Real-Time Robot-Human Interaction by Tracking Hand Movement & Orientation Based on Morphology


School of Mechatronic Engineering, Universiti Malaysia Perlis (UniMAP)
Kampus Ulu Pauh, 02600 Arau-Perlis, MALAYSIA
asthussain@yahoo.com

Abstract—In this paper we present a method that allows real time tracking on a hand in 3D space and notes its orientation and position accordingly, with the goal of ultimately tying it to a robotic spherical wrist as well as the wrist’s 3D position. Several image processing techniques were used in conjunction with mathematical morphological filters formulae in order to understand the hand’s position and orientation. The proposed methods have showed great success in identifying the Nonlinear systems, variable 3D levels of hand movements and rotations correctly, which could be applied in different types of robotic manipulators, computer simulations or a number of human-computer remote handling interactions. Real time took place in system response.

I. INTRODUCTION

Vision based hand motion tracking is currently a very active and attractive area of research [1]. Still, the problem of identifying a hand correctly and simulating its movement completely is far from being solved. A number of algorithms that have been proposed previously used feature extraction on the hand image and then classified it by using information obtained beforehand.

Examples of such techniques include the use of Neural Networks [2], Genetic Algorithms [3], and Hidden Markov Models [4] among others. While some of these techniques have reported certain levels of success, these solutions suffer from the restrictions in preset gestures and positions, which are fed into the system beforehand.

Furthermore, several human hands to computer solutions available require invasive add-ons, such as wearing gloves [5] or targets [6]. This paper describes a method to identify the human hand in real time and analyze the resultant information so as to obtain the latest position and orientation of hands without such restrictions. While the background in this case was controlled to be plain, the results obtained showed high consistency on hands with different sizes and colours, also with the simulated movement and rotation that matched the tester’s hand.

After matching and obtaining the hand’s orientation and position in 3D space, the results can be used as an input for a computer device as part of a human machine interaction system to be one of the many available options. In our case, the hand’s position and orientation were fed right into a simulated robotic manipulator, whose end-effectors position and orientation matched the user’s hand spot-on. Furthermore, additional information from the hand, such as opening or closing the end-effectors gripper, was taken and sent directly to the simulated robot, which emulated this and other actions.

The simulated robotic manipulator used in this paper is based on the SCADA manipulator, but with an added spherical wrist as its end-effectors. Thus, a manipulator with six degrees of freedom was obtained, along with the addition feature of opening or closing its end-effectors. By using this manipulator, the user hand can guide the manipulator’s end-effectors in any point within its workspace (as x, y and z coordinates) and rotate it about any of the three principal axes in a number of degrees.

Real time refers to events simulated by a computer at the same speed that they would occur in real life. In graphics animation, a real-time program would display hand moving across the screen at the same speed that they would actually move in real life. In this application it was time delay 50 ns, which is applicable in such application.

II. HAND SEGMENTATION

Before we can use and analyze the webcam images to find out the position and orientation of the hand, we will need to segment and identify the human hand from the images. Most research used colour cues to segment the hand and isolate it
from the rest of its surrounding environment, due to the distinct colour of skin and its ease of isolation [7].

There are some recent applications of hand gesture recognition; SOMM: Self organizing Markov map for gesture recognition [8] which is a filter based technique. This is another application based on statistics technique, hand gesture recognition based on dynamic Bayesian network frame work star [9]. Some others techniques used fuzzy or neural networks, hand gesture recognition using multivariate fuzzy decision tree and user adaptation [10].

However, this has been found to be not as robust as other technical methods. In this research, a successive procedure has been applied in order to determine the human hand and its features.

Edge detection has been used in the place of colour segmentation, which showed more robust results for different hand colours and lighting conditions. Granted, a fixed white background was a restriction in segmentation, but it ensured 100% correct hand segmentation. A number of processes followed edge detection in order to obtain the human hand image in front of the camera correctly.

The camera which was used to take the video feeds was a very basic commercial webcam. The video feed ran at 30 frames per second at a resolution of 640×480. Of course, processing time render the number of frames per second much lower than that, but to achieve the real-time response of the algorithm, processing time in the system only reduced a couple of frames per second. Ultimately, this was enough for quick response and for most applicable applications.

After each frame is taken (snapshot), it is converted from the RGB spectrum into a greyscale image. This is performed by simply averaging the Red, Green and Blue intensity values in each of the snapshot’s pixels. The result is a matrix with the number of rows and columns matching the number of pixels in the snapshot. For example, for our given resolution, we will obtain a 640×480 matrix, with each element in this matrix having an intensity value matching its greyscale equivalently (which in turn is the average of the three RGB components for that pixel). Usually these intensity values are normalized to be float number in the range of 0 to 255 (to save memory usage), with a higher intensity value denoting a brighter pixel.

