Dirasat, Engineering Sciences, Volume **32**, No. **1**, 2005

Real-Time Capturing and Visualization of Human Hand Poses Using Low Cost Web Cameras

Majid A. Al-Taee, Tarek Abu-Jabarah, Ihab Sultan and Ashraf Al-Karmi *

ABSTRACT

This paper presents a real-time tracking system for three-dimensional (3D) hand poses using a personal computer and two low-cost web cameras. A new computer algorithm is developed and implemented to capture and visualize a wide range of 3D hand poses. The proposed algorithm employs efficient 3D modeling and digital image processing techniques to map the extracted image features to an articulated 3D object like human hand poses. The fundamental components of this algorithm include a set of image acquisition functions, image processing and features selection functions, mapping functions, and a number of 3D modeling functions. The image processing and features selection functions are based on intensity information of the acquired images with static background. Neither machine training nor Cyber-Gloves or color information is required in the proposed system and therefore no restrictions are imposed on users except following a few simple predefined steps. The overall performance of the proposed system is examined experimentally and a robust real-time motion tracking of various hand poses is demonstrated.

KEYWORDS: Computer vision, Real-time Systems, Motion tracking, 3D human-hand modeling, Digital image Processing.

1. INTRODUCTION

Motion tracking of the human body or parts of it and recognizing their actions in video sequences are becoming of increasing importance in many applications (Plankers and Fua, 2001; Haritaoglu et al., 2000; Li et al., 2000). Among these applications, the importance of hand pose tracking is evident in different areas such as human-computer interfaces, sign language recognition, video coding, gestural communication, etc. (Rosales et al., 2001; Segen and Kumar, 1998). Different levels of accuracy are required by different applications (Athitsos and Sclaroff, 2002). Systems requiring accurate 3D hand parameters tend to use magnetic tracking devices and other non vision-based methods (Liang and Ouhyoung, 1998; Ma et al., 2000; Sagawa and Takeuchi, 2000).

Computer vision systems that estimate 3D hand poses typically do it in the context of tracking; different approaches have been studied resulting in the development of a number of working systems. Depending on the tracking approach being used, these systems can be broadly classified into the following two major categories:

a) Machine learning systems (Rosales et al., 2001; Athitsos and Sclaroff, 2002) that depend on comparing the acquired data with a number of pre-learned poses. Other poses have to be learnt or the system would fail to recognize them. Such systems are mostly suffering from complexity of the training process and the erroneous results that occur in the case of an unrecognized pose.

b) Recognizing the hand as a set of peaks and valleys where the hand pose can be estimated at the current frame as long as the system knows the pose in the previous frame (Heap and Hogg, 1996; Shimada and Shirai, 2001; Wu et al., 2001). In such systems, the previous frame parameters must be known and therefore only poses close to those parameters can be considered. Such systems suffer from the reduced number of poses that can be recognized because

* Computer Engineering Department, Al-Balqa' Applied University, Al-Salt (1); and Electrical Engineering Department, University of Jordan. Received on 28/5/2003 and Accepted for Publication on 16/1/2005.

sometimes the peaks and valleys cannot be detected for all fingers.

The latter approach is adapted in the present work and the process of capturing the 3D hand poses is based on acquiring sequences of non-colored images. Tracking of the hand's movement and capturing its different 3D configurations is achieved with the aid of using two low-cost web cameras. A new real-time computer algorithm that employs efficient image processing and 3D modeling techniques to map the extracted image features to an articulated 3D hand poses is developed and implemented in this work.

Compared with the equivalent existing tracking systems, the proposed system offers the advantage of real-time image acquisition, processing, and visualization of numerous hand poses together with the use of non-specialized and low-cost cameras. The acquisition rate of the used cameras is limited to 30 frames. Moreover, the process of features extraction is significantly simplified by using the two object views of the web cameras. Such a practical arrangement makes it possible to obtain the object's depth directly from the acquired images avoiding the necessity of using any intermediate mapping process such as the learning framework (Rosales et al., 2001) or a large database of synthetic views (Shimada et al., 2001; Athitsos and Sclaroff, 2002). Experimental tests of the proposed hand pose tracker demonstrate robustness and a vast number of hand poses that can be recognized.

2. SYSTEM OVERVIEW

An overview of the proposed system can be viewed in Fig. 1. The developed software algorithm, which is implemented in C++ and runs under Windows 2000 operating system, is designed to be robust while being simple and easy to modify. The system hardware consists of a microcomputer system based on Pentium processor with 400 MHz clock frequency and two right- angled web cameras as illustrated in Fig. 1(a). The system software performs three sequential functions; features selections, motion tracking, and a visualization function, as illustrated in the flowchart of Fig. 1(b).

The system operation starts by reading consecutive images (via the two web cameras) with a maximum rate of 30 frames. Separating the hand object from other undesired objects in the background is the prime task of the feature selection (or feature extraction). Detection of the hand's edges is then performed. This initial phase of the process involves some user interaction with the system in order to adjust certain threshold values depending on the light intensity of the workplace.

The second phase of the process updates the entire hand features each real-time cycle. This includes coordination points of the base and center of the hand's palm, and 14 links corresponding to the five fingers' joints. At this stage, an error detection and correction scheme is activated as necessary to avoid abnormal cases that may occur during the tracking process of the hand poses. Finally, the results obtained by motion tracking modules are then transferred to an articulated 3D hand model in order to visualize the captured hand poses on the computer screen in real time.

3. FEATURES SELECTION

The initial phase of the features-selection process is the image segmentation. The resulting image consists of the foreground object with distinct edge pixels located on its boundary. Locating an initial edge pixel and continuing search for a new edge pixel using a typical pixel-by-pixel search procedure in a preset direction can achieve this objective. However, problems of either not finding more pixels in the desired direction or the existence of some discontinuity points between an edge pixel and the subsequent one are encountered, as shown in Fig. 2(a). In such cases, modifying the search procedure to consider an array of 3×3 or 5×5 pixels is found necessary to avoid such problems. Once a subsequent pixel is found, the algorithm returns to its default pixel-by-pixel search procedure. Implementing such a procedure yields a continuous edge of the hand as illustrated in the example of Fig. 2(b).

The second phase of this process deals with the extraction of the desired hand features, which includes the coordination points of base and center of the hand's palm together with the base, tip, length and width of each finger. In this important phase of the process, a new algorithm that is based on drawing patterns of half circles in a clockwise order is developed and implemented. The following steps summarize the essential functions of the proposed features-selection module.

Step I: In this step, the algorithm starts by drawing a

set of concentric half circles centered at pixel position $\{P(c_0)\} = \{x(c_0), y(c_0)\}$ where $x(c_0) = image\ width/2$ and $y(c_0) = 0$. $P(c_0)$ represents the base point of the palm, as shown in Fig. 3(a). The half circles start at a radius of 20 pixels and go up to a radius of 285 pixels, with steps of 10 pixels. These circles initially intersect with the edge of the hand. The number of intersection points is used to perform a number of sub-tasks. First, these half circles locate an initial base for the thumb. As soon as the intersection points of two consecutive circles become three and four consecutively, this ensures that the first two intersection points represent the initial base of the thumb. These intersection points are saved in a pair of registers in order to be used later in the processes of both features selection and motion tracking. Such a restriction on the number of intersection points guarantees that these points correspond to the base of the thumb.

