View-based Interpretation of Real-time Optical Flow for Gesture Recognition

Ross Cutler
Univerity of Maryland
College Park, Maryland
rgc@cs.umd.edu

Matthew Turk
Microsoft Research
Redmond, Washington
mturk@microsoft.com

Abstract

We have developed a real-time, view-based gesture recognition system. Optical flow is estimated and segmented into motion blobs. Gestures are recognized using a rule-based technique based on characteristics of the motion blobs such as relative motion and size. Parameters of the gesture (e.g., frequency) are then estimated using context specific techniques. The system has been applied to create an interactive environment for children.

1 Introduction

For many applications, the use of hand and body gestures is an attractive alternative to the cumbersome interface devices for human-computer interaction. This is especially true for interacting in virtual reality environments, where the user is no longer confined to the desktop and should be able to move around freely. While special devices can be worn to achieve these goals, these can be expensive and unwieldy. There has been a recent surge in computer vision research to provide a solution that doesn't use such devices.

This paper describes a real-time vision-based gesture recognition system. Taking a bottom-up, view-based approach, rather than reconstruct the body, we use optical flow to segment the dominant moving body parts (e.g., hands, arms) into "motion blobs." Given the motion blobs and their general characteristics as input, actions are recognized using a rule-based approach. Once an action is detected, parameters of the action (e.g., frequency) are estimated until the action stops.

1.1 Application

Personal computers are widely used as educational tools in grades K-12 and higher. However, for children under the age of five, the keyboard and mouse can be an obstacle to interacting with the computer. We applied our gesture recognition system to create an environment that allows the child to interact with the computer using hand and body gestures. These gestures are used in playing "Simon Says" (e.g., wave your arms like a bird, jump up and down); conducting music (e.g., clapping to create sounds, waving to set a tempo to conduct music); and walking through a virtual world.

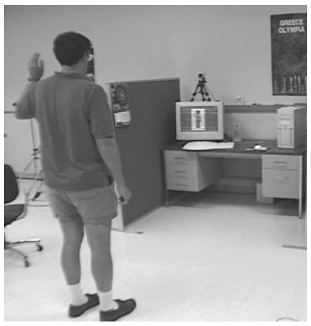

Figure 1. System

Figure 1 shows the system. A standard dual-processor 300 MHz Pentium II PC is used with a Sony DS-250 digital video camera.

2 Related Work

Research in motion-based recognition has greatly increased in recent years (see [10] for a recent survey). We will describe the most relevant work below.

Becker [6] developed a real-time system to recognize five basic T'ai Chi gestures. The hands and face are tracked using a stereo camera system that tracks skin blobs [3]. The 3D velocity of the hands is used as input to a Hidden Markov Model (HMM) based system. While users can perform each gesture with a different total duration, the velocity of the hands is assumed to be approximately the same across users.

Davis and Bobick [12] use a view-based, bottom-up approach to real-time gesture recognition. They construct a binary motion-energy image which represents where motion has occurred in an image sequence, and a motion-history image, which is a scalar-valued image where intensity is a function of recency of motion. These view-specific templates are matched against stored models of views of known actions (using moments to provide invariance to scale and translation). Actions are recognized in real-time, but are only invariant to linear changes in speed. This system has been applied to an interactive play-space for children [8], and to an interactive aerobics trainer [11] (in which they note difficulty in synchronizing the actions with the music).

Black and Yacoob [7] use a rule-based reasoning system to recognize facial expressions of people. The input to the reasoning system are actions such as "inward lowering of brows" and "mouth contracting." These inputs are determined by tracking patches on the eyes, brows, and mouth.

Wilson and Bobick [20] address the problem of recognizing gestures that exhibit meaningful variations. For example, a pointing gesture requires not only the gesture to be recognized, but the estimated pointing direction as well. They extend the standard HMM by including a global parametric variation in the output probabilities of the states of the HMM. EM is used to train the parametric HMM.

