Visual gesture recognition

J. Davis
M. Shah

Indexing terms: Motion-based recognition, Motion tracking, Gesture interpretation

Abstract: This paper presents a method for recognising human-hand gestures using a model based approach. A finite state machine is used to model four qualitatively distinct phases of a generic gesture. Fingertips are tracked in multiple frames to compute motion trajectories. The trajectories are then used for finding the start and stop position of the gesture. Gestures are represented as a list of vectors and are then matched to stored gesture vector models using table lookup based on vector displacements. Results are presented showing recognition of seven gestures using images sampled at 4 Hz on a SPARC-1 without any special hardware. The seven gestures are representatives for actions of left, right, up, down, grab, rotate, and stop.

1 Introduction

It is essential for computer systems to possess the ability to recognise meaningful gestures if computers are to interact naturally with people. Humans use gestures in daily life as a means of communication (e.g. pointing to an object to bring someone's attention to the object, waving 'hello' to a friend, requesting *n* of something by raising *n* fingers etc). The best example of communication through gestures is given by sign language. American sign language (ASL) incorporates the entire English alphabet along with many gestures representing words and phrases [3], which permits people to exchange information in a nonverbal manner.

Currently, the human–computer interface is through a keyboard and/or mouse. Physically challenged people may have difficulties with such input devices and may require a new means of entering commands or data into the computer. Gesture, speech, and touch inputs are few possible means of addressing such users' needs to solve this problem. Using computer vision, a computer can recognise and perform the user's gesture command, thus alleviating the need for a keyboard. Applications for such a vision system are the remote control of a robotic arm, guiding a computer presentation system, and executing computer operational commands such as opening a window or programme.

The method described in this paper presents a computer vision gesture recognition method which permits human users adorned with a specially marked glove to command a computer system to carry out predefined gesture action commands. Additionally, a subset of the gestures is composed of select ASL letters. Each gesture begins with the hand in the 'hello' position and ends in the recognisable gesture position. The current library of gestures contains seven gestures: left, right, up, down, rotate, grab, and stop (see Fig. 1).

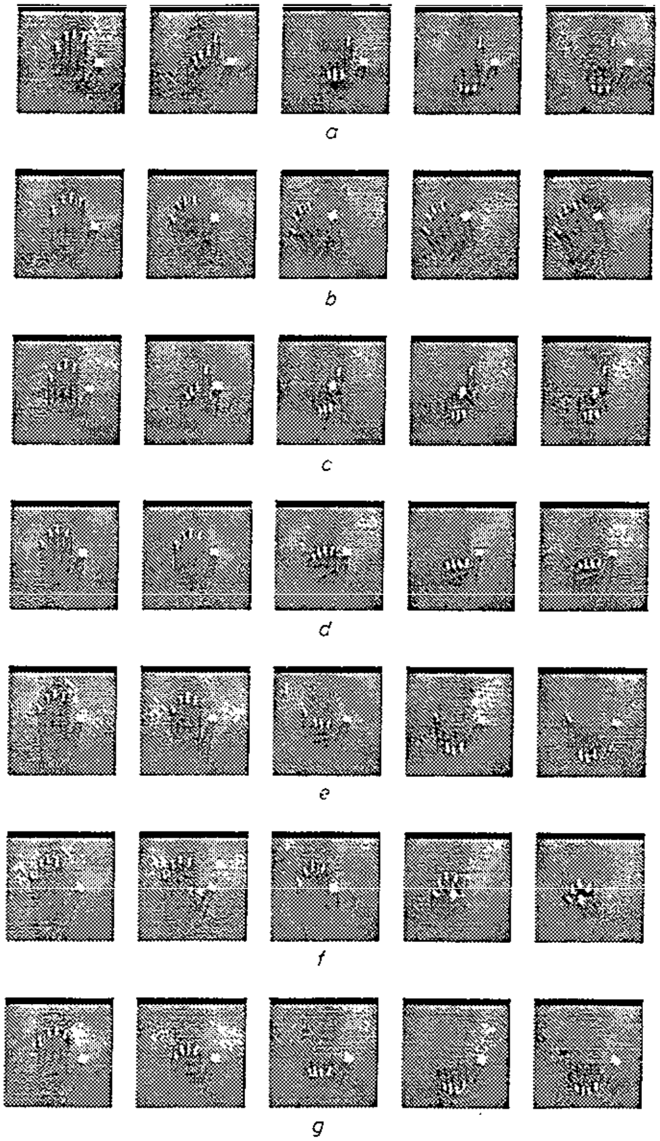
a
b
c
d
e
f
g

Fig. 1 *Gestures*
a left *b* right *c* up *d* down *e* rotate *f* grab *g* stop

Paper 1058K (E4), received 22nd October 1993

The authors are with the Computer Vision Laboratory, University of Central Florida, Orlando, FL 32816, USA

The research reported here was supported by the National Science Foundation grants CDA-9200369 and IRI-9220768.

