Machine Vision and Applications (2002) 13: 174–184

Machine Vision and Applications

Motion segmentation and pose recognition with motion history gradients

Gary R. Bradski[1] **and James W. Davis**[2]

[1] Microcomputer Research Labs, Intel Corporation, Ohio State University, SC12-303, 2200 Mission College Blvd., Santa Clara, CA 95052-8119, USA; e-mail: gary.bradski@intel.com
[2] Department of Computer and Information Science, 583 Dreese Lab, 2015 Neil Ave, Columbus, OH 43210, USA; e-mail: jwdavis@cis.ohio-state.edu

Accepted: 13 August 2001

Abstract. This paper presents a fast and simple method using a timed motion history image (tMHI) for representing motion from the gradients in successively layered silhouettes. This representation can be used to (a) determine the current pose of the object and (b) segment and measure the motions induced by the object in a video scene. These segmented regions are not "motion blobs", but instead are motion regions that are naturally connected to parts of the moving object. This method may be used as a very general gesture recognition "toolbox". We demonstrate the approach with recognition of waving and overhead clapping motions to control a music synthesis program.

Key words: Motion segmentation – Normal optical flow

1 Introduction and related work

Three years ago, a PC cost about U.S. $2500 and a low-end video camera and capture board cost about U.S. $300. Today, the computer could be had for under U.S. $700 and an adequate USB camera for under U.S. $70[1]. It is not surprising then that there is an increasing interest in real-time vision on low-end computers. A heightened interest in understanding and recognizing human movements has appeared in tracking and surveillance systems, human–computer interfaces and entertainment domains. 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.

Recently, there have been several popular approaches to the recognition of human motion [8–11, 13–16, 20, 22, 33], with much emphasis on real-time computation. Several survey papers review vision-based motion recognition [36], human motion capture [31, 32] and human-motion analysis [1]. The most common frameworks to recognition of body motion and gesture include the analysis of temporal trajectories of the motion parameters [6, 29, 35, 40], hidden Markov models (HMMs) and state-space models [37, 39, 41], and static-activity templates [12, 17, 19, 34].

One other possible motion representation to describe an action sequence could be the collection of optical flow over the image or region of interest throughout the sequence, but this is computationally expensive and many times not robust. Hierarchical [2] and/or robust estimation [5] is often needed, and optical flow frequently signals unwanted motion in regions such as those containing loose and textured clothing. Moreover, in the absence of some type of grouping, optical flow happens frame to frame whereas human gestures may span several seconds. Despite these difficulties, optical flow signals have been grouped into regional blobs and used for gesture recognition [13]. An alternative approach was proposed in [17] where successive layering of image silhouettes of a person into a single template was used to represent and recognize patterns of human motion. Every time a new video 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. In this way, MHIs span the time scales of human gestures. In [17], moment features of the entire MHI image were used to recognize particular activities.

The outline of this paper is as follows. Section 2 reviews previous MHI research. Sections 3–5 are summarized in the processing flow chart in Fig. 1 where numbers indicate which section that processing step is described. In this paper, we factor pose from motion and segment the motion regions. We take Hu moment shape descriptors [25] of the current silhouette to recognize pose. We generalize the MHI to directly encode actual time in a floating-point format that we call the timed motion history image (tMHI). 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 of each region is obtained. The end result is rec-

Correspondence to: G.R. Bradski
[1] See [33, 43] for free optimized supporting software.

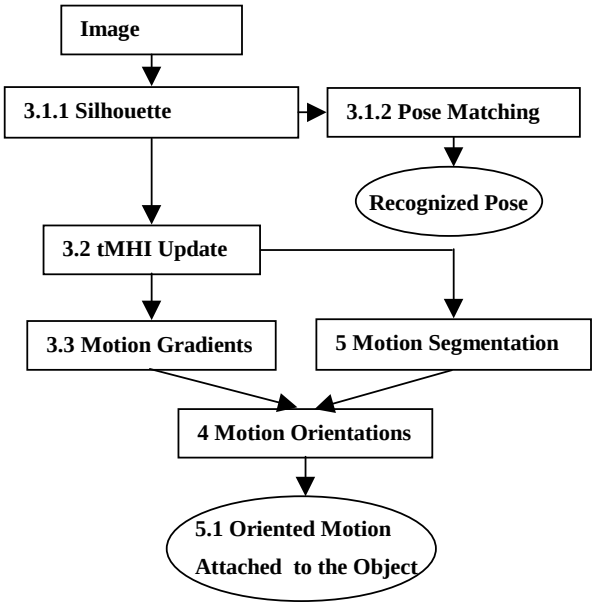

Fig. 1. Process flow chart with section numbers

ognized pose, and motion to that pose – a general "tool" for use in object motion analysis or gesture recognition. Section 6 compares the computational advantages of our approach with other optical flow approaches such as used in [13]. In Sect. 7 we use our approach to recognize walking, waving and clapping motions to control musical synthesis. Section 8 concludes the paper.

2 Previous motion template research

In previous work, Davis and Bobick [17] presented a real-time computer vision approach for representing and recognizing simple human movements. The motivation for the approach was based on how easily people can recognize common human movements (like sitting or push-ups) from low-resolution (blurred) imagery. In those examples there was an almost total lack of recognizable image structure, thus showing the role of motion for recognition. Accordingly, their method relied on "patterns of motion" rather than on structural features as the representation for human motion. In that method, the space-time image volume containing the motion is collapsed into a single 2D template while still perceptually capturing the *essence* of the movement and its temporal structure. The template is generated by layering successive image differences of a moving person, and is referred to as a MHI (similar to [26]). An example is shown in Fig. 2 for the movement of a person stretching her arm over her head.

For recognition of the templates, seven higher-order moments [25] were initially extracted from the MHI and also from a binarized version of the MHI (MEI). These fourteen moments were used as global shape descriptors and temporal recency localizers for the MHI. The moments were then statistically matched to stored examples of different movements. This method of recognition has shown promising results using a large database of movements.