3 Optical Flow Estimation

An optical flow algorithm estimates the 2D flow field from image intensities. While many techniques have been developed, accurate and dense estimates are difficult to achieve (see [5] for a recent survey). There are four general categories for optical flow algorithms: differential (gradient), region-based matching, energy-based, and phased-based. Differential techniques compute velocity from spatio-temporal derivatives (or a filtered version of the image). Region-based techniques compare patches of the image (or filtered image) at different disparities to determine the flow. Energy and phase-based methods apply velocity-tuned filters on the image sequence and extract the velocities from the filter's output. In a comparison of these four techniques, Barron et al [4] found the phase-based approach of Fleet and Jepson [14] to be the most accurate; unfortunately, it is also the slowest [16].

For view-based gesture recognition, our requirements for an optical flow algorithm are as follows:

- produces dense flow
- accuracy: direction within $25°$, magnitude within 50%
- able to work on large disparities (e.g., 15 pixels)
- tolerant to noise
- usable on interlaced and decimated images
- efficient; parallelizable; real-time (30 Hz) on 160x120 images
- functional in low-contrast environments

Dense flow is desirable since it has redundancy in data that allows for better performance and robustness. Note that the required accuracy is not large. This is mainly a tradeoff for speed, but we have found that these requirements are sufficient for our segmentation algorithm and recognition analysis.

We make the following assumptions about our environment: (1) the background is relatively static, and (2) the illumination changes slowly.

As noted in [4], region-based matching techniques (e.g., correlation-based) are more robust to noise than differential techniques. In addition, region-based techniques work well even when the input is interlaced or decimated. Camus [9] notes that correlation-based techniques are also less sensitive to illumination changes between frames than differential based techniques.

Computing optical flow in real-time has traditionally been done using expensive hardware (e.g., DSPs or field-programmable gate arrays). However, general purpose CPUs are now becoming available with Single-Instruction, Multiple Data (SIMD) instructions (e.g., Intel Pentium II, Sun UltraSparc). For this system, we used a 266 MHz Intel Pentium II PC. Our MMX implementation of the flow algorithm produced a 400% increase in speed compared to non-MMX optimized code, which was critical in making the system real-time.

We chose to use a correlation-based algorithm to estimate the optical flow, using the sum of absolute differences (SAD) instead of correlation for efficiency reasons. A major drawback to correlation-based algorithms is that they can be computationally expensive. Consider an image of size $I \times I$, a patch radius of P, and a search radius of R. We define a similarity metric D on the images I_1 and I_2 for the pixel (x, y) and disparity (dx, dy) as:

$$D(x, y, dx, dy) = \sum_{i=-P}^{P} \sum_{j=-P}^{P} E$$

where

$$E = \mid I_1(x+i, y+j) - I_2(x+i+dx, y+j+dy) \mid .$$

To determine the flow between images I_1 and I_2 at a point (x, y) we minimize $D(x, y, dx, dy)$ for $(dx, dy) \in [-R, R] \times [-R, R]$. A straightforward implementation of this algorithm would require $O(I^2 P^2 R^2)$ pixel comparisons. However, there are redundant comparisons in this technique, and by storing intermediate results we can reduce the complexity to $O(I^2 R^2)$ [13]. Note that although we have reduced the computational complexity, we have increased the required memory bandwidth, as intermediate results must be stored in memory. For this technique to give an overall time savings, we must have a fast memory system. Unfortunately the currently available Pentium II systems do not have sufficient memory bandwidth to make this technique feasible. That is, the memory bottleneck is so severe that it overrides the reduction in complexity, and so we instead implement the more computationally expensive algorithm.

In order to satisfy the requirement that our flow algorithm work well in low contrast conditions, we use color (in YUV space[1]) to define our similarity metric:

$$D'(x,y,dx,dy) = \sum_{i=-P}^{P} \sum_{j=-P}^{P} E_Y + E_U + E_V$$

where

$$E_C = \mid I_{1C}(x+i, y+j) - I_{2C}(x+i+dx, y+j+dy) \mid .$$

We found that using YUV color in this way gave significantly better flow estimates than simply using the grayscale image data. In particular, color helps to distinguish the hand in front of the face, even though both objects have similar grayscale values.