The left gesture, or ASL 'L', is performed by moving the fingers to the letter 'L' position, with the thumb and index finger in the upright position and the rest of the fingers down to the palm (see Fig. 1*a*). Right, or ASL 'B', is performed by rotating the hand (all finger) 45 degrees to the right (see Fig. 1*b*). For up, or ASL 'G', the fingers are moved to the '#1' position, resembling pointing upward (see Fig. 1*c*). The down gesture is made by moving the index finger through little finger down so they are parallel with the ground (see Fig. 1*d*). Rotate, or ASL 'Y', requires moving the index, middle, and ring finger down to palm (see Fig. 1*e*). Grab, or ASL 'C', is performed by moving fingers to a position which resembles grabbing a small object (see Fig. 1*f*). Finally, stop is similar to down except that all fingers but the thumb move down to the palm, instead of being parallel with the ground (see Fig. 1*g*).

Our system has been created to recognise a sequence of multiple gestures. The user must start in the designated start position upon initialisation of the system and is able to make gestures until the termination gesture (stop) is recognised by the system. There are several advantages of this system over other recognition methods. First, it uses inexpensive black-and-white video. Incorporating colour markers on a glove as interest points [1] requires costly colour imaging, whereas a binary marked glove, as used in this research, can be detected in low-cost black-and-white imaging. Secondly, a simple vision glove is employed (i.e. no mechanical glove with LEDs or bulky wires). Current gesture input devices require the user to be linked to the computer, reducing autonomy [6]. Vision input overcomes this problem. Thirdly, a duration parameter for gestures is incorporated. For example, if this recognition system were connected to a robotic arm and the user makes a gesture for left, the robotic arm would continue to move left until the user moves the hand from the gesture position back to the start position. Therefore the user can control the execution duration of the robotic arm. Finally, owing to finite state machine (FSM) implementation of a generic gesture, no warping of the image sequences is necessary.

2 Related work

Baudel and Beaudouin-Lafon [6] implemented a system for the remote control of computer-aided presentations using hand gestures. In this system, the user wears a VPL dataglove which is linked to the computer. The glove can measure the bending of fingers and the position and orientation of the hand in 3D space. The user issues commands for the presentation by pointing at a predefined active zone and then performing the gesture for the desired command. Gesture models include information pertaining to the start position, arm motion (dynamic phase), and stop position of the gesture. The command set includes such commands as next page, previous page, next chapter, previous chapter, table of contents, mark page, and highlight area. Two main types of errors that can occur with this system are system errors and user errors. System errors relate to the difficulties in identifying gestures that differ only in the dynamic phase, while user errors correspond to hesitations while issuing a command. With trained users, the recognition rate was 90–98%. This system does not use vision to recognise gestures, but instead uses a linked hardware system to track the hand and arm movements, which makes movement less natural for the user.

Cipolla, Okamoto, and Kuno [1] present a real-time structure-from-motion (SFM) method in which the 3D visual interpretation of hand gestures is used in a man–machine interface. A glove with coloured markers attached is used as input to the vision system. Movement of the hand results in motion between the images of the coloured markers. The authors use the parallax motion vector, divergence, curl, and deformation components of the affine transformation of an arbitrary triangle, with the coloured points at each vertex, to determine the projection of the axis of rotation, change in scale, and cyclotorsion. This information is then used to alter an image of a model. The information extracted from the coloured markers does not give the position of the entire hand (each finger), but provides a triangular plane for the SFM algorithm. The structure-from-motion method used here assumes rigid objects, which is not true in the case of hand gestures.

