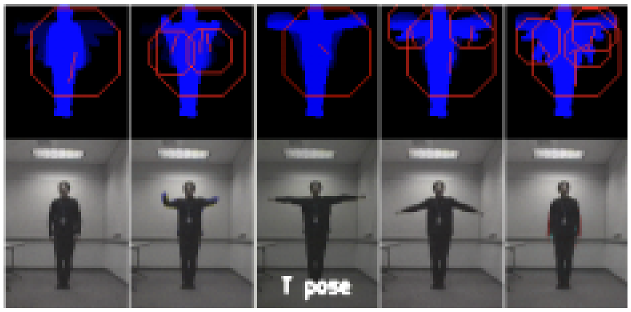

Figure 10 Arms up, T pose, arms down.

5. Comparison to Optical Flow Methods

The motion segmentation method in Cuttler and Turk [9] employed a block matching method of optical flow; we'll use this as our comparison. Our motion segmentation method was implemented using OpenCV, an optimized open source computer vision library maintained by Intel Corporation [28]. The code optimization results are recorded in "CPU cycles per element" (pixel) so that performance numbers are independent of CPU speed. In OpenCV, generic block matching optical flow is reported to run at 1003 cycles per element for an 8x8 window with a search range 8 which can catch motion disparities <= 16 pixels, comparable to the disparities reported by Cuttler and Turk. Cuttler and Turk report optimized block matching results that are about 2.7 times faster due to their use of a sparse correspondence search pattern and only calculating motion for areas indicated by frame differencing. As a result of these approximations, they are able to speed up their algorithm to about 369 cycles per element. These are good results, yet our tMHI method is over 3 times faster at 106 cycles per element. For segmentation, they use a region growing method which takes about 76 cycles per element. We use flood fill which takes 34 cycles per element. In total then, Cuttler and Turk's method consumes 445 cycles per element, our tMHI method uses 140 cycles per element for a factor of 3.2 speed up. Thus, say, using 160x120 video images at 30Hz on a 500MHz Pentium III, the optical flow based method would use about half the CPU, our algorithm would use about one sixth of the CPU leaving more time to do things with the recognized gestures.

6. Example of Use -- Conducting Music

To demonstrate the utility of the tMHI segmentation as a motion gesture recognition "tool", we decided to control a vocal music synthesizer with 3D spatial sound with the following gesture movement controls:

- Detect walk-on/walk-off to set and reset the music.
- Waving gestures control the music tempo.
- Waving with the left(right) arm moves the music left(right). Waving both arms centers the music.
- A full, "jumping jack" over the head clap sends the music outward.
- Clapping with hands held over the head pulls the music back in.

Figure 11 shows the recognition of the above gestures. In the figure the large circle shows direction and amount (pointer length) of global motion, the small circles show segmented motion occurring in quadrants around the user. From this we see that walking on or off can easily be detected by a large sideward motion with two segmented sideward motions on the same side. In the rest of the gestures, waving and clapping can be detected by the sinusoidal motion patterns that they make.