Taking this grey scale image, Sobel edge masking was used as a filter to process it. After the image passes through the filter, its gradient magnitude is computed as the square root of the sum of the square of each of the vertical and horizontal filter. The resultant pixels are more intense around the edges and less intense inside them. Since both vertical and horizontal Sobel edge detection masks were used, the human hand’s edges are all recognized and saved in this new image matrix. Naturally, the result is a greyscale image with high intensity at the border of the hand and its fingers, as well as high intensity at any edges or marks on the palm or outside surface of the hand. A simple thresholding was performed in order to transform this image into a black and white Boolean image, where each pixel was classified into either white / intense / object or black / non intense / background.

Changing this certain threshold gives better adaptability to lighting conditions or different coloured hands or similar differences. However, since the background is fixed as white, thresholding at a lower intensity value can give a good result from which the algorithm can continue on to detect further hand features. The end result is the hand border edges as well as some edges inside of it are identified from a binary image matrix identifying. Since the edges that lie inside the hand border are inconsequential, there is needless to sacrifice any additional processing power for their cause. However, some noise may be introduced at areas outside the hand’s border. This noise can be disposed through removing any object in the image that consists of a relatively low number of pixels using morphological opening.

Morphological opening is done by scanning through all the pixels in the image and recording which of the pixels are non-zero. This is followed by using a union-find connected components algorithm to identify which of the pixels belong to each of the objects in the image.

Finally, after all the objects in an image have been identified, each of their areas is calculated by simply counting the number of pixels it consists of. When this is done, objects that contain a number of pixels lower than the threshold will have their entire pixel value inverted, which in essence means the objects get removed.

One final process that applied on the image consists of morphological thinning of the hand object in the image, where each subsequent pixel are reduced to single pixel, and the image in its entirety is reduced to lines. It removes pixels so that an object without holes shrinks to a minimally connected stroke, and an object with holes shrinks to a connected ring halfway between each hole and the outer boundary. The thinning algorithm is as follows:

1) Divide the image into two distinct subfields in a checkerboard pattern.
2) In the first sub iteration, delete pixel \( p \) from the first subfield if and only if the conditions \( G_1 \) and \( G_2 \) are all satisfied.
3) In the second sub iteration, delete pixel \( p \) from the second subfield if and only if the conditions \( G_1 \), \( G_2 \), and \( G_3 \) are all satisfied.

**Condition \( G_1 \):**

\[
X_H(p) = 1
\]  
where

\[
X_H \sum_{i=1}^{8} b_i
\]  
\[
b_i = \begin{cases} 
1 & \text{if } x_{2i-1} = 0 \text{ and } (x_{2i} = 1 \text{ or } x_{2i+1} = 1) \\
0 & \text{otherwise}
\end{cases}
\]

\( x_1, x_2, \ldots, x_8 \) are the values of the eight neighbours of \( p \), starting with the neighbour at east and numbered in counter-clockwise order.

**Condition \( G_2 \):**

\[
2 \leq \min(n_1(p), n_2(p)) \leq 3
\]
where
\[ n_1(p) = \sum_{k=1}^{k} x_{2k-1} \lor x_{2k} \] (5)
\[ n_2(p) = \sum_{k=1}^{k} x_{2k} \lor x_{2k+1} \] (6)

**Condition G2:**
\[ (x_2 \lor x_3 \lor x_6) \land x_1 = 0 \] (7)

**Condition G2:**
\[ (x_6 \lor x_7 \lor x_8) \land x_9 = 0 \] (8)

The two sub iterations together make up one iteration of the thinning algorithm. The iterations are repeated until the image stops changing. These first couple of steps are shown in Figure 1.

![Fig. 1. Hand segmentation image progression, a) Original snapshot, b) Conversion to greyscale, c) Filtered image, d) Conversion to black and white, e) After noise removal, f) After thinning](image)

**III. HAND FEATURE EXTRACTION**

When the algorithm reaches this step, it will obtain the boundary pixels of the hand, as well as some of the interior pixels. In order to identify where the hand lies in 3D space, it is required to pinpoint a reference point on the hand. Taking the centroid of the corresponding pixels is a common procedure, but this will result in very high ill conditioning, since the number of pixels in each snapshot in the image is highly irregular. Furthermore, slight movement or rotation will result in a change of the centroid, despite the hand having a possibility of remaining stationary.

Due to these reasons, the hand’s reference point that was chosen is the point lying at the right on the middle of the wrist joint. Sharpness of hand edge detection does not affect the performance of the detection result of hand orientation. This point is shown in Figure 3.