Fukumoto, Mase, and Suenaga [4] present a system called finger-pointer which recognises pointing actions and simple hand forms in real-time. The system uses stereo image sequences and does not require the operator to wear any special glove. It also requires no special image processing hardware. Using stereo images, their system uses the 3D location of fingers rather than the 2D location. The coordinates of the operator's fingertip and the direction it is pointing, are determined from the stereo images, and then a cursor is displayed in the target location on the opposing screen. The system is robust in that it is able to detect the pointing regardless of the operator's pointing style. Applications of this system can be similar to the gesture controlled computer-aided presentations of Baudel and Beaudouin-Lafon [6] and also can be used in a video browser with a VCR.

Darrell and Pentland [2] have also proposed a glove-free environment approach for gesture recognition. Objects are represented using sets of view models, and then are matched to stored gesture patterns using dynamic time warping. Each gesture is dynamically time warped to make it of the same length as the longest model. Matching is based upon the normalised correlation between the image and the set of 2D view models where the view models are composed of one or more example images of a view of an object. This method requires the use of special purpose hardware to achieve real-time performance, and uses grey level correlation which can be highly sensitive to noise. Also, their method was only tested in distinguishing between two gestures.

3 Generic gesture

For a system to recognise a sequence of gestures, it must be able to determine what state the user's hand is in (i.e. whether or not the hand is dormant, moving, or in gesture position). Our approach relies on the qualitatively distinct events (phases) in gestures, rather than on frame by frame correlation. Each gesture the user performs begins with the hand in the start position (all fingers upright, as if one was about to wave 'hello' to another person). Next, the user moves the fingers and/or entire hand to the gesture position. Once in position the system will attempt to recognise the gesture and, if the system is connected to a peripheral device (e.g. robotic arm), it will execute the gesture command until the hand begins moving back to the start position. The system will then wait for the next gesture to occur. Thus, the user is constrained to the following four phases for making a gesture.

(i) Keep hand still (fixed) in start position until motion to gesture begins.

(ii) Move fingers smoothly as hand moves to gesture position.

(iii) Keep hand in gesture position for desired duration of gesture command.

(iv) Move fingers smoothly as hand moves back to start position.

Since these four phases occur in a fixed order, a finite state machine (FSM) can be used to guide the flow and recognition of gestures based on the motion characteristics of the hand (see Fig. 2). A 1 or 0 in the state diagram represents motion or no motion, respectively, between two successive images. Due to undesirable motion, or lack of motion, a bound must be set to control premature advancement of one phase to the next. A three frame similarity constraint, which states that, 'at least three consecutive images must have the same motion property to advance to the next phase', was found to inhibit this premature phase advancement. Therefore, motion must be detected in at least three sequential images for gesture motion to begin. Similarly, there must be no motion in at least three sequential images for the hand to be found fixed. Notice that the machine requires two successive 1s or 0s, depending on the particular phase, from three sequential images to advance to the next phase, thus reflecting the three frame similarity constraint.

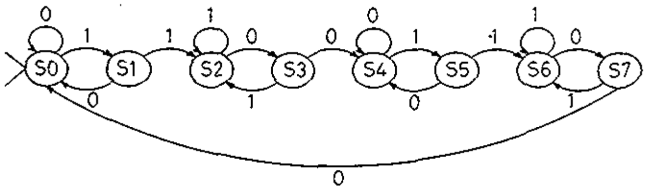

Fig. 2 *State diagram*

States S0 and S1 depict initial phase (phase 1), states S2 and S3 depict motion to gesture phase (phase 2), states S4 and S5 depict gesture recognition phase (phase 3) where an external device such as a robotic arm could execute the gesture until exiting this phase and states S6 and S7 depict motion to initial phase (phase 4)

Owing to the nature of this machine, no warping of image sequences is necessary (i.e. it is not required to have a fixed number of images for each gesture sequence). The FSM compensates for varying numbers of images by looping at the current phase as long as the three frame similarity constraint is satisfied. The actual number of frames which constitute the motion of a gesture yields no information for use with this system. The only useful information is the start and end position of the fingertips. The system does not care how the fingers or hand arrive in the gesture position; it wants to know the location of each fingertip before the gesture and when the gesture is made. Since the path of the fingertips to the gesture position is irrelevant, we need not perform any warping of the image sequence to match with models.

Only the locations and total displacement of the fingertips play a crucial role in gesture recognition, as compared to other motion characteristics such as instantaneous velocity. Therefore, we need only to track each fingertip from the initial position to the final gesture position. The FSM permits the determination of which phase the user is currently executing, and it also tracks the fingertips of a variable length frame sequence to the gesture position.