The next sub-task performed by the half circles would be to locate the initial base points for the remaining four fingers. Once the number of intersection points becomes equal to or greater than eight, the radius of the current half circle (r_1) and the last eight intersection points ($Pi_{(1)}$, $Pi_{(2),\ \ldots\ ,}$ $Pi_{(8)}$) are saved. Next, when the number of the intersection points goes down to zero, this means that the current half circle is no longer intersecting with the hand. The radius of this last half circle (r_2) is also saved for the next step. It should be mentioned here that the difference between r_1 and r_2 approximates the length of the longest finger in the hand; the middle finger.

Step II: In this step, the previously saved radii and intersection points are used to find a new center point $\{P(c_1)\} = \{x(c_1), y(c_1)\}$for a second set of half circles, as shown in Fig. 3(b). The *x*-*y* coordinates of the point $P(c_1)$ are found using the following simple procedure:

a) Find the difference between r_1 and r_2.

b) Find $y(c_1)$ by subtracting half of the difference from r_1:

$$y(c1) = r1 - \left(\frac{r2 - r1}{2}\right) \tag{1}$$

c) Find $x(c_1)$ as the average of the last intersection point ($Pi_{(8)}$) found in step I and the fifth intersection point before $Pi_{(8)}$, ($Pi_{(8\text{-}5)}$). These two points are specifically selected because of the fact that sometimes the half circles intersect with only two points just at the base of the middle finger. This will mistakenly consider the number of intersection points to be eight, whereas what is required is two intersections per finger at positions slightly above the base of each finger. Averaging the intersection points in the above-explained manner, guarantees that $x(c_1)$ is calculated as required. Mathematically, $x(c_1)$ is obtained using:

$$x(c1) = \frac{p_{i(8)} - p_{i(8-5)}}{2} \tag{2}$$

Now, once the computation of $P(c_1)$ is completed, the algorithm starts drawing a new set of concentric half circles, starting from the top of the image (with a radius of 285 pixels) and moving downwards with steps of -10 pixels. Starting from the top of the image guarantees faster intersection of the new set of half circles with the four fingers. The values of this new set of intersection points ($P'_{i(1)}, P'_{i(2)},\ldots, P'_{i(8)}$) are used in the next step to extract additional features.

Step III: In this step, the new intersection points ($P'_{i(1)}, P'_{i(2)},\ldots, P'_{i(8)}$) are used to find a new center point and a new radius for a new set of half circles for each of the four fingers, as shown in Fig. 3(c). Each pair of consecutive intersection points is used to find the center and radius of the half circles that correspond to one finger. The *x*-*y* coordinates of the fingers` center points $P_f(c_1)$ through $P_f(c_4)$ are obtained as follows:

$$\{P_f(c1)\} = \{x_f(c1), y_f(c1)\} = \left\{\frac{x(P'_{i(1)}) - x(P'_{i(2)})}{2}, \frac{y(P'_{i(1)}) - y(P'_{i(2)})}{2}\right\} \tag{3}$$

$$\{P_f(c2)\} = \{x_f(c2), y_f(c2)\} = \left\{\frac{x(P'_{i(3)}) - x(P'_{i(4)})}{2}, \frac{y(P'_{i(3)}) - y(P'_{i(4)})}{2}\right\} \tag{4}$$

$$\{P_f(c3)\} = \{x_f(c3), y_f(c3)\} = \left\{\frac{x(P'_{i(5)}) - x(P'_{i(6)})}{2}, \frac{y(P'_{i(5)}) - y(P'_{i(6)})}{2}\right\} \tag{5}$$

$$\{P_f(c4)\} = \{x_f(c4), y_f(c4)\} = \left\{\frac{x(P'_{i(7)}) - x(P'_{i(8)})}{2}, \frac{y(P'_{i(7)}) - y(P'_{i(8)})}{2}\right\} \tag{6}$$

The radii of the new sets of fingers` half circles are obtained from:

$$r_{f1} = \sqrt{\left(x_f(c1) - x(P'_{i(1or2)})\right)^2 + \left(y_f(c1) - y(P'_{i(1or2)})\right)^2} + \varepsilon \quad (7)$$

$$r_{f2} = \sqrt{\left(x_f(c2) - x(P'_{i(3or4)})\right)^2 + \left(y_f(c2) - y(P'_{i(3or4)})\right)^2} + \varepsilon \quad (8)$$

$$r_{f3} = \sqrt{\left(x_f(c3) - x(P'_{i(5or6)})\right)^2 + \left(y_f(c3) - y(P'_{i(5or6)})\right)^2} + \varepsilon \quad (9)$$

$$r_{f4} = \sqrt{\left(x_f(c4) - x(P'_{i(7or8)})\right)^2 + \left(y_f(c4) - y(P'_{i(7or8)})\right)^2} + \varepsilon \quad (10)$$

where ε is a small offset added to the calculated radius to ensure that the finger's half circles cross the finger's boundary.

The calculated center points and radii are used to draw new sets of half circles as illustrated in Fig. 3(c). If the number of intersection points is equal to or greater than two, the program selects the first two points before and after the center and uses them to find a new center and a new radius in the downward and upward direction until it reaches one of the finger ends, the base and tip. Once the finger's initial tip is found, using the half circles, the edge tracking technique is used to find a more accurate tip of the finger. This procedure is repeated for all the fingers, except the thumb. Knowing the fingers' tips and bases, the length, width and angle of inclination of each finger can be obtained, as shown in Fig. 3(d).

Step IV: This step deals with the process of features selection of the thumb. Unlike other fingers, the base and tip of the thumb are obtained in a similar manner to that used to find the hand edge. The process starts from the base intersection points of the thumb that were already obtained in step I. The edge detection process then moves forward until it reaches the tip of the thumb. Using this features selection procedure; the tip, base, length, width, and the inclination angle of the thumb are obtained, as can be shown in Fig. 4.

4. MOTION TRACKING

The motion-tracking module of the proposed algorithm is responsible for updating the above mentioned hand features in every real-time cycle. The following three distinct steps describe the main functions performed by this module:

Step I: This step is responsible for updating the palm base and thumb features only. Analyzing the location and possible poses of the thumb, it was found that a certain field can be specified for this finger. Referring to Step I of the features selection procedure described in Section 3, the so-called initial base of the thumb was obtained from the first three and four intersections of a half circle centered at the palm base point $\{P(c_0)\}$ with the edge of the hand. This half circle is used to identify the lower boundary of the thumb field, as shown in Fig. 4. The upper boundary of the thumb field on the other hand, is defined by another half circle with radius (r_1), which intersects with the other four fingers at points ($Pi_{(1)}$, $Pi_{(2),\ldots}$, $Pi_{(8)}$). This information was already obtained and used in Step I of the previous section to identify the initial bases of the four fingers. The entire area bounded by the upper and lower points is considered as the thumb area that is processed independently. The base and tip of the thumb are updated by adopting a similar procedure to that described in Step IV of the features selection.