![Fig. 2. Steps of Hand Segmentation Algorithm](image)

![Fig. 3. Proposed reference point for tracking hand movement](image)
With each snapshot taken, the algorithm tracks the hand object and identifies the reference point (this is applicable for both hands, it does not matter right or left hand usage). This point can be used to identify movement in the horizontal and vertical directions easily (denoted here as x and z axis, respectively), as the positions relative to those directions are nothing more than the co ordinations of this reference point. However, pinpointing the x and y coordinates from which the reference point is identified is not as simple as first thought.

The main idea behind identifying the reference point lies in scanning the hand object vertically and determining the width of the scanned hand horizontal line each time. The reference point lies at a vertical position containing a great difference in width of its horizontal line when compared to nearby horizontal lines in its surrounding area. Also taken into account is the fact that this reference point lies at a lower part of the segmented hand image. The algorithm checks to see where is a jump in the width of horizontal parts of segmented hand, which is the cue for reaching the wrist area.

However, it is not as simple as just described. For one, the vertical position where the largest change in width occurs can change at a very erratic pace, which would disrupt the true value of the vertical hand position, among other values. The reference point needs to remain constant for relative non movement, which demands a solution for this problem. The proposed solution in this algorithm lies in separating the vertical height of the image into designated horizontal lines, where the reference point has to lie on one of these lines. In this way, a sort of dead-band is formed, where only changes in the vertical position reference point are taken into account if they are deemed large enough, and small changes do not contribute to the result of vertical position (see Figure 4).

A second problem lies in identifying the vertical position of the reference point, that when the hand is rotated, the wrist joint ceases being horizontal. This way the scanning procedure should not be done on pure horizontal lines perpendicular to the vertical scanning operation, but rather scanning needs to be done at a slight angle. To solve this problem, after rotation about the y-axis has been found using a preliminary estimation of the reference point, an image rotation algorithm is put in place to cancel the effect of the hand’s rotation by rotating it at a negative value of the original rotation about the y-axis.

This way, the image is maintained at the upward perpendicular position, from which the algorithm continues to calculate the biggest difference in width in order to find the value of the hand’s supposed vertical position. This is shown in Figure 5. After obtaining the vertical position, the horizontal position of the reference point is calculated as the mid-point between the two ends of the wrist joint. With this, the reference point can lead to correct evaluations of the human hand’s horizontal and vertical positions in 3D space easily. While keeping the segmented hand image maintained at the upward perpendicular position, other properties may be of use as well, such as the largest width in the hand image, which will be used at a later part in the algorithm to determine the rotation feature, as well as whether or not the robot gripper should be closed.

The next feature to be obtained is the hand’s length. In order to obtain this, the farthest point on the hand from the reference point is calculated. The calculation is done by obtaining the distance in both x and y between each of the hand’s pixels and the reference point, with the pixel that has the largest distance magnitude declared the farthest away
point and thus, the tip of the hand. With this point, the algorithm has the values of the most important features of the human hand, most notably its length, height, width, vertical position and horizontal position. It is through these features that the position and orientation of the human hand in 3D space can be obtained. These features are noted in Figure 6.

IV. SYSTEM TEST

The simulated six DOF (degrees of freedom) manipulator robot controlled by hand movements and orientation, the application concentrated on hand orientation complexity (three DOF), rather than arm level and position. Other three DOF are estimated and based on hand level positions.

We did not care about other linear movements, which are done simply before e.g. forward, back ward are depending to size changing, right, and left, up and down are depending to centre of gravity.

V. APPLICATION AND RESULTS

This system been implemented and tested to tele guides a manipulator robot using morphology intelligent system via an internet connection. We have used Skype webcam connections; one side is the webcam only and the other side on other computer system fitted with a Skype and a morphology program and a simulated robot to act as a real robot according to the hand movement’s state on the first point.

Matlab 2010 been used, PC is core 2 duo 1.8 GHZ, 4GB RAM and internet connection speed is 8 MB/S.

First application: a tele guided intelligent operated manipulator system; we have used 2 Skype accounts IDs over USA server to make sure that the tele operation is done over the global. The response depends to the speed of the internet connection speeds on both sides. Fig (9) shows how the application took place and achieved successfully.

Second application: local guided intelligent operated manipulator system is done on the same computer system, it gives the same results in terms of accuracy and tele control but this time it is faster than in first application, it is done in real time, due to the application took place on the same machine of computerized simulated manipulator system. Fig (10) shows how the application took place successfully.
VI. CONCLUSIONS

The simulated manipulator robot has been controlled successfully in real time by using morphing technique which is based on non-linear hand orientation. This application satisfied real time deadline.

The system has been implemented and tested for both a tele and local guided intelligent operated manipulator systems successfully.

REFERENCES