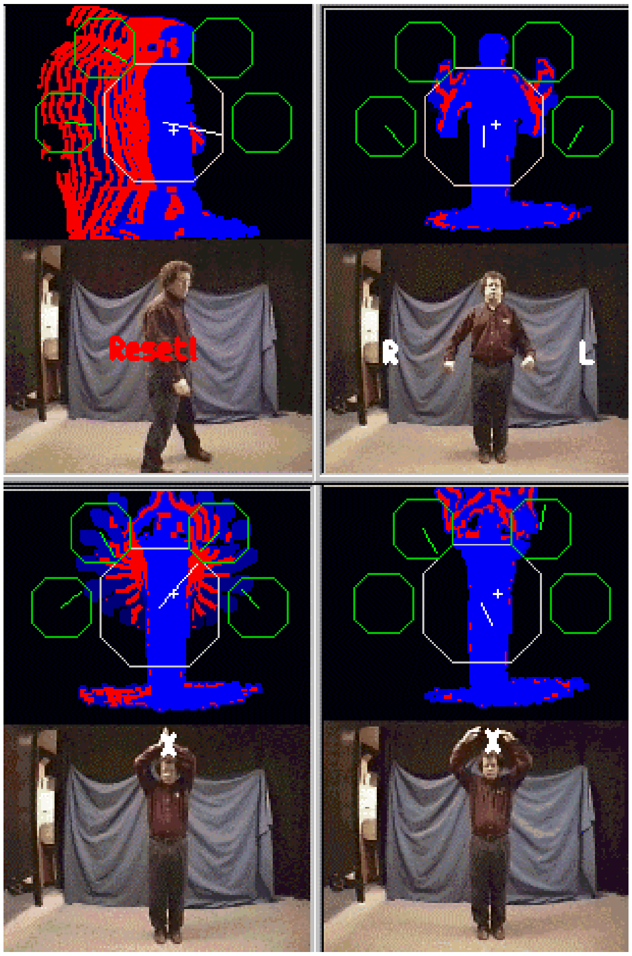

Figure 11 Across from top left to bottom right: Walk-on; down beat; full clap; over head clap detected.

In Figure 12 at top, the angle of motion of the left hand clapping over the user's head shows a sinusoidal pattern. In Figure 12 at bottom, the angle of motion of the left hand downbeats shows a sinusoidal pattern. Many techniques can be used to recognize such patterns, for example, using a Hidden Markov Model (HMM) to learn the sinusoidal parameters.

For our application, the sinusoidal movement patterns were circularly rotated prior to recognition (and display) so that the maximal extent of the gesture would be at bottom. Using this representation, we found recognition to be quite reliable just by detecting a large negative derivative followed by an upward derivative. This catches the movement right at the lowest point of the downbeat or at the point where the hands meet in clapping. These recognitions were automatically detected and displayed as shown in Figure 11. The exact settings for detecting gestures were set using a training tape and then run on another tape. The recognition rate was 100%.

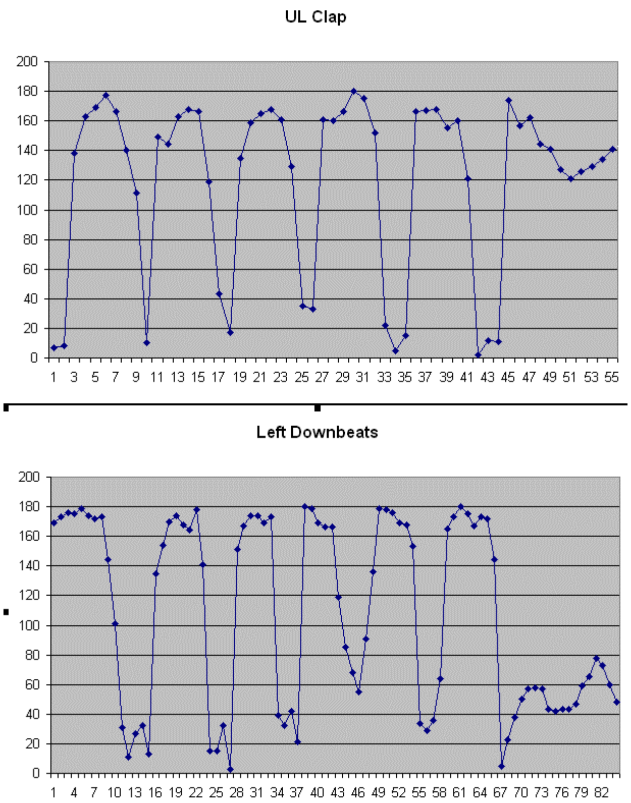

Figure 12 Angle of motion over time shows a sinusoidal pattern for the left hand clapping at top, and the left hand waving downbeats at bottom.

7. Summary

In this paper we extended earlier motion template research [13] by offering a method of calculating normal optical flow motion orientations directly from the motion history template. We also presented a novel method of normal optical flow motion segmentation based on the tMHI that segments motion into regions that are meaningfully connected to movements of the object of interest. This motion segmentation, together with silhouette pose recognition, provides a very general and useful "tool" for gesture recognition. In addition, this new algorithm is computationally quicker than motion segmentation algorithms based on optical flow.

Reference

[1] Aggarwal, J. and Q. Cai. Human motion analysis: a review. IEEE Wkshp. Nonrigid and Articulated Motion Workshop, Pages 90-102, 1997.

[2] Bergen, J., Anandan, P., Hanna, K., and R. Hingorani. Hierarchical model-based motion estimation. In Proc. European Conf. on Comp. Vis., pages 237-252, 1992.