Step II: The base and tip points of the remaining four fingers are updated in this step for each real-time cycle. The known base and tip values of each finger are used to find a centerline representing the corresponding finger in the previous frame. This line plays an important role in minimizing the possible errors that result in the updating process of the bases and tips, especially in the cases where the intersection points are very close to each other. The updating process for each finger is based on drawing a set of half circles in the upward and downward direction to update the tip and base, respectively. For example, starting from the base point of the previous frame, the upward movement updates the corresponding tip and reversing this procedure updates the corresponding base. The radius of these half circles is calculated using Eqns. 7 –10.

Step III: This step can be considered as a sub-task of that described in Step II. When two or more fingers are joined together and no intermediate edge(s) can be detected, the following two-pass procedure is performed to overcome the difficulty of updating the base and tip points of the joined fingers.

a) The first pass searches for intersection points that exist around finger's centerline, starting with the left side of the centerline. If an intersection point is detected, the procedure incremented with a step of ten pixels in the upward direction. Then, it moves in the left direction until it reaches the left edge of the finger in question. Once the left edge is detected, the search procedure moves back to the right a distance equal to a half width of the current finger and saves the coordinates of the current position. Next, the procedure moves a distance of thirty pixels (on the same centerline) in the upward direction. Similarly, the procedure then moves to the left edge, back to the right by a half width of the current finger, and saves coordination of the second position. At this stage, the program draws a line with a slope defined by the obtained position points; starting from the base of the finger in the previous frame and moving upward until it intersects with the edge. This represents a new centerline of the finger upon which the base and tip of the finger are updated. If the intersection point is found to the right of the centerline, the above procedure will repeat the same steps except that it moves in opposite directions to those mentioned above.

b) The second pass deals with the conditions where no intersection points are detected either before or after the centerline. This means that more than two fingers are joined together. In such cases, the procedure calculates a new slope, depending on the tips and bases of the outer fingers, as follows:

$$Slope = \frac{x_{(tip1)} + x_{(tip2)} - x_{(base1)} - x_{(base2)}}{y_{(tip1)} + y_{(tip2)} - y_{(base1)} - y_{(base2)}} \quad (11)$$

The new slope, which is the reciprocal of the average slopes of both lines representing the two outer fingers, is used by the algorithm to draw estimated centerlines corresponding to the inner joined fingers. Examples of edge adjustment performed by the second pass are shown in Fig. 5.

5. SYNTHESIZE OF 3D HAND POSES

The major advantage resulting from using two camera views is that the above described features extraction and motion tracking modules are equally applied to acquired images of both camera views. Also, the lengths and inclination angles of the fingers become the only major parameters of interest in each view. The mapping process of the hand image features to the articulated 3D hand model can be explained with reference to both front and bottom camera views.

5.1 Front View

The front view image of the hand poses provides information about the movement in the horizontal plane together with the angle of inclination for each finger. Mathematical formulation of the procedure used to calculate the required parameters is presented here with reference to Fig. 6. Two cases are considered in the following analysis; the partially curled and fully curled finger conditions.

5.1.1 Partially Curled Finger

From the triangle in Fig. 6(a), the following equation can be deduced:

$$\vartheta = 90 - \frac{1}{2}\phi \quad (12)$$

where ϕ is the required angle of inclination. Using the same triangle, the finger length as seen by the front view camera is given by

$$X = Z \times \cos(\vartheta + 180 - \phi) \quad (13)$$

Substituting eqn. 12 in eqn. 13 yields

$$X = Z \times \cos\left\{\left(90 - \frac{1}{2}\phi\right) + 180 - \phi\right\} \quad (14)$$

Rearranging eqn. 14 in terms of Z gives

$$Z = \frac{X}{\cos\left(270 - \frac{3}{2}\phi\right)} \quad (15)$$

The value of Z can also be obtained from:

$$Z = \sqrt{Y^2 + Y^2 - 2 \times Y \times Y \times \cos(\phi)} = \sqrt{2 \times Y^2 (1 - \cos(\phi))} \quad (16)$$

where Y equals one third of the full length of the finger. Now, equating eqn 15 with eqn. 16 results in

$$2 \times Y^2 (1 - \cos\phi) = \frac{X^2}{\cos^2\left(270 - \frac{3}{2}\phi\right)} \quad (17)$$

Knowing both *X* and *Y*, the required angle of inclination (ϕ) can be calculated from eqn. 17.

5.1.2 Fully Curled Finger

When a finger is fully curled as illustrated in Fig. 6(b), the following equation can be deduced:

$$\cos(180-\phi)=\frac{X}{Y} \tag{18}$$

Similarly; knowing the values of *X* and *Y*, eqn. 18 can be used to calculate angle of inclination (ϕ).

5.2 Bottom View

Information acquired from the bottom camera view is used to verify the validity of the angle values obtained from the previous analysis of the front view. The length of the finger seen in the front view is initially subtracted from the full length of the finger. The difference value is now compared with the detected finger length in the bottom view. If the difference between the latter two values is found to be within a certain margin, the analysis procedure returns a true value and all the obtained information is considered valid. Otherwise, if the difference exceeds the allowable margin, the analysis procedure is terminated and returned to the features selection module where an error correction procedure is activated. This procedure, which is based on an iterative searching process, is capable of identifying any selected feature (such as the base and tip points) that is positioned outside the segmented image. When identified, such points are shifted to their appropriate positions.

At this stage, the obtained lengths of the hand fingers together with their angles of inclination are sent to an articulated 3D hand model for visualization purposes. It should be mentioned here that the obtained hand features could also be sent (via an appropriate hardware interface) to local or remote destination devices.

6. VISUALIZATION OF 3D HAND POSES

Three-dimensional articulated hand model is designed and implemented to visualize the 3D hand poses. This is achieved by providing the articulated model with the hand features that are extracted during a real-time period of approximately 33 milliseconds. The model consists of 15 links; the palm and 14 links corresponding to the fingers' knuckles. Each finger has three knuckles as illustrated in Fig. 7. The five joints connecting the fingers to the palm allow rotation with two degrees of freedom, whereas the 9 joints between the finger links allow rotation with one degree of freedom. The model is made to be simple to minimize the computation overhead on the real-time cycle resulting in a faster responding model. In spite of its design simplicity, the model however enables the user to change the viewing angle, colors, and zooming in and out of the object. Other effects like lighting and texture mapping (applying images to the 3D model) are considered less important here as the 3D model is intended to be used only for illustration purposes.