In our method, each image in the sequence is analysed to find the location of the fingertips (Section 4). If the hand is found to be in motion to gesture position, motion correspondence is used to track the points to the resulting gesture position (Section 5). Finally, the trajectories computed by the motion correspondence algorithm are converted to vector form to be matched with the stored gestures (Sections 6 and 7).

4 Fingertip detection

The goal of fingertip detection is to identify the 2D location of the marked fingertips on the vision glove. The location of the fingertips determines the position of the fingers at any time. Our point detection process for extracting the fingertip locations in each image is based upon histogram segmentation. Since we are using a *sequence of images in which the intensity of the fingertips* is known *a priori* to be significantly different from the remaining regions, a multimodal histogram of the image can be generated in which the fingertip regions correspond to the rightmost peak. A threshold can be established after smoothing (averaging) the histogram and finding the intensity value in the valley between the last two peaks (see Fig. 3*b*). Then any value greater than this threshold we shall treat belonging to a fingertip region, and any value less than this threshold will be discarded. Segmenting the image in this fashion results in a binary image where the only features are the fingertip regions (see Fig. 3*c*).

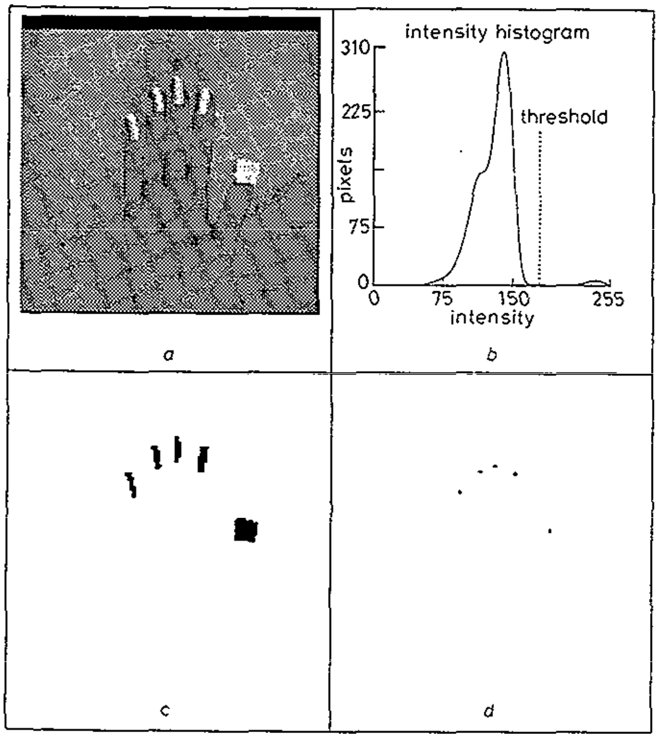

Fig. 3 *Point detection process*

a initial image

b smoothed histogram of entire image with threshold set at dashed line; rightmost peak corresponds to fingertip regions (brightest regions)

c binary image obtained from *a* by applying threshold determined from histogram in *b*

d centroids of five regions corresponding to five fingertips found in *c*

We prefer to have the five fingertip regions represented by five points for ease of calculations, storage, display, etc. A logical representation for a fingertip region is its centroid (see Fig. 3*d*). We use the centroid points in the motion correspondence stage.

5 Motion correspondence

Once the fingertips are detected in each frame, we need to compute trajectories using motion correspondence. Motion correspondence maps points in one image to points in the next image such that no two points are mapped onto the same point. Rangarajan and Shah's [5] motion correspondence algorithm was chosen for its exploitation of a proximal uniformity constraint, which says objects follow smooth paths and cover a small (proximal) distance in a small time. It was stated previously, in the phase 2 gesture constraint, that the fingers must move smoothly to the gesture position. Additionally, the three frame similarity constraint for motion, which requires at least three frames of motion, implies that the fingertips move a small (proximal) distance in each successive frame. Therefore, the algorithm, using a proximal uniformity constraint, agrees with the previously stated gesture motion constraints.

Using this algorithm, a path, known as a trajectory, is generated for each of the m points, starting with the points in the first image and ending with the points in the nth image. The resultant disjoint paths for each finger together is called the trajectory set [5].