Already, interactive systems have been successfully constructed using the underlying motion history approach as the primary sensing mechanism. One example includes a virtual aerobics trainer [15] that watches and responds to the user as he or she performs the workout (see Fig. 3a). In this system, MHIs are used to watch and recognize the various exercise movements of the person, which affects the response of the virtual instructor. Another application using the MHI approach is The KidsRoom [7]. At one point in this interactive, narrative playspace for children, virtual monsters appear on large video projection screens and teach the children how to do a dance. The monsters then dance with the children, complementing the kids whenever they perform the dance movements (see Fig. 3b). The dancing movements of the children are recognized using MHIs. Thirdly, a simple interactive art demonstration can be constructed from the motion images, reminiscent of Krueger-style interactive installations [27]. By mapping different colors to the various values within the MHI and displaying the result on a large projection screen, a person can have fun "body-painting" over the screen (see Fig. 3c). Other applications in general that must be "aware" of the movements of the person (or people) could also benefit from using this approach. For example, gesture-based computer video games [8, 20] or immersive experiences (e.g. Hawaii flight simulator [9], as shown in Fig. 3d) designed to respond to user gesture control could use the previous template approach or employ our new method as an additional sensing measurement or qualification of the person's movements. Even simpler gesture-based systems that rely on detecting characteristic motion (e.g. hand swipes to change a channel or a slide in a presentation) could profit from the fast and simple motion orientation features we will describe.

The main limitation of the previous motion history approach is that characterization of the template is token (label) based (e.g. "crouching" or "sitting"), where it cannot yield much information other than recognition matches (e.g. it cannot determine that "up" motion is currently happening in a particular image location). We therefore wish to develop additional methods for analysis and recognition that are more sensitive to the local motion information. In the next section, we develop a method of analysis that extracts both pose and directional motion information from the MHI.

3 Pose and motion representation

3.1 Silhouettes and pose recognition

The algorithm as shown in Fig. 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], infra-red back-lighting [16], frame differencing [17], color histogram back-projection [9], texture blob segmentation, range imagery foreground segmentation, etc. We chose a simple background subtraction method for the purposes of this paper.

3.1.1 Silhouette generation

Although there is recent work on more sophisticated methods of background subtraction [18, 24, 30], we use a fast,

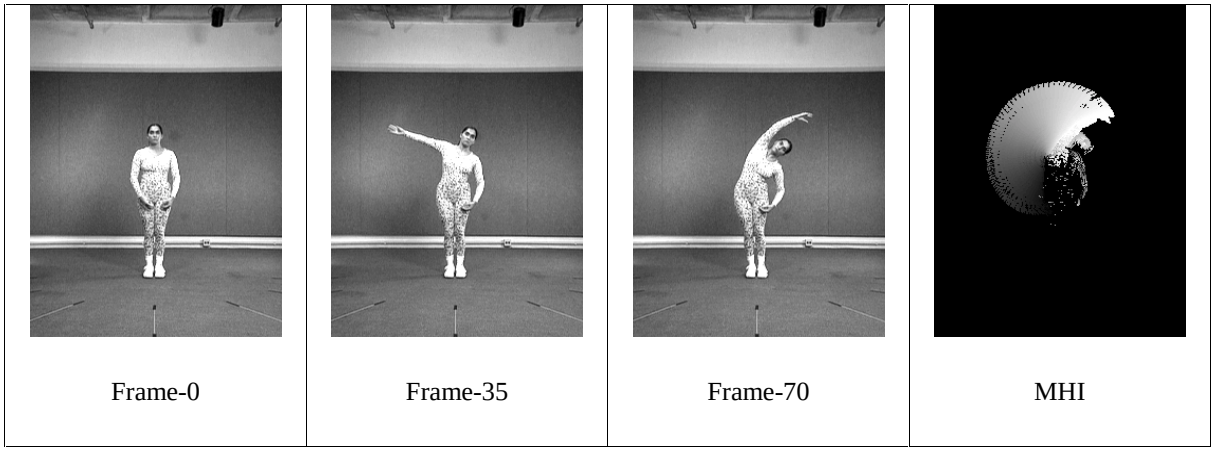

Fig. 2. Motion history image for arm-stretching movement generated from layered image differences

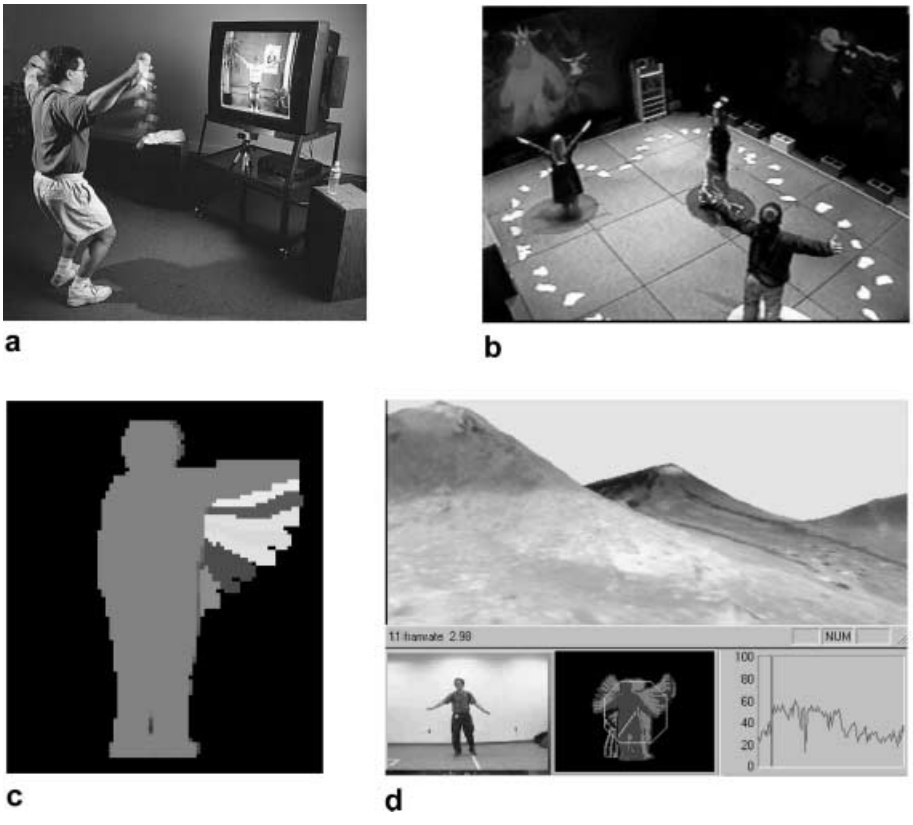

Fig. 3a–d. Interactive systems employing MHIs. **a** Virtual aerobics trainer (photo used with permission from Sam Ogden); **b** The KidsRoom; **c** Interactive body painting; **d** Hawaii flight simulator