A number of Application Programming Interfaces (API's) are developed to simplify and speed up the process of code writing that generates 3D scenes; of those API's, the OpenGL (Neider et al., 1994) is the most common interface. The model is first built using 3D Studio MAX (Murdock, 2000) to find the best view that displays the movement of the hand palm and fingers. Then, after obtaining the desired sizes of the hand elements together with the proper camera angle, the OpenGL model is constructed using C++ and Windows programming code. Additional Windows programming codes (Adams, 1994) are also found necessary for the windows opening, full screen mode functionalities and resize options.

7. COMPLEXITY AND REAL-TIME CONSIDERATIONS

Real-time systems are computer systems whose behavior depends on a certain time requirements. The violation of these requirements may damage the overall operation of the system. Time complexity analysis of real-time algorithms therefore represents an essential requirement to provide a satisfactory real-time response (Shimada et al., 2001; Kone, 2001). Such analysis decides whether or not the system response will be acceptable. The time taken by an algorithm is the sum of both compile time and processing time (or execution time). However, as the compile time is not involved in real-time operation of the algorithm, the present analysis is limited to the processing time (Tp) of the proposed

algorithm. The Tp can be estimated either analytically or experimentally (Sahni, 1998). The latter approach is adopted in the present work to estimate Tp of the entire modules of the proposed algorithm.

The practical arrangement of Tp measurement is shown in Fig. 8(a). Once the algorithm execution starts, a pulse is sent (via parallel port) to the preset (Pr) input of the D flip-flop (DFF) making its output (Q) at logic-1. Similarly, at the end of executing the visualization module of the algorithm, another pulse is sent to the clear (Clr) input of the DFF resetting its output to logic-0, as illustrated. The time difference between the rising and falling edges of the generated pulse approximately represents the processing time of the algorithm.

Experimental measurements showed that Tp consumes 80-90% of the real-time period as illustrated in Fig. 8(b). The minimum processing time (Tp_{min}) occurs with the conditions of no joint fingers and no error correction. Otherwise, the Tp may consume up to 90% (Tp_{max}) of the real-time period. Finally, it should be mentioned here that the system response to the user commands becomes slower as the processing time becomes closer to the real-time period and vice versa.

8. RESULTS AND DISCUSSION

The hand tracking system described above has been implemented and tested for different hand poses. Figures 9-11 show selected samples of the demonstrated experimental results. Each of these figures illustrates the different steps of hand pose acquisition, tracking, and visualization. The upper two images in these figures illustrate the acquired images from the front and bottom cameras. The resolution of the image frames acquired in these figures is 320 × 240 pixels. The lower two images on the other hand illustrate the feature extraction and visualization phases of the tracking process. The results presented in these figures are for one, two and four curling fingers, respectively. The successful motion tracking of hand poses is quite obvious in these experiments.

The developed software algorithm, which has been implemented in C++, is designed to be modular, robust, and easy to modify. The error detection and correction procedures that were implemented in the motion analysis modules, significantly contributed in improving the overall system performance. However, in spite of what has been achieved in this work, there are still several directions that can effectively contribute in extending the demonstrated functional capability of the proposed system as well as improving its performance. Some of these directions are suggested in the following steps:

a) The web cameras used in the present system had a relatively low capture rate and introduce certain amount of noise in the captured images. Although such noise can be removed throughout the process of motion capture, better results can be obtained by using better cameras.

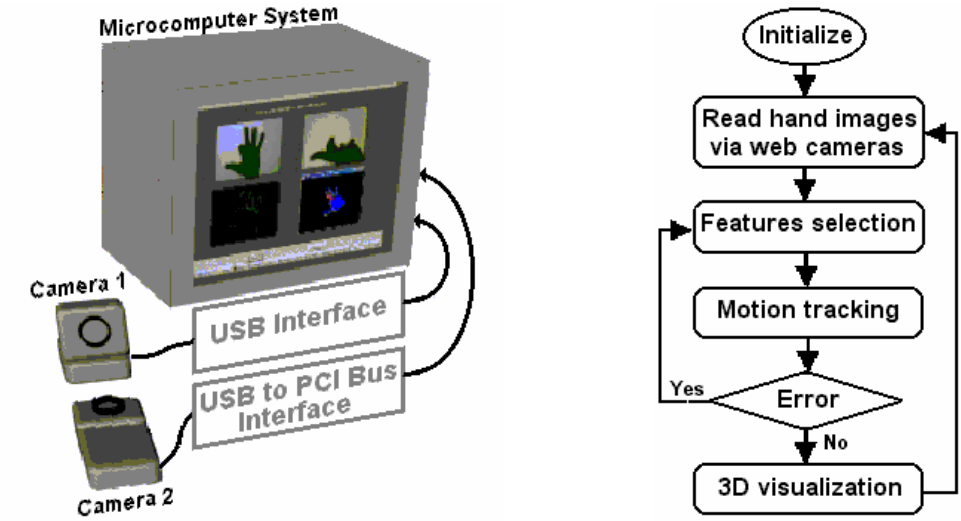
b) Like other real-time computer vision system, computer speed makes significant difference. In the present system, effort was made to compromise between the maximum capture rate of the used cameras and the relatively low speed of the used microcomputer system. This imposes some limitation on the speed of the hand movements and the motion tracking cannot be guaranteed for a sudden change between any two successive frames. Also, some of the rarely occurring hand poses (i.e. crossing fingers) are excluded to satisfy the allowable timing frame specified by the real-time cycle.

c) The work presented in this paper focuses on the motion tracking of a single hand. Extension to motion tracking of two hands poses can be achieved with a faster computer system that is capable of processing additional software modules without overrunning the time frame of the specified real-time cycle.

9. CONCLUSIONS

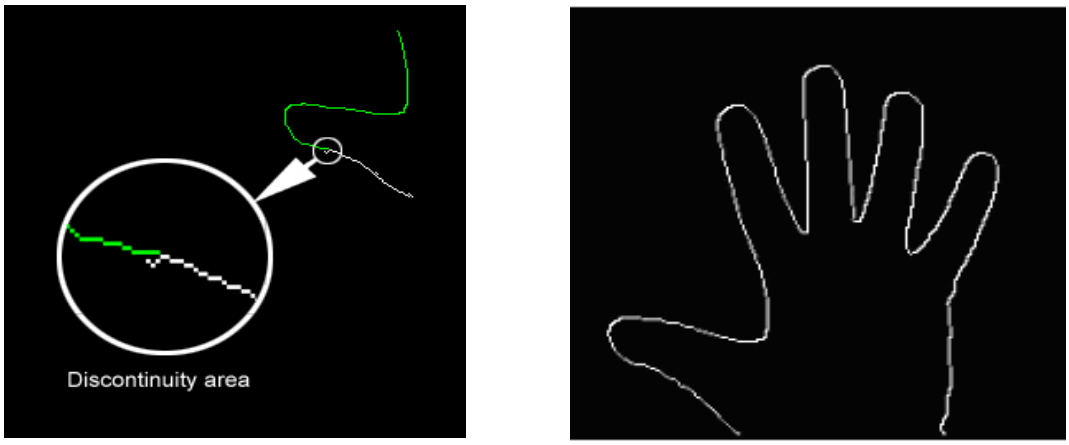
A new computer algorithm for tracking and visualization of hand poses in real-time is presented. Although the presented experiments deal so far only with a single hand, the extension to full hands is obviously possible with appropriate software and hardware modifications. The main advantage of the proposed technique is that the required 3D parameters of the hand poses are directly extracted from the acquired images without using any complex intermediate mapping process