3.1 Motion Detection

If we assume that the background is relatively static, then we can reduce the optical flow computation time by only computing the flow for pixels that satisfy a motion detection criterion. Specifically, we compute the temporal derivative of a Gaussian smoothed image I_t^* and threshold it by the value $K\sigma$, where σ is the average standard deviation of the white noise in the video system for the YUV color channels (measured off-line; for the DS-250 $\sigma = 2.5$), and K is a constant (typically 8). We compute the flow at pixel (x, y) only if $M(x, y) = 1$, where

$$M(x,y,t) = \begin{cases} 1 & \text{if } E_Y^* + E_U^* + E_V^* > K\sigma \\ 0 & \text{otherwise} \end{cases}$$

where

$$E_C^* = \mid I_{t,C}^*(x,y) - I_{t+1,C}^*(x,y) \mid .$$

The Gaussian smoothing serves two purposes: first, it reduces the noise inherent in the images. Second, it gives a support patch from which we compare the change in motion (that is, we are comparing the motion for a patch instead of a single pixel).

$M(x, y, t)$ can be improved by using three sequential images instead of two in the following way:

$$M'(x,y,t) = M(x,y,t-1) \wedge M(x,y,t)$$

[1]The YUV color space was used since the Sony DS-250 camera outputs YUV:411 data. The RGB color space was found to give similar results.

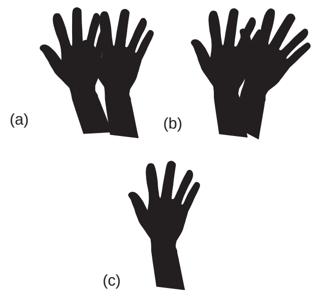

Figure 2. Mask applied to a hand moving left to right. Let A,B, and C be the hand at time $t-1$, t, and $t+1$, respectively. (a) $M(x,y,t-1) = A \vee B$); (b) $M(x,y,t) = B \vee C$; (c) $M'(x,y,t) = B$

While $M(x, y, t) = 1$ if pixel (x, y) is moving in image I_t or I_{t+1}, $M'(x, y, t) = 1$ only if (x, y) is moving in I_t (see Figure 2). This further reduces the number of pixels to compute the flow on. Moreover, it also eliminates the problem of computing the flow of a pixel (x, y) in I_t which is occluded in I_{t+1} (e.g., (x, y) is in the background near the leading edge of a moving object).

3.2 Flow Estimate Confidence Measures

There are many ways to estimate the confidence in the flow estimate. For example, Anandan [1] fits a cubic surface to the SSD values and defines a confidence measure based on the curvatures of this surface. Another way to verify that the flow is correct is to use the left-right check [13]; that is, compute the flow for image I_t with respect to I_{t+1}, then repeat for I_{t+1} with respect to I_t. Only the flow estimates that are the same (or nearly so) should be used.

While both these techniques give good results, they are also computationally expensive. Instead, we simply rely on the motion mask $M'(x, y, t)$ to eliminate the two most common flow estimate errors, namely occlusion boundaries and low texture areas. We find this technique to give sufficiently good flow estimates for our later analysis.

3.3 Search Pattern

To determine the flow between images I_1 and I_2 at a point (x, y) we minimize $D'(x, y, dx, dy)$ for $(dx, dy) \in [-R, R] \times [-R, R]$. However, Ancona and Poggio [2] shows that the 1D search pattern $(dx, dy) \in [-R, R] \times \{0\} \cup \{0\} \times [-R, R]$ gives a reasonable approximation to the full 2D search pattern. The advantage to the 1D search space is it is that much less computationally expensive. We utilize this in our algorithm, adding the diagonals to the search pattern (see Figure 3). Our tests show that this 1D

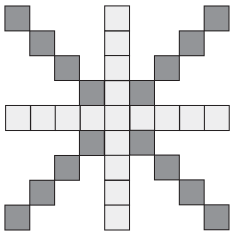

Figure 3. Search pattern

search pattern gives good results with our data sets for disparities ≤ 15 pixels.