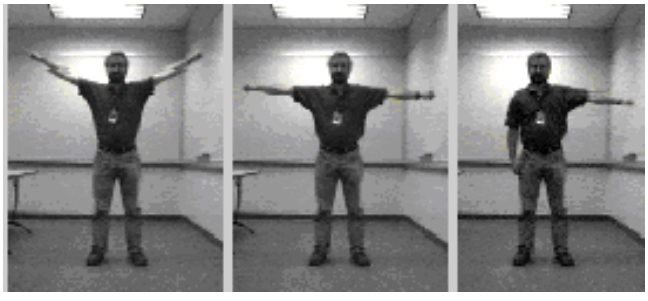

Fig. 4. Test postures Y, T and ⊢

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 the flesh-colored regions and overlay them when they cross the foreground silhouette.

3.1.2 Mahalanobis match to Hu moments of silhouette pose

For recognition of silhouette pose, seven higher-order Hu moments [25] provide shape descriptors that are invariant to translation and scale. Since these moments are of dif-

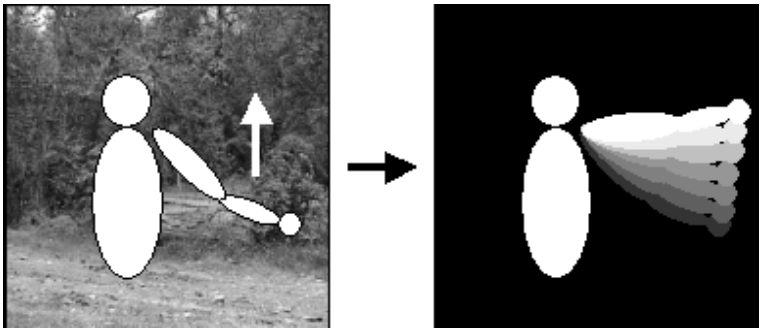

Fig. 5. Successive silhouettes of an upward arm movement encoded in floating-point timestamps yields the tMHI. Brighter tMHI values depict more recent motion

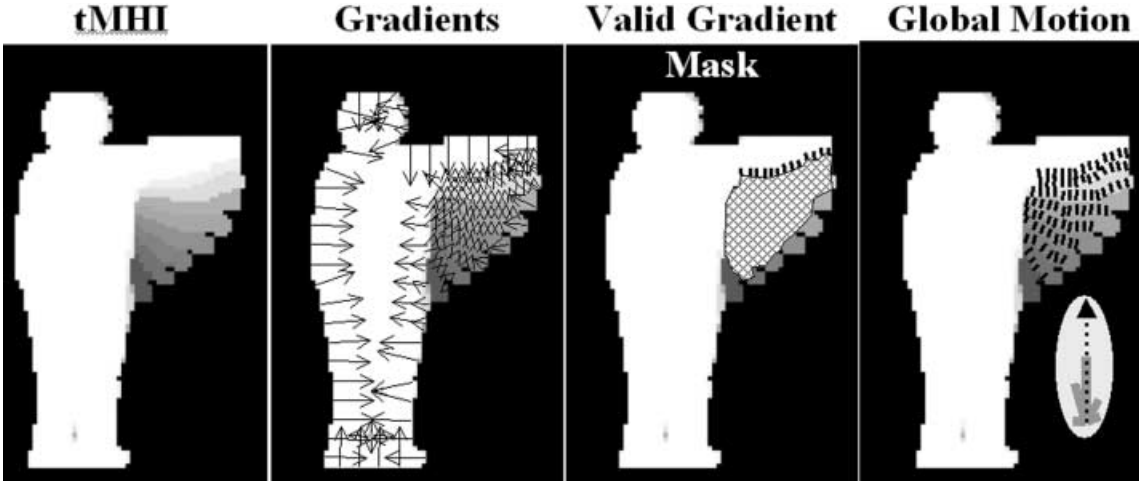

Fig. 6. tMHI; gradients; mask; global orientation

Table 1. Discrimination results of posture recognition. Distance to correct pose model (*in bold*) is much smaller than distances to incorrect poses and therefore it is easy to set a recognition threshold

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

ferent orders, we use the Mahalanobis distance metric [38] for matching based on a statistical measure of closeness to training examples:

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

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 five people performing five repetitions of three gesture poses (Y, T, and ⊢) shown in Fig. 4. A sixth person who had not practiced the gestures was brought in to perform the gestures for testing. 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. An alternative approach to pose recognition uses gradient histograms of the segmented silhouette region [8].

3.2 Timed Motion History Images (tMHI)

In this paper, we use a floating-point MHI [14] 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 timestamp, 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 tMHI. Figure 5 shows a schematic representation of a tMHI for a person doing an upward arm movement.

3.3 Motion history gradients

Notice in the right image in Fig. 5 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 center left image Fig. 6). 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