such as a training framework or a large database of synthetic views. The obtained features of the acquired hand images can also be sent (via an appropriate hardware interface) to local or remote destination devices for various control purposes. In this case, only the features of the acquired hand images are sent instead of the hand image itself. The use of non-specialized low-cost cameras in the proposed system would also make it a suitable choice for numerous applications such as on-line sign language interpretation, human-machine interface and various virtual reality applications.



(a) (b)

Fig. 1: Overview of the proposed system.
(a) System hardware elements
(b) System software modules



(a) (b)

Fig. 2: Hand-edge detection.

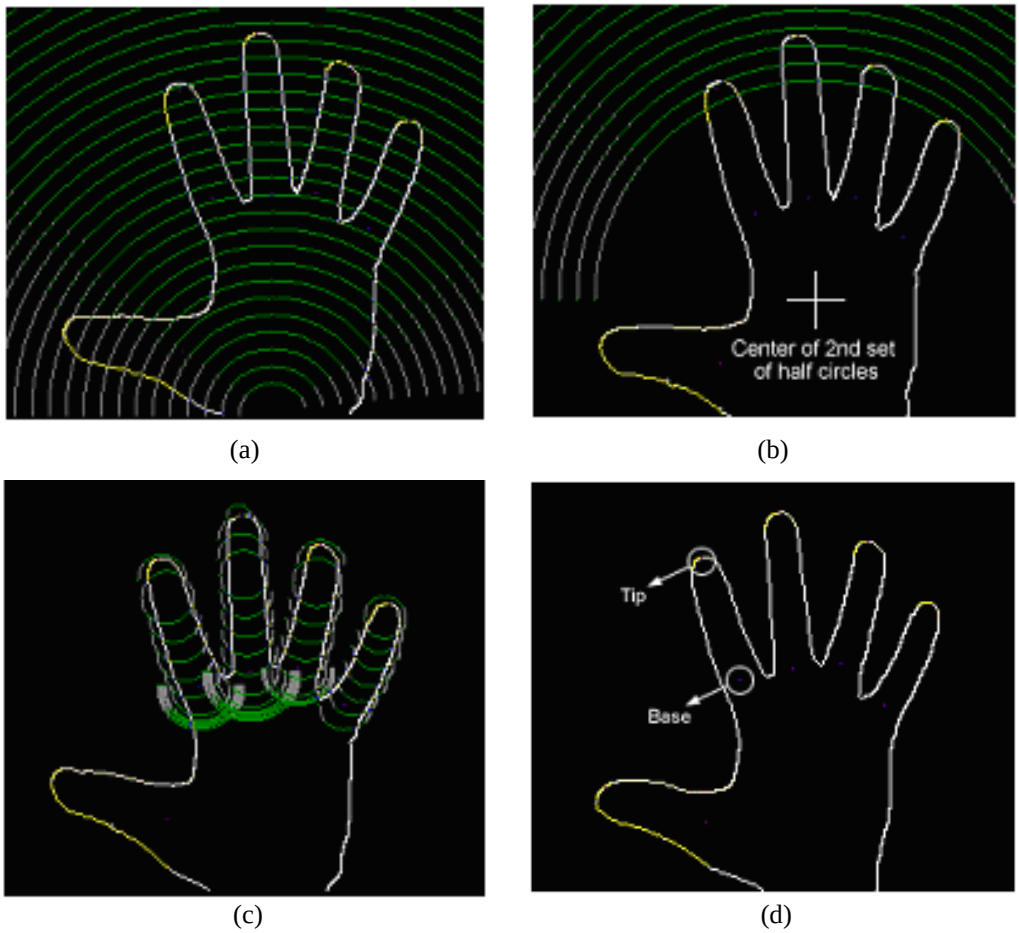
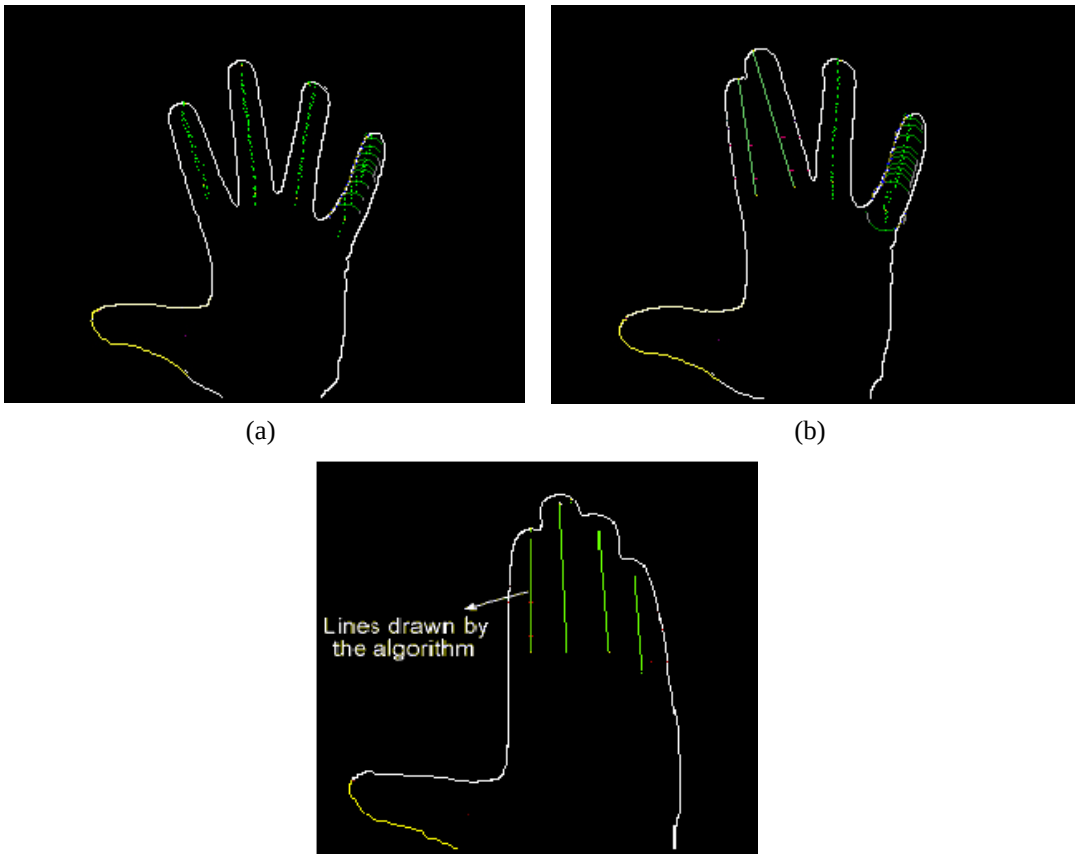


Fig. 3: Steps of features selection.