4 Optical Flow Segmentation

In order to analyze the optical flow at a higher level, we first segment it into "motion blobs." There are many ways to segment optical flow (e.g., [15], [19], [18]). However, we are constrained in choosing a technique that works with inaccurate flow and will run in real-time. In addition, the objects we are tracking are deformable, and unlike rigid objects, the flow cannot be easily modeled. Our problem is simplified since we are interested only in the dominant motion, and the nearly static background ensures that these motions will belong to the user. In addition, we assume that the motion blobs do not overlap.

Our segmentation algorithm is similar to region growing. A random flow vector is selected and added to a new cluster. All nearby flow vectors with similar directions to the cluster's average direction are added to the cluster, continuing recursively until there are no nearby flow vectors to add, at which point the process repeats with a new cluster. The flow magnitude is not used in the segmentation, since our tests show that the magnitude can vary considerably within a motion blob, while the direction is relatively robust.

A blob is modeled as an ellipse; the major axis is determined by the best fitting line though the origins of the blob's flow vectors using a robust L_1 line fitting algorithm [17]. The major axis length is twice the median distance from the blob's centroid to the each flow vector (projected on the major axis); the minor axis length is similarly determined.

See Figures 4 and 5 for examples of the segmented flow; the flow vectors are averaged for display purposes.

5 Gesture Recognition and Parameter Estimation

Our view-based approach to gesture recognition uses a rule-based technique to identify an action based on a set of conditions. The following information about the motion blobs are used as input to the action predicates: the number of blobs; the absolute motion of the blobs (horizontal, vertical, or rotational); the relative motion between blobs (opposing or same direction); the relative size of the blob(s) (the blob's major axis length divided by the user's height (in pixels)); and the relative positions for the blobs (e.g., the clapping blobs must have close $\bar{y}$ values). The conditions for the six actions in the interactive environment are given in Table 1. These conditions must be satisfied over N consecutive frames, where N is typically about 10; for robustness, some misses (one or two frames) are allowed. Figure 6 shows the ideal motion blobs for these actions.

A blob's absolute motion is determined by analyzing the history of the origin and major axis direction. Specifically,

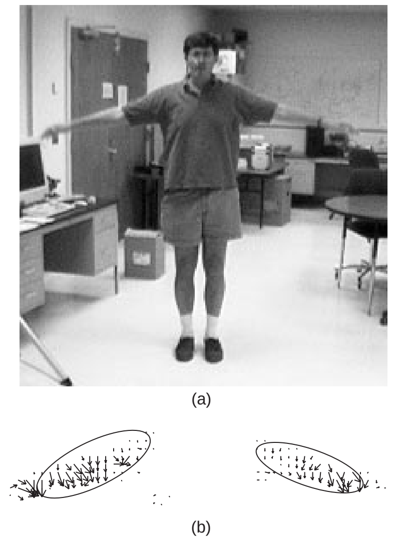

Figure 4. (a) Flapping action; (b) Flapping flow and segmented motion blobs

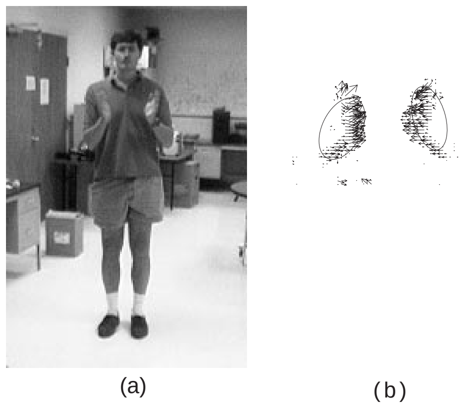

Figure 5. (a) Clapping action; (b) Clapping flow and segmented motion blobs

Action	# of Blobs	Abs. Motion	Relative Motion	Size
waving	1	rot.	NA	0.2
jumping	1	vert.	NA	0.8
clapping	2	horiz.	oppose	0.2
drumming	2	vert.	oppose	0.4
flapping	2	rot.	same	0.8
marching	2	vert.	oppose	0.4