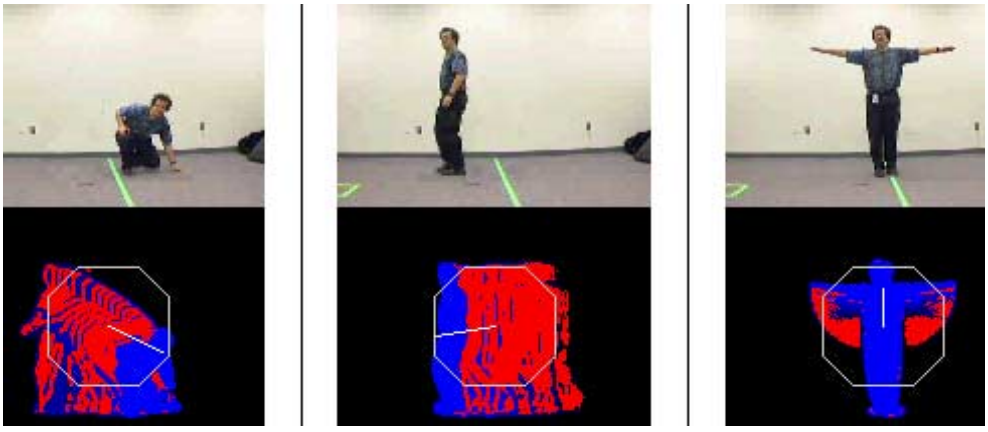

Fig. 7. Global motion: kneeling, walking, arms up

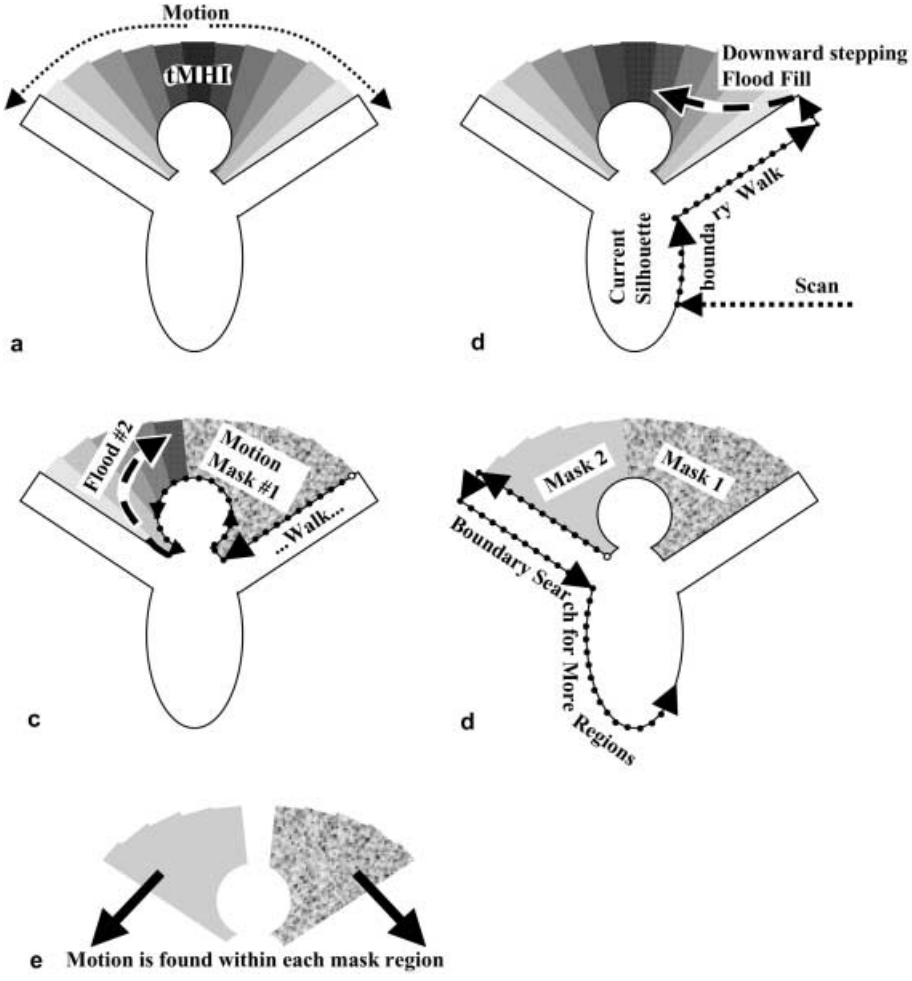

Fig. 8. a tMHI from flapping one's arms. **b** Find current silhouette region; "walk" the boundary until a lower region outside within one dT of the current region is found to downfill; **c** Store the downfill area as a motion mask; Continue walking the boundary looking for areas to downfill until **d** the current silhouette region has been circumnavigated. **e** Find the motion direction within each of the stored mask regions

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

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 value) 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 tMHI pixels that have a contrast which is too low (inside a silhouette) or too high (large temporal disparity) in their local neighborhood. Figure 6 center left shows raw tMHI gradients. Applying the above criteria to the raw gradients yields a masked region of valid gradients in Fig. 6 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 [14]. But, an even simpler measure is to find the global motion orientation.

4 Global gradient orientation

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

$$\overline{\phi} = \phi_{\text{ref}} + \frac{\sum_{x,y} \text{angDiff}(\phi(x,y), \phi_{\text{ref}}) \times \text{norm}(\tau, \delta, \text{t}, \text{MHI}_{\delta}(\text{x}, \text{y}))}{\sum_{\text{x,y}} \text{norm}(\tau, \delta, \text{MHI}_{\delta}(\text{x}, \text{y}))} \quad (4)$$

where $\overline{\phi}$ 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, $\text{norm}(\tau, \delta, \text{MHI}_{\delta}(\text{x}, \text{y}))$ is a normalized tMHI value (linearly normalizing the tMHI from 0–1 using the current timestamp τ and duration δ), and $\text{angDiff}(\phi(x, y), \phi_{\text{ref}})$ 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 6 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 7 shows global motion directions for the movements of kneeling, walking and lifting the arms.

5 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 [13] does not guarantee that the motion corresponds to the actual movement of objects in a scene. We want to group motion regions that are 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.

5.1 Motion attached to object

By construction, the most recent silhouette has the maximal values (i.e. 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. The algorithm for creating masks to segment motion regions is as follows (with reference to Fig. 8):

1. Scan the tMHI until we find a pixel with the current timestamp. This is a boundary pixel of the most recent silhouette (Fig. 8b).
2. "Walk" around the boundary of the current silhouette region looking outside for recent (within dT, e.g. the time difference between each video frame), unmarked motion history "steps". When a suitable step is found, mark it with a downward floodfill (downfill) (Fig. 8b,c). If the size of the fill is not big enough, zero out the area.
3. Store the segmented motion mask that was found (Fig. 8c,d).
4. If the boundary "walk" has not yet circumnavigated the current silhouette, go to 2.
5. Calculate the motion orientation within each mask found in 3 above (Fig. 8e).

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 two parameters: (1) the maximum allowable downward step distance dT (e.g. how far back in time a past motion can be considered to be connected to the current silhouette); and (2) the minimum acceptable size of the downward floodfill (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 Sect. 3.3. These segmented regions may then be labeled with their weighted regional orientation. Since these segmentation masks are derived 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.