Fig. 4: The thumb field.

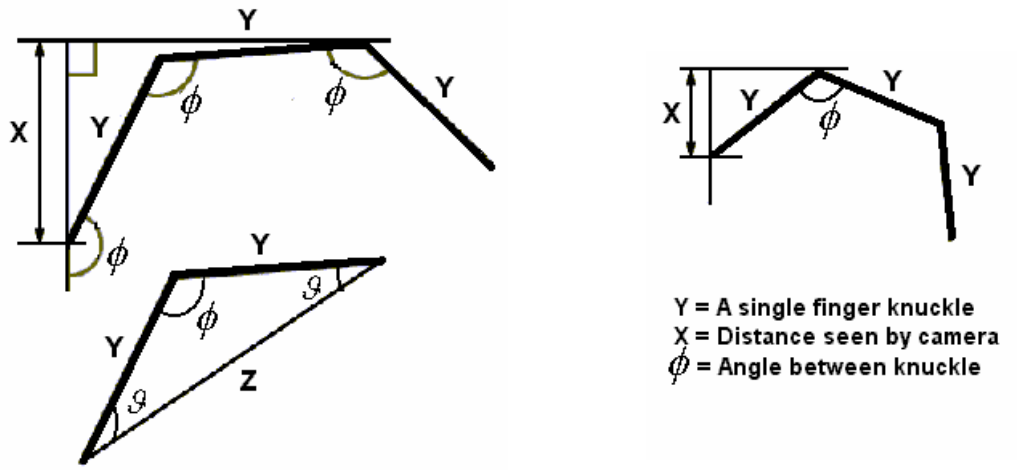


(a) (b)

(c)

Fig. 5: Algorithm adjustments for joined fingers.

a) Normal case
b) Two joined fingers
c) Four joined fingers



(a) (b)

Fig. 6: The angle of inclination.

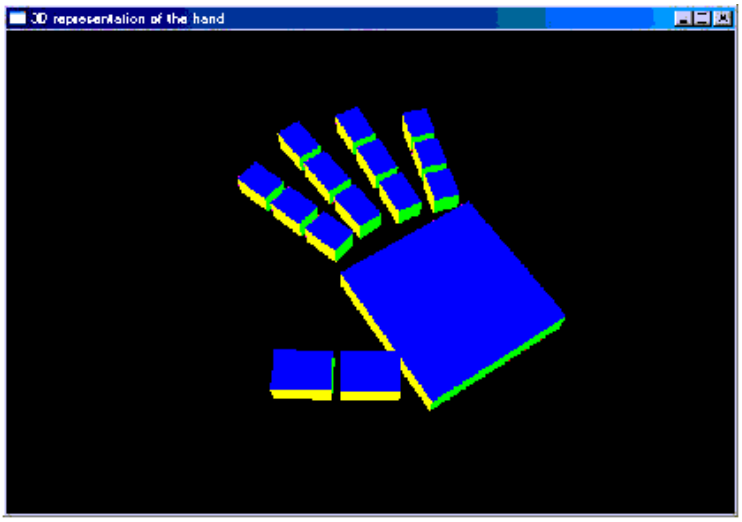
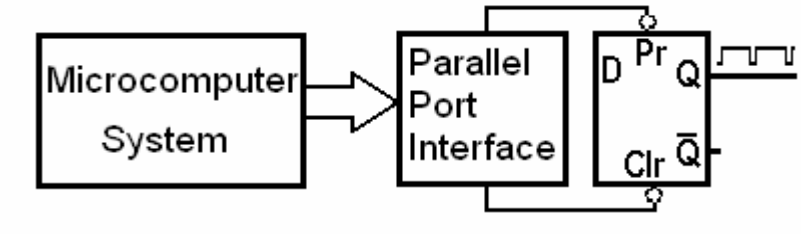
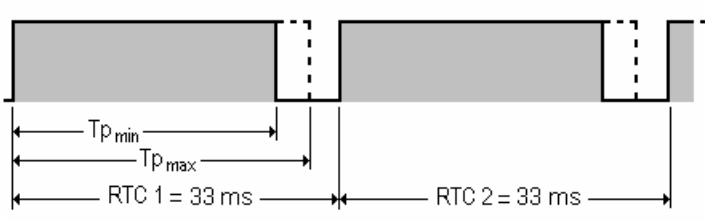


Fig. 7: Articulated 3D hand model.



(a)



(b)

Fig. 8: Measurement of real-time response.

(a) Experimental setup

(b) Real-time cycle (RTC)