Table 1. Action conditions

Action	Events	Parameters
waving	left/right beat	frequency, velocity
jumping	up/down	height, velocity
clapping	contact	frequency, velocity
drumming	left/right beat	frequency, velocity
flapping	up/down	frequency, velocity
marching	L/R foot up/down	frequency, velocity

Table 2. Action events and parameters

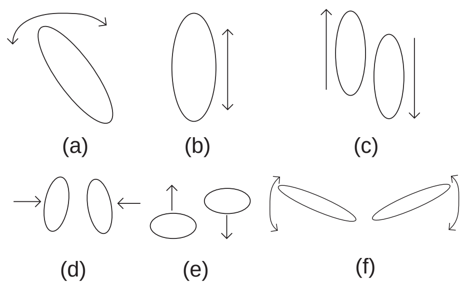

Figure 6. Action motion blobs: (a) waving, (b) jumping, (c) marching, (d) clapping, (e) drumming, (f) flapping

horizontal and vertical motions have roughly constant horizontal and vertical origin components; rotational motions have a roughly monotonic major axis direction.

Since we are interested in only the dominant motions, only the largest motion blobs are used for gesture recognition. The sum of all motion vectors within a blob is used to determine whether to use the both in the gesture recognition.

Once an action is recognized, the system will go into a mode specific to that action, estimating the action's parameters until the action ceases. For example, the waving action, which consists of the user waving his hand (as in conducting music), estimates the frequency of movement (i.e. the tempo). The frequency is determined by detecting the zero-crossings of the average flow magnitude for

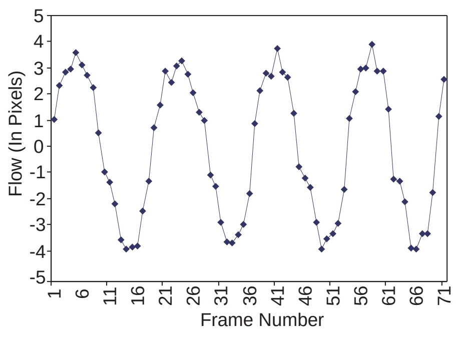

Figure 7. Waving average flow magnitude

the motion blob, which is shown in Figure 7. These zero-crossings correspond to the left- and right-most movements in the waving action, and are labeled as the left and right beat. The time intervals between the left and right beat are averaged over several intervals and inverted to obtain the frequency. This frequency, along with the left and right beat and the maximum average flow (over a period of time) can be used as input to control an interactive environment. The events and parameters for the other actions (see Table 2) are computed in a similar fashion.

6 Results

In order to test our gesture recognition system, we have developed an interactive environment for children. The opening screen for this system displays seven animated characters performing the seven actions in Table 1. If the child wants to conduct music, he or she mimics the animated character that is conducting music. Once the action is recognized, the system goes into a conducting mode, shown in Figure 8. In this mode, the user can set the tempo of a song by the frequency of the hand; the left and right beats can be used to progress (incrementally) through the notes in a song. After the user stops waving, the starting screen returns to offer more activities from which to choose.

We have informally tested this system with both children and adults. Participants have found it to be fun, intuitive, and compelling. The immediate feedback of the musical sounds and animated characters is engaging, especially for children. (After using it at the lab, the young son of one of the authors couldn't understand why his home computer wouldn't do the same thing for him!) No special clothing is required to use the system (non-textured clothes work fine). People can move the camera, change the lighting conditions, move around somewhat while gesturing, and allow others to be in the scene (if they're not moving much).