5.2 Motion segmentation examples

Figure 9 shows a hand opening and closing in front of a camera. The small windows capture the two motion regions and the large window is the overall motion orientation. Note that the small arrows correctly catch the finger motion while the global motion is ambiguous. Figure 10 shows a kicking motion from left to right. In the leftmost image, the hands have 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 11 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 s ago.

6 Comparison to optical flow methods

In the sections above, we presented a method of computing normal optical flow that measures motion orthogonal to object boundaries. This method was implemented using OpenCV, an optimized open-source computer vision library maintained by Intel Corporation [33]. In OpenCV, the code optimization results are recorded in CPU cycles per element (pixel) so that performance numbers are independent of CPU speed. The normal optical flow calculations take 106 cycles per element. There are three other widely used methods of

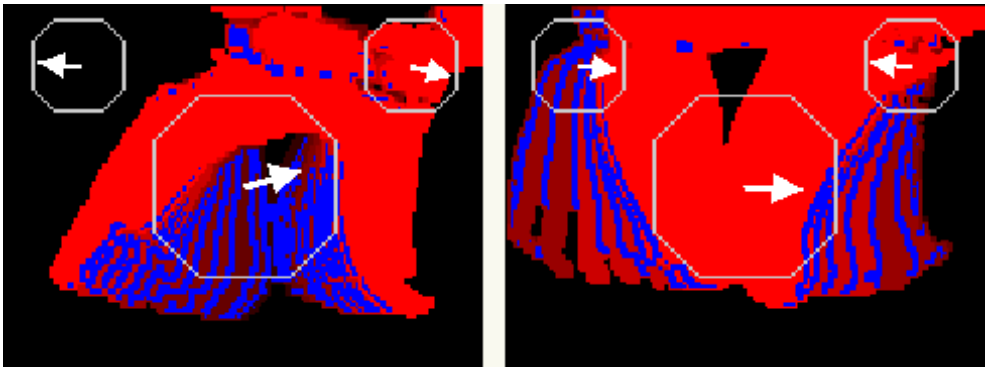

Fig. 9. Segmented and global finger motion for hand open and close

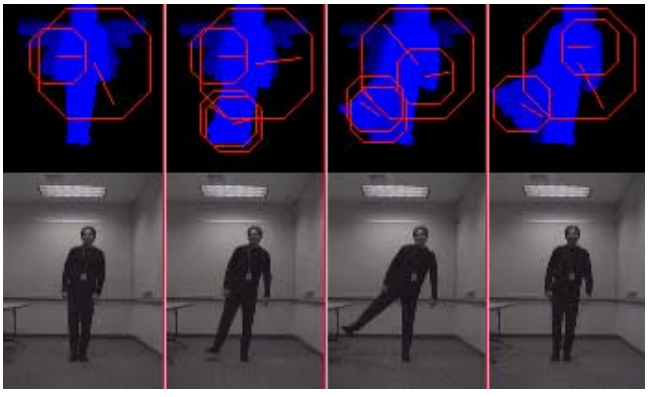

Fig. 10. Kicking motion

optical flow: Lukas–Kanade [28], Horn–Schunck [23] and block matching [4]. These methods of optical flow all rely on a brightness constancy assumption that feature location pixel values between two successive frames do not vary with time or motion displacement. This constraint may be expressed as two equations:

$$I(x + \mathrm{dx}, \mathrm{y} + \mathrm{dy}, \mathrm{t} + \mathrm{dt}) = \mathrm{I}(\mathrm{x}, \mathrm{y}, \mathrm{t}), \tag{5}$$

and

$$\frac{\partial I}{\partial x}\mathrm{dx} + \frac{\partial \mathrm{I}}{\partial \mathrm{y}}\mathrm{dy} + \frac{\partial \mathrm{I}}{\partial \mathrm{t}}\mathrm{dt} = 0. \tag{6}$$

Using a Taylor series expansion of $I(x + \mathrm{dx}, \mathrm{y} + \mathrm{dy}, \mathrm{t} + \mathrm{dt})$, we get

$$I(x + \mathrm{dx}, \mathrm{y} + \mathrm{dy}, \mathrm{t} + \mathrm{dt}) = I(x, y, t) + \frac{\partial I}{\partial x}\mathrm{dx} + \frac{\partial \mathrm{I}}{\partial \mathrm{y}}\mathrm{dy} + \frac{\partial I}{\partial t}\mathrm{dt} \ldots . \tag{7}$$

Letting $\frac{\mathrm{dx}}{\mathrm{dt}} = u$ and $\frac{\mathrm{dy}}{\mathrm{dt}} = v$ and combining Eq. (7) with Eqs. (5) and (6) we get the optical flow constraint equation:

$$\frac{\partial I}{\partial t} = \frac{\partial I}{\partial t}u + \frac{\partial I}{\partial t}v. \tag{8}$$

The optical flow constraint equation has more than one solution, so the different techniques diverge here.

Lukas–Kanade equations are derived assuming that pixels in a neighborhood of each tracked feature move with the same velocity as the feature. In OpenCV, to catch large motions with a small window size (to keep the "same local velocity" assumption), Lukas–Kanade is done in an image pyramid. The total operation consumes 487 cycles per element for a 7×7 window which is 4.6 times slower than the tMHI method for normal optical flow. Horn–Schunck equations are derived assuming a smoothness of flow constraint together with a Lagrangian minimization of deviations from the optical flow constraint equation. In OpenCV, this is a relatively fast procedure consuming 299 cycles per element, or 2.8 times slower than the tMHI method for normal optical flow. Finally, the block matching method uses the idea of brightness constancy to assume small blocks around image features look the same frame to frame. Brute force matching is the slowest method consuming 1003 cycles per element for an 8×8 window with a search range 8. Block matching is 9.5 times slower than the tMHI method for normal optical flow.