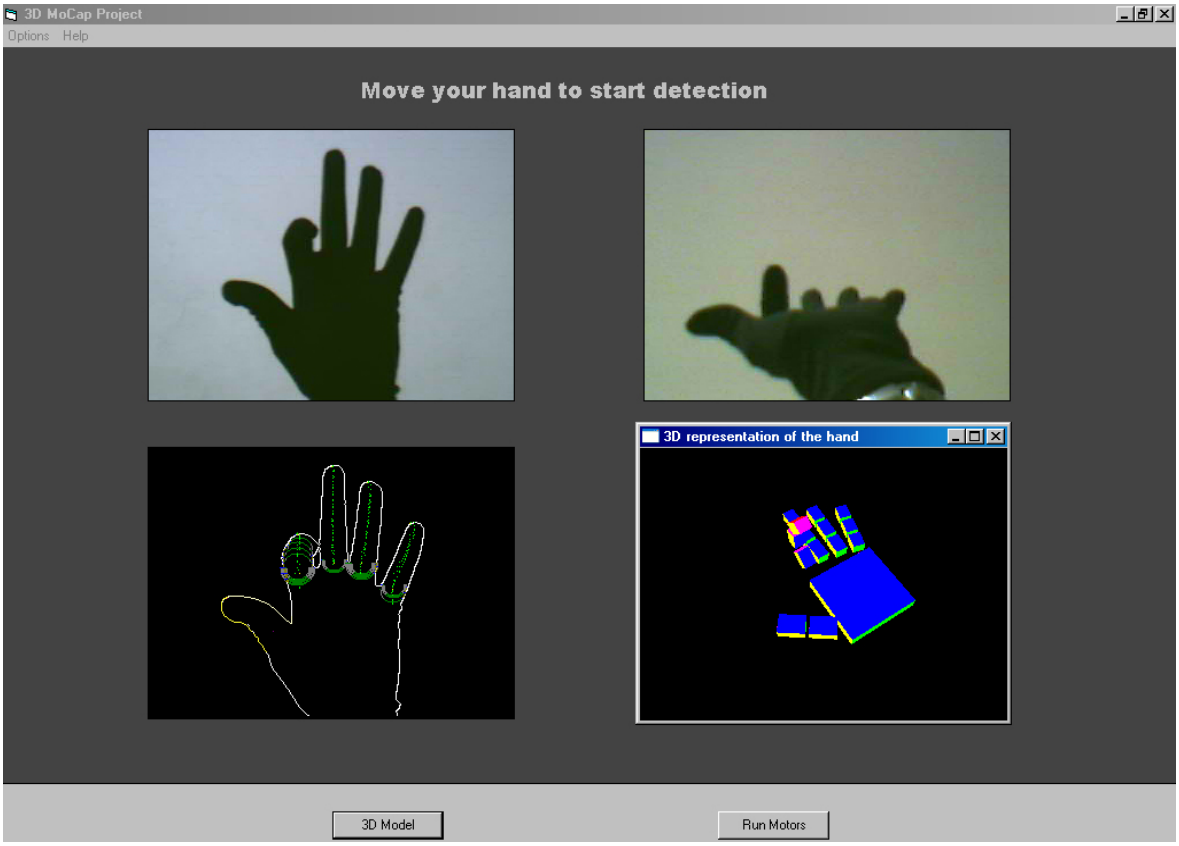


Fig. 9: Steps of features selection and visualization of a single curled finger.

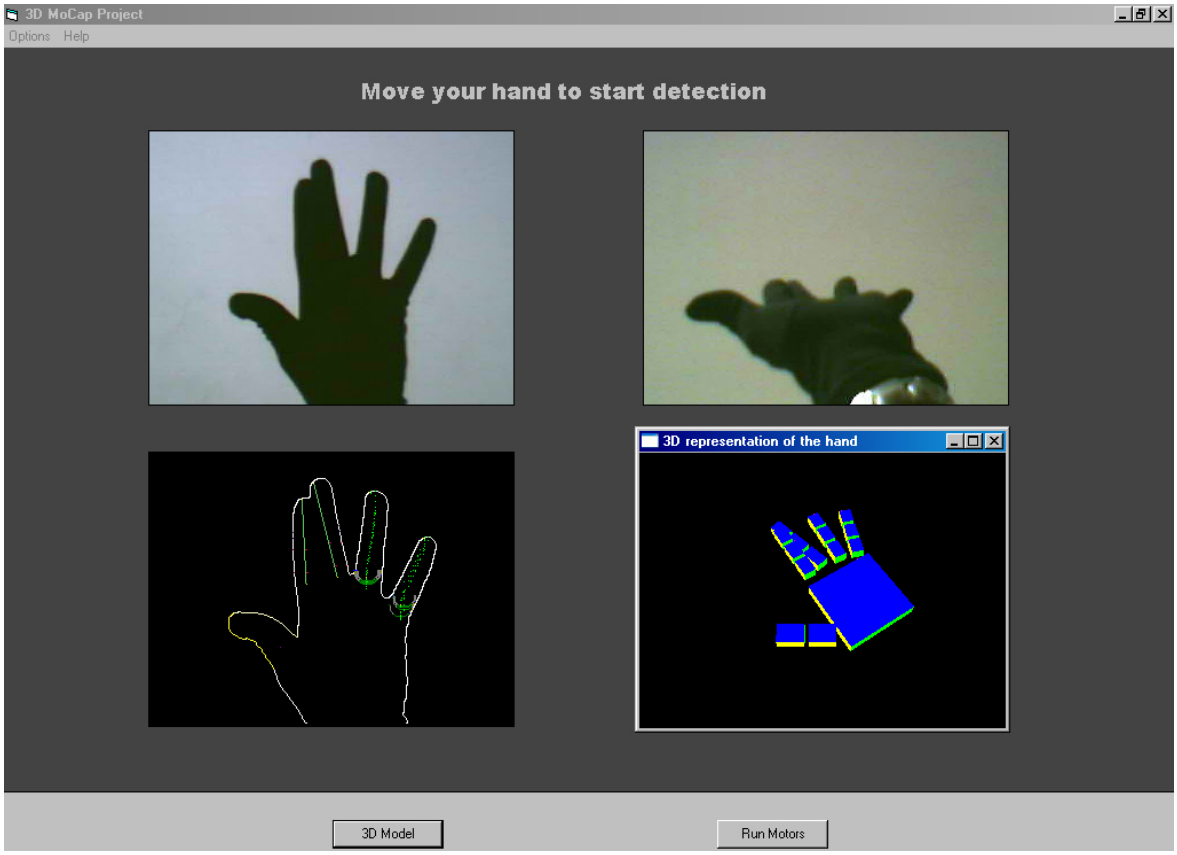


Fig. 10: Steps of features selection and visualization of two curled fingers.