Rangarajan and Shah's algorithm establishes correspondence among points by minimising a proximal uniformity function δ, which prefers the proximal uniform path, such that

$$\delta(X_p^{k-1}, X_q^k, X_r^{k+1}) = \frac{\|\overline{X_p^{k-1}X_q^k} - \overline{X_q^k X_r^{k+1}}\|}{\sum_{x=1}^{m}\sum_{z=1}^{m}\|\overline{X_x^{k-1}X^k_{\Phi^{k-1}(x)}} - \overline{X^k_{\Phi^{k-1}(x)}X_z^{k+1}}\|} + \frac{\|\overline{X_q^k X_r^{k+1}}\|}{\sum_{x=1}^{m}\sum_{z=1}^{m}\|\overline{X^k_{\Phi^{k-1}(x)}X_z^{k+1}}\|} \qquad (1)$$

where Φ^k is one to one onto correspondence between points of image k and image $k+1$, $1 \leqslant p, q, r \leqslant m$, $2 \leqslant k \leqslant m-1$, $q = \Phi^{k-1}(p)$, $\overline{X_q^k X_r^{k+1}}$ is the vector from point q in image k to point r in image $k+1$, and $\|X\|$ denotes the magnitude of vector X [5]. The first term in the equation represents the smoothness constraint and the second represents the proximity constraint.

The algorithm uses three frames to determine the correspondence. The authors assume the correspondence between frame 1 and frame 2 is known. They propose the use of optical flow to yield the initial correspondence between frame 1 and frame 2. It has been determined by studying gesture movements that the Euclidean ordering of the points for each fingertip should be the same in the first two frames (i.e. the coordinate ordering: farthest left to farthest right of the fingertips, from the thumb to the little finger, should be similar in both frames). Therefore, the initial correspondence can be derived from the location of the points. Rangarajan and Shah's correspondence algorithm is a noniterative greedy algorithm which keeps the overall proximal smoothness function minimised as much as possible in addition to being fair to each individual assignment [5].

6 Gesture modelling

In general, human finger movements are linear, with extrema moving from an extended position down to the palm/wrist area (e.g. from the hand in the 'hello' position to the hand making a fist). Even though we have the ability of limited rotational movement in the fingers, we mostly move the fingers up and down to the palm, with the thumb moving left and right over the palm. Since the fingers move relatively linearly (some move curvilinearly at times), we can approximate each fingertip trajectory by a single vector. Each vector will originate at the location of the corresponding fingertip before motion to the

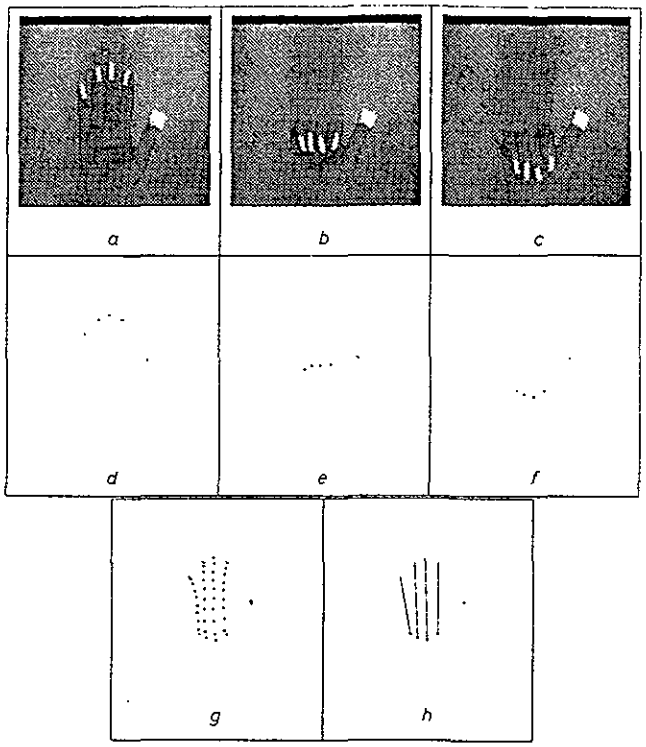

Fig. 4 *Vector extraction from gesture*
a–c image sequence: selected images shown
d–f respective fingertip points for images
g fingertip trajectories
h vector representation of trajectories