The motion segmentation method in Cutler and Turk [13] employed a block matching method of optical flow; we will use this approach for another comparison. As stated above, block matching optical flow in OpenCV runs at 1003 cycles per element for an 8×8 window with a search range 8 which can catch motion disparities $\leqq$ 16 pixels. Cutler and Turk have similar motion disparities and 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 in 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 that takes about 76 cycles per element. We use a floodfill that takes 34 cycles per element. In total then, Cutler and Turk's method consumes 445 cycles per element, while our tMHI method uses 140 cycles per element giving us a factor of 3.2 speed up. Thus, using 160×120 video images at 30 Hz on a 500 MHz Pentium III, the optical flow based method would use about half the CPU and our algorithm would use about one sixth of the CPU leaving more time to do things with the recognized gestures.

7 Example: conducting music

To demonstrate the utility of the tMHI segmentation as a motion gesture recognition tool, we decided to control a

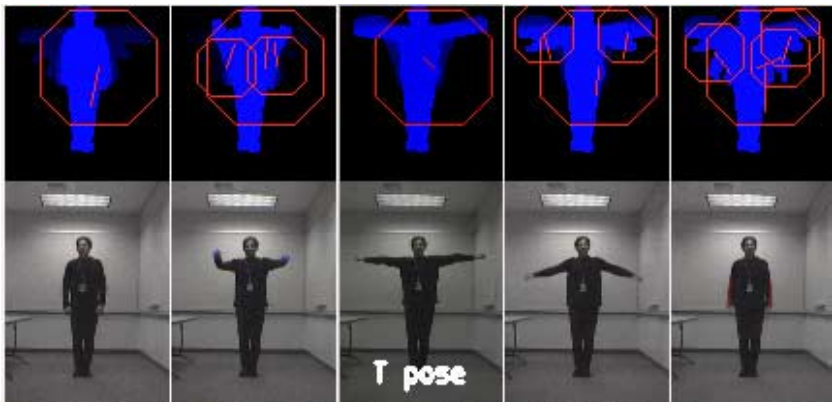

Fig. 11. Arms up, T pose, arms down

vocal music synthesizer with 3D spatial sound using the following gesture movement controls:

- Detect walk-on/walk-off to set and reset the music.
- Waving gestures to control the music tempo.
- Waving either the left or right arm to move the music left or right respectively. Waving both arms centers the music.
- A full "jumping jack" over the head clap to send the music outward.
- Clapping with hands held over the head to pull the music back in.

Figure 12 shows the recognition of the above gestures. In the figure the large circle shows direction and amount (the pointer length) of the global motion, and 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.

In Figure 13 at top, the angle of motion of the left hand clapping over the user's head shows a sinusoidal pattern. In Fig. 13 at bottom, the angle of motion of the left hand downbeats shows a sinusoidal pattern. Many techniques can be used to recognize such distinctive patterns, for example using an 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 the bottom. Using this representation, we found recognition to be quite reliable just by detecting a large negative derivative followed by an upward derivative. By doing this we catch the movement right at the lowest point of the downbeat or at the point where the hands meet in clapping, which is what the user would expect for conducting music. These events were automatically detected and are displayed in Fig. 12 using L or R for waving motions and an X for clapping. To test recognition, The sinusoidal rotation parameters for detecting gestures were set using a training tape of walking on and off, 10 repetitions of the two types of claps, 10 repetitions of the two-handed beats, and 20 repetitions of the single-handed beats, and then was tested on another tape taken later of the same protocols. 100% of the beats and claps as well as waking on and off were correctly detected.

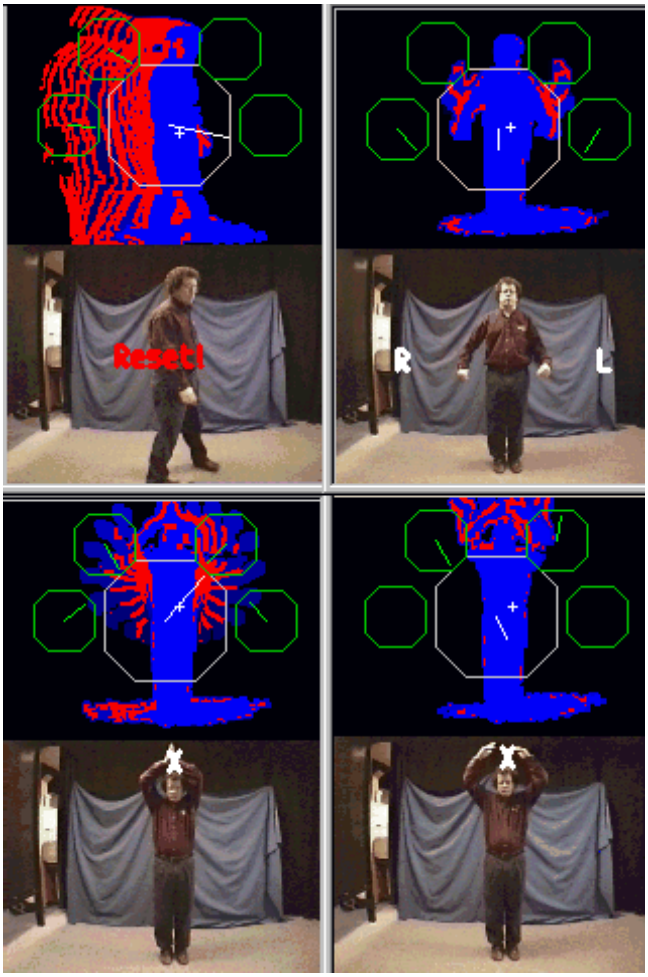

Fig. 12. Across from top left to bottom right: walk-on; down beat; full clap; overhead clap detected

For a more challenging recognition task, we used the pose recognition tape made for Sect. 3.1.2. Here we focused not on the static Y, T and ⊢ poses, but on the gesture of moving into these poses (see Fig. 4). Figure 11 shows movement into and out of a T pose. Note that the Y and T gestures are highly confusable and were made more confusable by the fact that the subjects were not instructed to make gestures, just to assume these poses five times in repetition. For recognition features, we used only the segmented normal optical