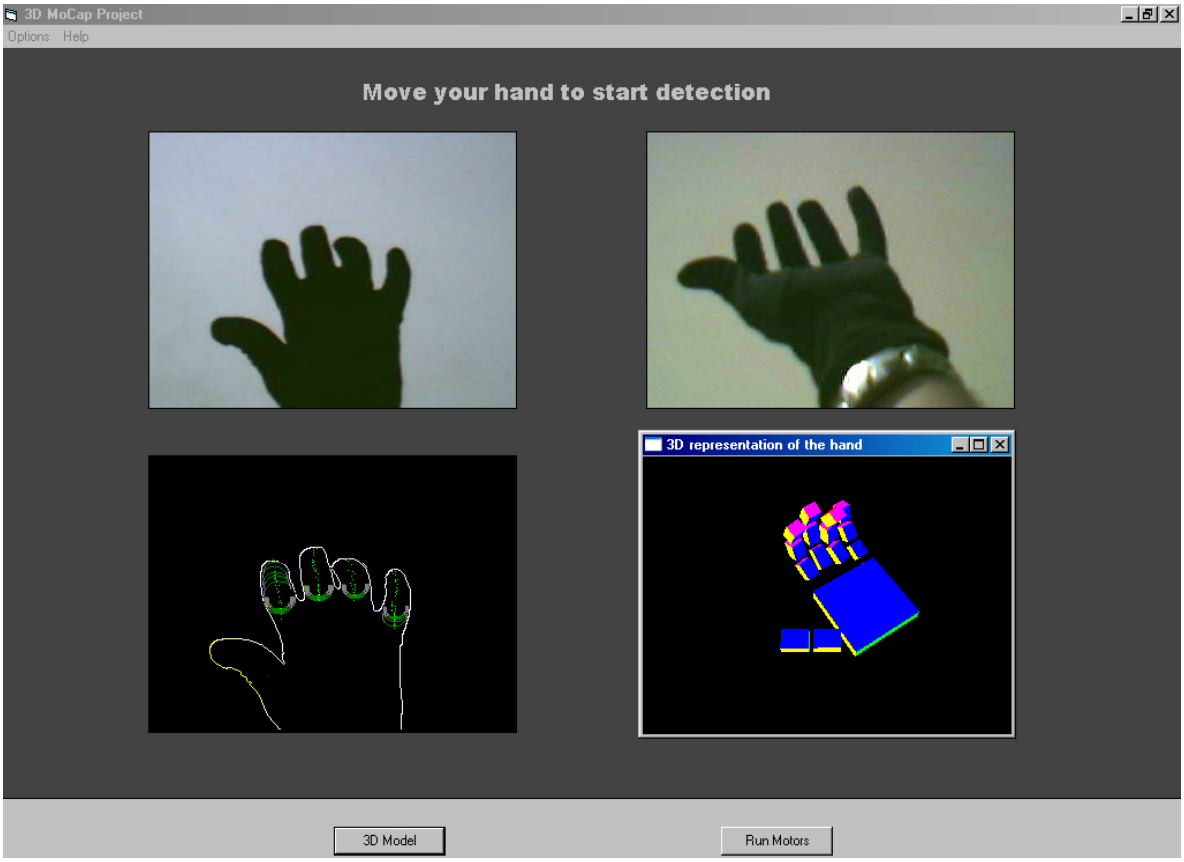


Fig. 11: Steps of features selection and visualization of four curled fingers.

REFERENCES

Adams, L. 1994. *Windows Visualization Programming with C/C++*. McGraw Hill.

Athitsos, V. and Sclaroff, S. 2002. An Appearance-based Framework for 3D Hand Shape Classification and Camera Viewpoint Estimation. *IEEE Int. Conf. on Automatic Face and Gesture Recognition (FGR'02).*

Haritaoglu, I., Hartwood, D. and Davis, W. 2000. Real-time Surveillance of People and their Activities. *IEEE Trans. on PAMI.* 22: 809-830.

Harwood, D., Haritaoglu, I. and Davis, L.1998. A Real Time System for Detecting and Tracking People. *Proc. Int. Conf. on Face and Gesture Recognition*. April 14-16. Nara-Japan.

Heap, T. and Hogg, D. 1996. Towards 3D Hand Tracking Using a Deformable Model. *J. Face and Gesture Recognition*, 140–145.

Kone, O. 2001. A Local Approach to the Testing of Real-time Systems. *The Computer Journal-British Computer Society*, 44(5): 1-12.

Li, Y., Goshtasby, A. and Garcia, O. 2000. Detecting and Tracking Human Faces in Videos. *Proc. Int. Conf. ICPR'00.* 1: 807-810.

Liang, R. and Ouhyoung, M. 1998. A Real-time Continuous Gesture Recognition System for Sign Language. *J. Face and Gesture Recognition.* 558–567.

Ma, J., Gao, W. and Wu, C. 2000. A Continuous Chinese Sign Language Recognition System. *J. Face and Gesture Recognition.* 428–433.

Murdock, K. L. 2000. *3D Studio MAX R3 Bible*. John Wiley and Sons Inc.

Neider, J., Davis, T. and Woo, M. 1994. *OpenGL Programming Guide*. Addison-Wesley publishing Company.

Plankers, P. and Fua, P. 2001. Tracking and Modeling People in Video Sequences. *J. Computer Vision and Image Understanding.* 81: 285-302.

Rosales, R., Athitsos, V., Sigal, L. and Sclaroff, S. 2001. 3D

Hand Pose Reconstruction Using Specialized Mappings. *IEEE Int. Conf. on Computer Vision (ICCV).* Canada. 378–385.

Sagawa, H. and Takeuchi, M. 2000. A Method for Recognizing a Sequence of Sign Language Words Represented in a Japanese Sign Language Sentence. *J. Face and Gesture Recognition.* 434-439.

Sahni, S. 1998. *Data Structures, Algorithms, and Applications in C++*. WCB/McGraw-Hill.

Segen, J. and Kumar, S. 1998. Human-computer Interaction Using Gesture Recognition and 3D Hand Tracking. *Proc. Int. Conf. ICIP*. Chicago. 188-192.

Shimada, N., Kimura, K. and Shirai, Y. 2001. Real-time 3-D Hand Posture Estimation Based on 2-D Appearance Retrieval Using Monocular Camera. *J. Recognition, Analysis and Tracking of Faces and Gestures in Real-time Systems.* 23–30.

Wu, Y., Lin, J.Y. and Huang, T.S. 2001. Capturing Natural Hand Articulation. *IEEE Int. Conf. on Computer Vision (ICCV).* Canada. 2: 426-432.

التتبع والعرض بالزمن الحقيقي لحركة يد الانسان باستخدام كاميرتي ويب قليلة الكلفة

ماجد الطائي، وطارق ابو جبارة، وايهاب سلطان، واشرف الكرمي*

ملخـص

تقدم هذه الورقة منظومة تتبع ثلاثية الابعاد لحركة يد الانسان باستخدام حاسوب شخصي وكاميرتي ويب قليلة الكلفة. تم تطوير واستخدام خوارزمية جديدة تعمل بالزمن الحقيقي للحاسوب لتخمين وعرض اوضاع مختلفة لليد على نطاق واسع. تستخدم الخوارزمية المقترحة تقنيات كفؤة للتمثيل ثلاثي الابعاد ولمعالجة الصور الرقمية وذلك لغرض استخلاص الخصائص المناسبة ومن ثم تحويلها الى نموذج اصطناعي مشابه ليد الانسان. تشتمل المكونات الاساسية لهذه الخوارزمية على مجموعة من وظائف جمع الصور، ووظائف معالجة الصور واستخلاص الخصائص، ووظائف التحويل، وعدد من وظائف التمثيل ثلاثي الابعاد لليد. تعتمد وظائف معالجة الصور واستخلاص الخصائص على كثافة المعلومات في الصور الرقمية قيد المعالجة مع خلفية ثابتة. ان المنظومة المقترحة لاتستخدم الكفوف الحساسة او الوان الصور في عملية جمع البيانات ومعالجتها وانما تعتمد بصورة اساسية تقنية الرؤية الذاتية للحاسوب. ان هذه التقنية لاتفرض اية قيود على المستخدمين باستثناء اتباع عدد محدود من الخطوات لبدء العمل. تم اختبار خصائص الاداء للمنظومة المقترحة مخبريا وتمت ملاحظة الاداء الرصين للمنظومة في تتبع حركة اليد بالزمن الحقيقي ولاوضاع مختلفة.

الكلمات الدالة: رؤية الحاسوب، انظمة الزمن الحقيقي، تتبع الحركة، تمثيل يد الانسان، معالجة الصور الرقمية.

* قسم هندسة الحاسوب، جامعة البلقاء التطبيقية، السلط (1)؛ وقسم الهندسة الكهربائية، الجامعة الأردنية، عمان، الأردن. تاريخ استلام البحث 2003/5/28، وتاريخ قبوله 2005/19/16.