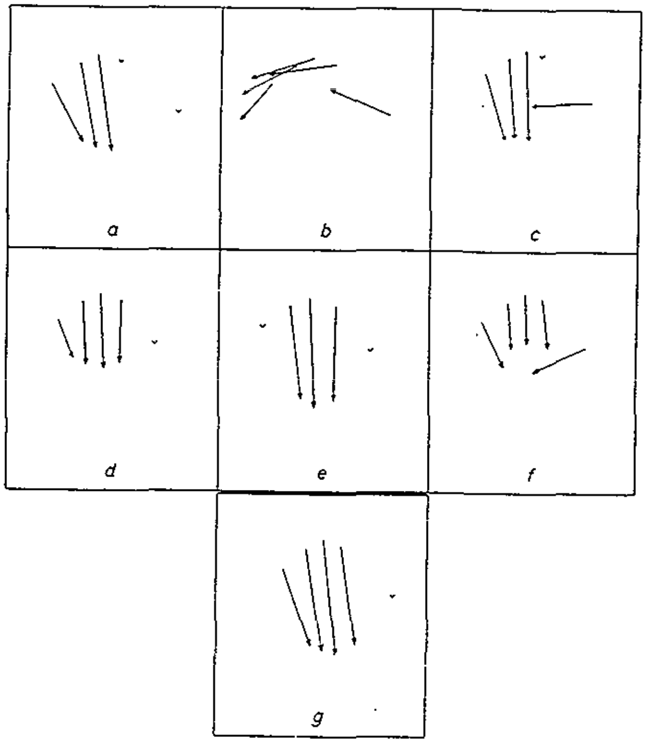

Fig. 5 *Vector representation of gestures*
a left *b* right *c* up *d* down *e* rotate *f* grab *g* stop

gesture position, and will terminate at the location in the gesture position. We disregard the actual path each fingertip makes because, as stated previously, we are concerned with only the beginning and ending location of each fingertip. Therefore, if there is some curvature to the fingertip trajectory, it will be disregarded. The motion information leading to the gesture position is not needed. Motion correspondence is used only to map the starting points to the ending points by means of tracking the points inbetween in the trajectories. The gesture can be determined by using these start and end points. See Fig. 5 for vector representations of the gesture set.

A library model is created from averaging m test models of the same gesture and is represented in a data structure (see Fig. 6) which contains

(i) The gesture name.

(ii) The mean direction and mean magnitude (i.e. mean displacement) for each fingertip vector.

(iii) The gesture's motion code.

gesture name;
finger 1 $\overline{Dir}$; finger 1 $\overline{Mag}$;
finger 2 $\overline{Dir}$; finger 2 $\overline{Mag}$;
finger 3 $\overline{Dir}$; finger 3 $\overline{Mag}$;
finger 4 $\overline{Dir}$; finger 4 $\overline{Mag}$;
finger 5 $\overline{Dir}$; finger 5 $\overline{Mag}$;
motion code;

Fig. 6 *Gesture model data structure fields*

The direction of a fingertip is determined from the starting point (x_0, y_0) and stopping point (x_n, y_n) in the trajectory. The direction Θ is easily calculated by

$$\Theta = \arctan \frac{y_n - y_0}{x_n - x_0} \qquad (2)$$

The magnitude (i.e. displacement of a fingertip) is also determined from the start and stop points in the trajectory set and is given by

$$\text{Disp} = \sqrt{[(x_n - x_0)^2 + (y_n - y_0)^2]} \qquad (3)$$

A motion threshold is necessary at this stage to account for shifts that can occur while the hand is in a relatively stable position, which would otherwise register as motion.

We use a five-bit motion code to store the motion activity of the five fingertips for the gesture and which also acts as a table key for the model. Each bit of this code corresponds to a specific fingertip vector, with the least significant bit storing finger 1's (thumb's) motion information and progressing to the most significant bit where finger 5's (little finger's) motion information is stored. A bit is set to 1 if its respective fingertip vector has motion (i.e. it's fingertip vector magnitude is above some displacement threshold). Thus, the motion code for a gesture with significant motion in fingertip vectors 3, 4, and 5 only is represented as

$$11100 \qquad (4)$$

This binary number in decimal notation is 28, which is stored as the gesture's motion code.

7 Gesture matching

Gesture matching consists of comparing the unknown gesture with the models to determine whether the unknown gesture matches with any model gesture in the system vocabulary.