[3] Beymer, D. and Konolige K. Real-Time Tracking of Multiple People Using Stereo, IEEE FRAME-RATE Workshop, http://www.eecs.lehigh.edu/FRAME/. 1999.

[4] Black, M. and P. Anandan. A frame-work for robust estimation of optical flow. In Proc. Int. Conf. Comp. Vis., pages 231-236, 1993.

[5] Bradski, G., Yeo, B-L. and M. Yeung. Gesture for video content navigation. In SPIE'99, 3656-24 S6, 1999.

[6] Bradski, G. Computer Vision Face Tracking For Use in a Perceptual User Interface. In Intel Technology Journal, http://developer.intel.com/technology/itj/q21998/articles/art_2.htm, Q2 1998.

[7] Bregler, C. Learning and recognizing human dynamics in video sequences. In Proc. Comp. Vis. And Pattern Rec., pages 568-574, June 1997.

[8] Cham, T. and J. Rehg. A multiple hypothesis approach to figure tracking. In Proc. Perceptual User Interfaces, pages 19-24, November 1998.

[9] Cuttler, R. and M. Turk. View-based interpretation of real-time optical flow for gesture recognition. Int. Conf. On Automatic Face and Gesture Recognition, page 416-421, 1998.

[10] Davis, J. Recognizing movement using motion histograms. MIT Media lab Technical Report #487, March 1999.

[11] Davis, J. and A. Bobick. Virtual PAT: a virtual personal aerobics trainer. In Proc. Perceptual User Interfaces, pages 13-18, November 1998.

[12] Davis, J. and A. Bobick. A robust human-silhouette extraction technique for interactive virtual environments. In Proc. Modelling and Motion capture Techniques for Virtual Environments, pages 12-25, 1998.

[13] Davis, J. and A. Bobick. The representation and recognition of human movement using temporal templates. In Proc. Comp. Vis. and Pattern Rec., pages 928-934, June 1997

[14] Elgammal, A., Harwood, D. and Davis L. Non-parametric Model for Background Subtraction, IEEE FRAME-RATE Workshop, http://www.eecs.lehigh.edu/FRAME/. 1999.

[15] Freeman, W., Anderson, D., Beardsley, P., et al. Computer vision for interactive computer graphics. IEEE Computer Graphics and Applications, Vol. 18, Num 3, pages 42-53, May-June 1998.

[16] Gavrila, D. The visual analysis of human movement: a survey. Computer Vision and Image Understanding, Vol. 73, Num 1, pages 82-98, January, 1999

[17] Haritaoglu, I. Harwood, D., and L. Davis. W4S: A real-time system for detecting and tracking people in 2 ½ D. European. Conf. On Comp. Vis., pages 877-892, 1998.

[18] Horprasert T., David Harwood, D. and Davis, L. A Statistical Approach for Real-time Robust Background Subtraction and Shadow Detection, IEEE FRAME-RATE Workshop, http://www.eecs.lehigh.edu/FRAME/. 1999.

[19] Hu, M. Visual pattern recognition by moment invariants. IRE Trans. Information Theory, Vol. IT-8, Num. 2, 1962.

[20] Krueger, M. Artificial reality II, Addison-Wesley, 1991.

[21] Martins, F., Nickerson, B., Bostrom, V. and Hazra, R. Implementation of a Real-time Foreground/Background Segmentation System on the Intel Architecture, IEEE FRAME-RATE Workshop, http://www.eecs.lehigh.edu/FRAME/. 1999.

[22] Moeslund, T. Summaries of 107 computer vision-based human motion capture papers. University of Aalborg Technical Report LIA 99-01, March 1999.

[23] Moeslund, T. Computer vision-based human motion capture – a survey. University of Aalborg Technical Report LIA 99-02, March 1999.

[24] Pinhanez, C. Representation and recognition of action in interactive spaces. MIT Media Lab Ph.D. Thesis, June 1999.

[25] Shah, M. and R. Jain. Motion-Based Recognition. Kluwer Academic , 1997.

[26] Therrien, C. Decision Estimation and Classification. John Wiley and Sons, Inc., 1989.

[27] Assembly optimized performance libraries in Image Processing, Signal Processing, JPEG, Pattern Recognition and Matrix math can be downloaded from at http://developer.intel.com/vtune/perflibst/

[28] Open Source Computer Vision Library (OpenCV) in C and optimized assembly modules on Intel's architecture can be downloaded from http://www.intel.com/research/mrl/research/cvlib