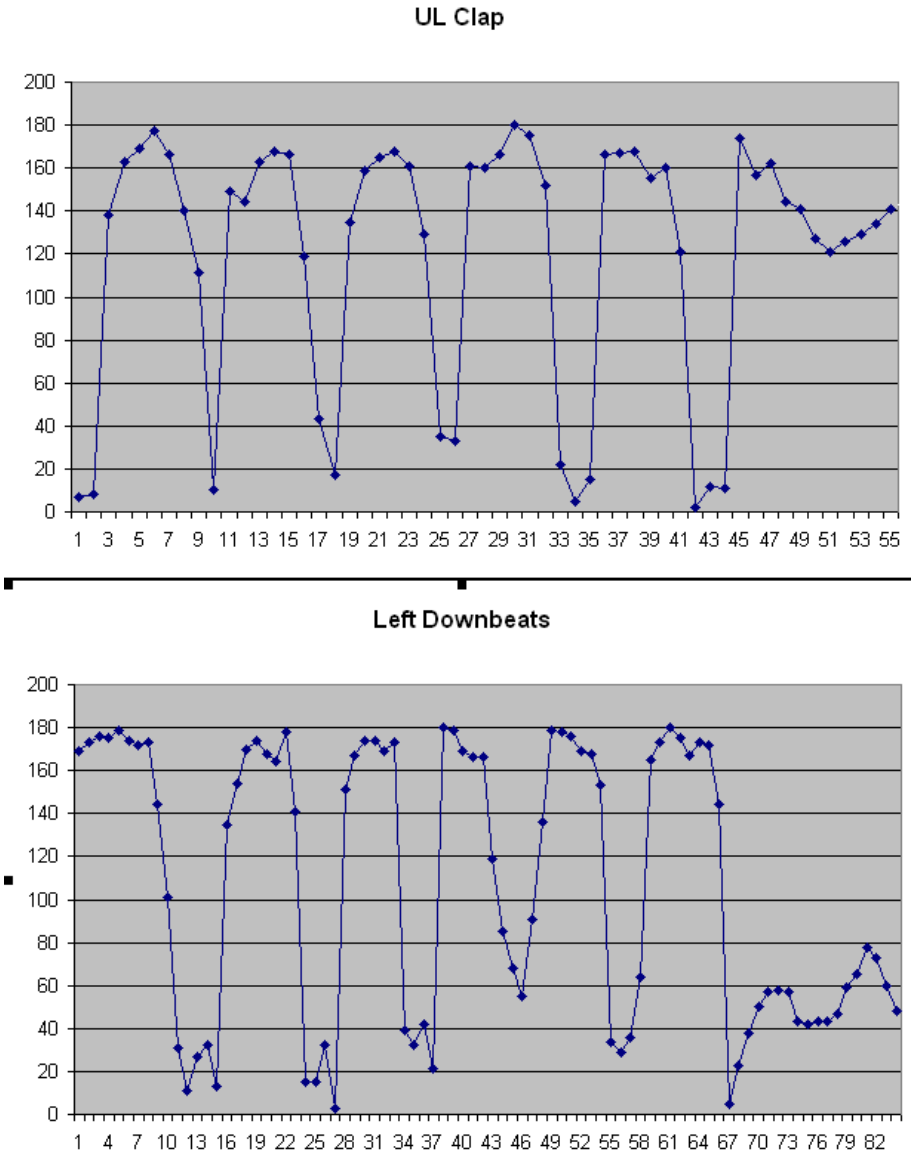

Fig. 13. Angle of motion over time shows a sinusoidal pattern for the left hand clapping at top, and the left hand waving downbeats at bottom

flow directions (rotated to span −180 to +180 degrees) and the position of the centroid of the flow regions relative to the center of mass of the silhouette normalized by the size of the bounding box of the foreground silhouette. Twenty subjects did gestures each. The training and testing sets consisted of 15 and 5 subjects, respectively. We used a 3-state HMM to model each gesture. A recognition rate of 96% overall was obtained.

No useful result could be obtained from the three other optical flow methods because the inexpensive low-resolution camera used could not resolve enough texture in the dark scene to enable tracking (see Figs. 4 and 11). This leads us to a discussion of the utility of the various optical flow algorithms. The tHMI normal flow approach excels in environments where one can create a robust foreground–background segmentation necessary to build the MHI. There are many techniques for doing this segmentation such as discussed in the beginning of Sect. 3. Arcade games are one good example of an environment where one can contrive to extract reliable foreground silhouettes. The other three optical flow methods are more problematic in an environment such as an arcade because users might wear clothing with little texture. In environments where one cannot guarantee good segmentation (e.g. parking lots), or if the goal is to extract structure from motion, then optical flow methods are preferred. All the flow methods depend critically on running at frame rates fast enough to capture the motions of interest; recognition and geometric reconstruction improve in general at higher frame rates. If CPU resources are at a premium, the tHMI approach imposes the lowest computational load.

8 Summary

In this paper we extended earlier motion template research [17] by offering a method of calculating normal optical flow motion orientations directly from the MHI. We also presented a novel method of motion segmentation based on segmenting layered motion regions that are meaningfully connected to movements of the object of interest. This motion segmentation, together with silhouette pose recognition, pro-

vides a very general and useful tool for gesture and motion recognition. In addition, this new algorithm is computationally faster than other motion segmentation algorithms based on optical flow.