Motion codes enable the matching scheme to consider only those models which have a similar motion code as the unknown gesture and also provide information to which motion category the unknown gesture belongs. The library models, when loaded into memory, can be stored in an array of linear linked lists in which the array is indexed by the motion codes (0–31). During the matching stage, the unknown gesture need only be compared with the library models that are indexed by the unknown gesture's motion code in the linear linked list. Since it is known *a priori* that many of the stored models differ in their respective motion codes, random access is obtained to only those models in which a comparison is logical.

With only a subset of library models to compare to the unknown gesture, we have reduced the search complexity, which is now dependent on the different motion codes of the current library of gestures. A match is then determined by comparison between the stored models and the unknown gesture. A match is made if all vector fields (magnitude and direction for each fingertip) in the unknown gesture are within some threshold of the corresponding model entries.

8 Results

Ten sequences of over 200 frames were digitised at 4 Hz, stored, and then used for the recognition programme. Each run was performed in the same fashion, starting with the gesture for left and progressing to the ending gesture stop, as shown horizontally in Table 1.

Table 1: Results

Run	Frames	Left	Right	Up	Down	Rotate	Grab	Stop
1	200	✓	✓	✓	✓	✓	✓	✓
2	250	✓	✓	✓	✓	✓	✓	✓
3	250	✓	✓	✓	×	✓	✓	✓
4	250	✓	✓	✓	✓	✓	✓	✓
5	300	✓	✓	✓	✓	✓	✓	✓
6	300	✓	✓	✓	✓	✓	✓	✓
7	300	✓	✓	✓	✓	✓	✓	✓
8	300	✓	✓	✓	✓	✓	✓	✓
9	300	✓	✓	✓	✓	*	*	*
10	300	✓	✓	✓	✓	✓	✓	✓

✓ = recognised, × = not recognised, * = error in sequence

The number of images for each sequence depended on the duration of each gesture performed (i.e. a long execution for a gesture requires more images). The overall results on the ten sequences of images yielded almost perfect scores with the exception of run 9, where an error in the sequence caused the remaining gestures to be unrecognisable. A shift of the hand above the threshold limit or occlusion of points owing to lighting conditions may cause premature advancement of one phase to another, which in turn may result in the FSM continuing asynchronously with the image sequence. If the image sequences were performed in real-time, the user could possibly compensate for the sequence error by shifting back at a proper time interval, thus preventing erroneous output for the remainder of the image sequence.

An image set in which the fixed order shown in Table 1 was altered to [up, down, grab, rotate, left, right, stop] resulted in perfect recognition, which implies that gesture order is not a concern. A 3D plot of each fingertip with respect to the frame number is shown in Fig. 7 for this series of images. A gesture recognition diagram (see Fig. 8) was created for this sequence to show the length and order of execution of the gestures.