Table 3 gives the execution times of each component in the system (per image frame). The image flow and segmentation times depend on the amount of movement in the

Figure 8. Conducting mode

Component	Time (ms)
Gaussian smooth	6
subtract/threshold	3
image flow	15
segmentation	5
gesture recognition	1

Table 3. Component execution times

image; the times given here are typical. These execution times are for a single CPU; the other CPU in our system is used primarily for the user interface (animation, music).

7 Conclusions

We have used real-time optical flow to segment a user's dominant motions. Using a rule-based technique and general characteristics of the motion blobs, we can recognize among a set of gestures. The parameters for each gesture are estimated in a context specific manner. The system has been successfully applied to an interactive system for children. Future work for this system includes: adding more gestures (e.g., pointing) and allowing simultaneous gestures (e.g., clapping while marching); using fuzzy logic for the reasoning system to make the gesture recognition more robust; enhancing the motion segmentation to allow intersecting motion blobs; and adding the ability to handle multiple users in the field of view, with associated gestures for interaction (e.g., shaking hands).

References

[1] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–310, 1989.

[2] N. Ancona and T. Poggio. Optical flow from 1D correlation: Application to a simple time-to-crash detector. Technical Report AI Memo 1375, MIT AI Lab, 1993.

[3] A. Azarbayejani and A. Pentland. Real-time self-calibrating stereo person tracking using 3-D shape estimation from blob features. Technical Report 363, MIT Media Lab Perceptual Computer Section, 1996.

[4] J. Barron, D. Fleet, S. Beauchemin, and T. Burkitt. Performance of optical flow techniques. In *Proceedings of the Computer Vision and Pattern Recognition*, pages 236–242, 1992.

[5] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–467, 1995.

[6] D. A. Becker. Sensei: A real-time recognition, feedback and training system for T'ai Chi gestures. Master's thesis, MIT Media Lab Perceptual Computer Section, 1997.

[7] M. Black and Y. Yacoob. Tracking and recognizing facial expressions in image sequences, using local parameterized models of image motion. Technical Report CAR-TR-756, University of Maryland, College Park, 1995.

[8] A. Bobick and et al. The kidsroom: A perceptually-based interactive and immersive story environment. Technical Report 398, MIT Media Lab Perceptual Computer Section, 1996.

[9] T. Camus. Real-time quantized optical flow. *Journal of Real-Time Imaging*, 3:71–86, 1997.

[10] C. Cedras and M. Shah. Motion-based recognition: A survey. *Image and Vision Computing*, 13(2):129–155, 1995.

[11] J. Davis and A. Bobick. Virtual pat: A virtual personal aerobics trainer. Technical Report 436, MIT Media Laboratory Vision and Modeling Group, 1997.

[12] J. W. Davis and A. F. Bobick. The representation and recognition of action using temporal templates. Technical Report 402, MIT Media Lab Perceptual Computer Section, 1996.

[13] O. Faugeras and et al. Real time correlation based stereo: algorithm, implementations and applications. Technical Report 2013, INRIA, 1993.

[14] D. Fleet. *Measurement of Image Velocity*. Kluwer Academic Publisher, Norwell, 1992.

[15] A. Jepson and M. Black. Mixture models for optical flow computation. Technical Report RBCV-TR-93-44, University of Toronto, 1993.

[16] H. Liu, T.-H. Hong, M. Herman, and R. Chellappa. Accuracy vs. efficiency trade-offs in optical flow algorithms. In *European Conference on Computer Vision*, pages 174–183, 1996.

[17] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1988.

[18] H. Sawhney and S. Ayer. Compact representation of videos through dominant and multiple motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):814–830, 1996.

[19] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. Technical Report 279, MIT Media Lab Perceptual Computer Section, 1994.

[20] A. Wilson and A. Bobick. Recognition and interpretation of parametric gesture. Technical Report 421, MIT Media Lab Perceptual Computer Section, 1997.