References

1. Aggarwal J, Cai Q (1997) Human motion analysis: a review. In: IEEE Nonrigid and Articulated Motion Workshop, pp 90–102
2. Bergen J, Anandan P, Hanna K, Hingorani R (1992) Hierarchical model-based motion estimation. In: Proceedings of the European Conference on Computer Vision, pp 237–252
3. Beymer D, Konolige K (1999) Real-time tracking of multiple people using stereo. In: IEEE FRAME-RATE Workshop, http://www.eecs.lehigh.edu/FRAME/
4. Bierling, M (1988) Displacement estimation by hierarchical blockmatching. Proceedings of SPIE Conference Visual Communications and Image Processing, vol 1001, pp 942–951
5. Black M, Anandan P (1993) A framework for robust estimation of optical flow. In: Proceedings International Conference Computer Vision, pp 231–236
6. Black M, Yacoob Y (1995) Tracking and recognizing rigid and nonrigid facial motions using local parametric model of image motion. In: Proceedings International Conference Computer Vision, pp 374–381
7. Bobick A, Davis J, Intille S (1997) The KidsRoom: an example application using a deep perceptual interface. In: Proceedings Perceptual User Interfaces, pp 1–4, October
8. Bradski G, Yeo B-L, Yeung M (1999) Gesture for video content navigation. In: SPIE'99, 3656-24 S6
9. Bradski G (1998) Computer vision face tracking for use in a perceptual user interface. Intel Technology Journal, http://developer.intel.com/technology/itj/q21998/articles/art_2.htm, Q2
10. Bregler C (1997) Learning and recognizing human dynamics in video sequences. In: Proceedings Computer Vision And Pattern Recognition, pp 568–574, June
11. Cham T, Rehg J (1998) A multiple hypothesis approach to figure tracking. In: Proc. Perceptual User Interfaces, pp 19–24, November
12. Cutler R, Davis L (2000) Robust real-time periodic motion detection, analysis, and applications. IEEE Trans Pattern Anal Mach Intel 22(8):781–796
13. Cutler R, Turk M (1998) View-based interpretation of real-time optical flow for gesture recognition. In: International Conference On Automatic Face and Gesture Recognition, pp 416–421
14. Davis J (1999) Recognizing movement using motion histograms. In: MIT Media Lab Technical Report #487, March
15. Davis J, Bobick A (1998) Virtual PAT: a virtual personal aerobics trainer. In: Proc. Perceptual User Interfaces, pp 13–18, November
16. Davis J, Bobick A (1998) A robust human-silhouette extraction technique for interactive virtual environments. In: Proceedings Modelling and Motion Capture Techniques for Virtual Environments, pp 12–25
17. Davis J, Bobick A (1997). The representation and recognition of human movement using temporal templates. In: Proceedings Computer Vision and Pattern Recognition, pp 928–934, June
18. Elgammal A, Harwood D, Davis L (1999) Non-parametric model for background subtraction. In: IEEE FRAME-RATE Workshop, http://www.eecs.lehigh.edu/FRAME/
19. Essa I, Pentland A (1995) Facial expression recognition using a dynamic model and motion energy. In: Proceedings International Conference on Computer Vision pp 260–367
20. Freeman W, Anderson D, Beardsley P, et al. (1998) Computer vision for interactive computer graphics. IEEE Comput Graph Appl 18(3):42–53
21. Gavrila D (1999) The visual analysis of human movement: a survey. Comput Vision Image Understanding 73(1):82–98
22. Haritaoglu I, Harwood D, Davis L (1998). W4S: A real-time system for detecting and tracking people in 2 1/2 D. In: European Conference On Computer Vision, pp 877–892
23. Horn BKP, Schunck BG (1981) Determining optical flow. Artif Intel 17:185–203
24. Horprasert T, Harwood D, Davis L (1999) A statistical approach for real-time robust background subtraction and shadow detection. In: IEEE FRAME-RATE Workshop, http://www.eecs.lehigh.edu/FRAME/
25. Hu M (1962) Visual pattern recognition by moment invariants. IRE Trans Inf Theory IT-8(2):
26. Jain R, Nagel H (1979) On the analysis of accumulative difference pictures from image sequences of real world scenes. IEEE Trans Pattern Anal Mach Intel 1(2):
27. Krueger M (1991). Artificial reality II. Addison-Wesley, Boston
28. Lukas BD, Kanade T (1981) An iterative image registration technique with an application to stereo vision. In: International Joint Conference on Artificial Intelligence, pp 674–679
29. Madabhushi A, Aggarwal J (1999) A Bayesian approach to human activity recognition. In: Proceedings of IEEE Workshop on Visual Surveillance, pp 25–32
30. Martins F, Nickerson B, Bostrom V, Hazra R (1999) Implementation of a real-time foreground/background segmentation system on the Intel architecture. In: IEEE FRAME-RATE Workshop, http://www.eecs.lehigh.edu/FRAME/
31. Moeslund T (1999) Summaries of 107 computer vision-based human motion capture papers. University of Aalborg Technical Report LIA 99-01, March
32. Moeslund T (1999) Computer vision-based human motion capture – a survey. University of Aalborg Technical Report LIA 99-02, March
33. 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/opencv
34. Pinhanez C (1999) Representation and recognition of action in interactive spaces. PhD Thesis, MIT Media Lab
35. Polana R, Nelson R (1994) Low level recognition of human motion. In: Workshop on Motion of Nonrigid and Articulated Objects. pp 77–82
36. Rao C, Shah M (2000) A view-invariant representation of human action. Proc. In: International Conference on Control, Automation, Robotics and Vision
37. Shah M, Jain R (1997) Motion-based recognition. Kluwer Academic
38. Starner T, Pentland A (1995) Visual recognition of American sign language using hidden Markov models. In: Proceedings of International Workshop on Automatic Face and Gesture Recognition
39. Therrien C (1989) Decision estimation and classification. Wiley, New York
40. Wilson A, Bobick A (1999). Parametric hidden markov models for gesture recognition. IEEE Trans Pattern Anal Mach Intel 21(9):884–900
41. Yacoob Y, Black MJ (1997) Parameterized modeling and recognition of activities. Comput Vision Image Understanding 73(2):232–247
42. Yamato J, Ohya J, Ishii K (1992) Recognizing human action in time-sequential images using hidden Markov model. In: Proceedings of Computer Vision and Pattern Recognition, pp 379-385
43. Assembly optimized performance libraries in Image Processing, Signal Processing, JPEG and Matrix Math can be downloaded from http://developer.intel.com/utune/perflibst/

Gary R. Bradski graduated from U.C. Berkeley and after seven years of traveling, working and consulting went back to graduate school to get a Ph.D. (1994) from the Department of Cognitive and Neural System at Boston University studying mathematical models of biological memory and vision. He now manages a research group at Intel Labs working in computer vision, compuer graphics and probabilistic graphical models. Among the group's outputs is the Open Source computer Vision Library (http://www.intel.com/research/mrl/research/opencv). His current research interests are now focused on probabilistic graphical models with applications to vision and Intel's manufacturing processes.

James W. Davis received the MS (1996) and PhD (2000) degrees from Massachusetts Institute of Technology Media Laboratory specializing in computational vision. He is currently an Assistant Professor of Computer and Information Science and a Member of the Center for Cognitive Science at Ohio State University. Prof. Davis' research interests include the representation and computer recognition of human movements, human–computer interactive systems, motion capture technology, and human perception. The current emphasis of his work is a computational approach to motion categorization from limited visual input.