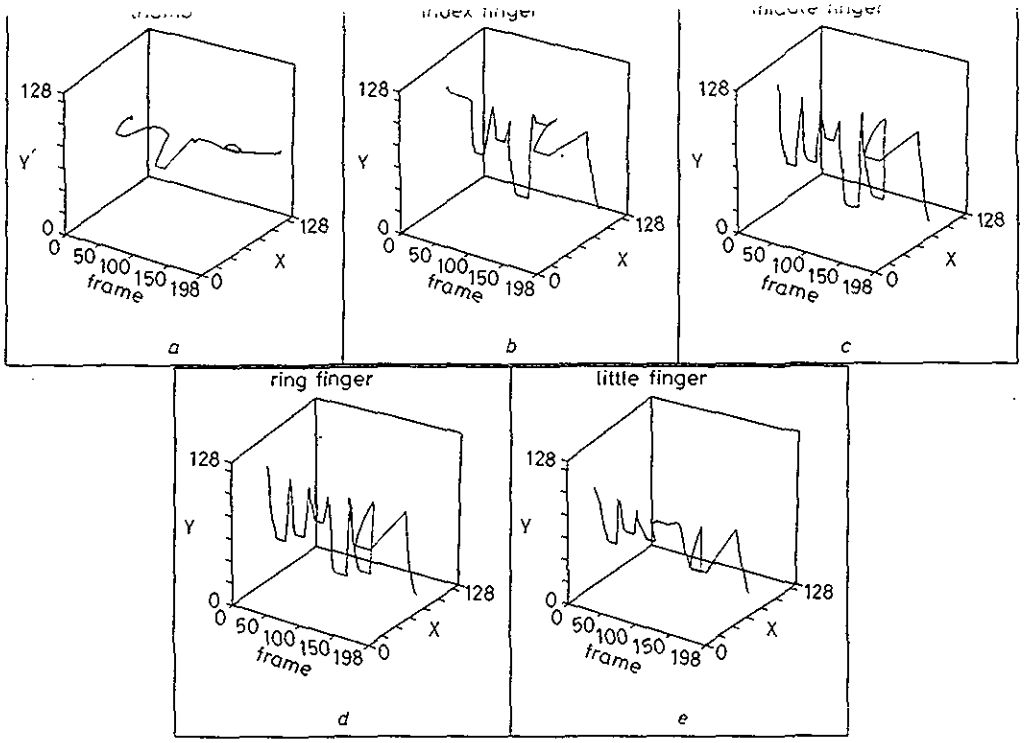

Fig. 7 *3D time plots*
a thumb trajectory *b* index finger trajectory *c* middle finger trajectory *d* ring finger trajectory *e* little finger trajectory

Recognition of a nine 128 × 128 frame gesture sequence sampled at 4 Hz took a CPU time of 890 ms on a SPARC-1, which implies 99 ms processing time per frame. It should be noted that our system did not require any special hardware. Our experiments show that sampling at a rate of 30 Hz is not necessary for gesture recognition and that 4 Hz is sufficient. The processing time needed for our method is small enough for implementation of this recognition algorithm in real-time with images sampled up to 10 Hz.

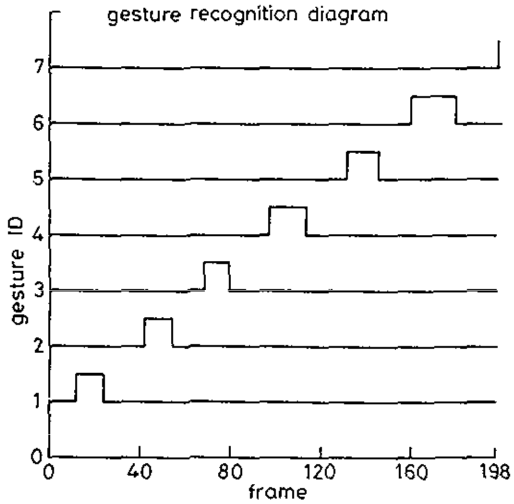

Fig. 8 *Gesture recognition diagram*
Depicts order of gesture execution, number of frames for execution for each gesture, and overall length of altered-order image sequence; gesture IDs: 1 = up, 2 = down, 3 = grab, 4 = rotate, 5 = left, 6 = right, and 7 = stop

9 Conclusion

In this paper, we have developed a computer vision method for recognising sequences of human-hand gestures within a gloved environment. A specialised FSM was constructed as an alternative to image sequence warping. We utilise vectors for representing the direction and displacement of the fingertips for the gesture. Modelling gestures as a set of vectors with a motion key allows the reduction of complexity in the model form and matching, which may otherwise contain multiple and lengthy data sets. We presented the performance of this method on real image sequences. Extensions being pursued are including more gestures, removing the glove environment, and relaxing the start/stop requirement.

10 References

1 CIPOLLA, R., OKAMOTO, Y., and KUNO, Y.: 'Robust structure from motion using motion parallax' (ICCV, IEEE, 1993), pp. 374–382
2 DARRELL, T., and PENTLAND, A.: 'Space–time gestures' (CVPR, IEEE, 1993), pp. 335–340
3 COSTELLO, E.: 'Signing: how to speak with your hands' (Bantam Books, New York, 1983)
4 FUKUMOTO, M., MASE, K., and SUENAGA, Y.: 'Real-time detection of pointing actions for a glove-free interface', *in* IAPR workshop on machine vision applications, 1992, pp. 473–476
5 RANGARAJAN, K., and SHAH, M.: 'Establishing motion correspondence', *CVGIP: image understanding*, 1991, **54**, pp. 56–73
6 BAUDEL, T., and BEAUDOUIN-LAFON, M.: 'Charade: remote control of objects using free-hand gestures', *CACM*, 1993, pp. 28–35