Hand gesture recognition and tracking

Submitted by:

Gabriel Mishaev
308639202

Evgeny Katzav
306532664

Supervisor:

Arie Nakhmanai

Winter 2008
**Table of Contents**

Abstract .................................................................................................................................................. 4  
Introduction ......................................................................................................................................... 5  
Working environment and utilized tools ............................................................................................ 7  
General work ........................................................................................................................................ 8  
Algorithm explanation main subjects ............................................................................................... 9  
  Part 1 – Creation of hand mask ........................................................................................................ 9  
    1. A: Color segmentation ................................................................................................................. 9  
    1. B: Noise filtering ....................................................................................................................... 13  
  Part 2 – Hand model and parameters ............................................................................................. 15  
    2. A: Creation of a model template ............................................................................................... 17  
  Part 3 – Gesture recognition ........................................................................................................... 24  
    3. A: Finger counting .................................................................................................................... 25  
    3. B: Comparison using minimum distance function .................................................................... 27  
  Part 4 – Tracking hand movement .................................................................................................. 30  
Statistical results ................................................................................................................................... 33  
Conclusions .......................................................................................................................................... 35  
Future aspects and ideas .................................................................................................................... 36  
Acknowledgments .................................................................................................................... 37  
Bibliography ........................................................................................................................................ 37  
Appendix .............................................................................................................................................. 38  
  The HSV color space ....................................................................................................................... 38
Table of Figures

Figure 1: Working area ........................................................................................................ 7
Figure 2: General flow chart ................................................................................................ 8
Figure 3: Sample picture used for color segmentation ......................................................... 10
Figure 4: The picture in the HSV color space ................................................................. 10
Figure 5: Initial mask after color segmentation ................................................................. 11
Figure 6: Edge detection ................................................................................................... 12
Figure 7: Mask of the hand .............................................................................................. 12
Figure 8: The original picture with its noisy mask ............................................................ 14
Figure 9: The filtered mask ............................................................................................... 14
Figure 10: Different hand models .................................................................................... 15
Figure 11: The hand silhouette ......................................................................................... 16
Figure 12: Skeleton of the hand silhouette ....................................................................... 16
Figure 13: Hand at a certain angle .................................................................................. 17
Figure 14: The vertical hand after rotation ....................................................................... 17
Figure 15: Hand palm after forearm removal .................................................................... 18
Figure 16: Normalized hand ............................................................................................ 18
Figure 17: Skeleton template ........................................................................................... 19
Figure 18: Skeleton with found end points ....................................................................... 19
Figure 19: Table of gestures ........................................................................................... 23
Figure 20: Example of a correct 3 fingers recognition ..................................................... 24
Figure 21: Counted fingers in different gestures ............................................................. 26
Figure 22: Improvement of circle counting ....................................................................... 26
Figure 23: Example of distance calculation ...................................................................... 27
Figure 24: V sign and rock gestures ................................................................................ 28
Figure 25: Fist recognition .............................................................................................. 29
Figure 26: The hand in blue, center mass in green ............................................................ 30
Figure 27: The hand in blue, center mass in green, index finger in red............................ 31
Figure 28: The wind rose used for direction determination .............................................. 31
Figure 29: Index finger heading west .............................................................................. 32
Figure 30: Recognition .................................................................................................... 34
Figure 31: Future reality .................................................................................................. 36
Figure 32: HSV color wheel allows the user to quickly select a multitude of colors .......... 39
Figure 33: HSV cone ........................................................................................................ 39
Abstract

The main goal of our project was to create a computer program which tracks the hand movement of the user from the captured video by a camera, and recognizes his hand gesture. The program identifies the human hand isolates it then counts the number of fingers shown on screen and finally describes the hand gesture which it recognized. The program also shows the direction of the moving hand and its general position on the screen. The purpose of all this things is to create an interface device for human-computer interaction.
Introduction

Although many of us use the mouse and keyboard as if it was the extension of our hands, it is still an unnatural and pretty complicated way to communicate. Some sort of a sign language with our hands comes to mind, however the problem is that the other part of this conversation does not understand your hand waving so easily. So for all humans who talk with their hands the next project tries to suggest a solution.

In this line of work a video is captured by a simple web cam, where a person moves his hand and displays different hand gestures. A vocabulary of 9 positions was created which the computer understands. The gestures differ in the number of raised fingers (some gestures have the same amount of fingers), and each gesture has its own meaning. In the project an image appearance based model was used to recognize and identify every gesture. As it was mentioned above the computer does not understand by itself the content of an image. Another obstacle we encountered is that the conditions were not permanent; after all we shot a video which is a series of pictures and not a single image. The shape of the hand and its angle changes during the movement and the whole position of the hand too, the hand is not stationary. Some light settings even affected the results.

We have tried to find a simple way of resolving the problems, so we divided the recognition process into several steps. First the hand is extracted from the video image using color segmentation techniques. From this point we calculated the number of raised fingers and a skeleton model of the hand gesture was created. From the parameters we obtained the program identifies the correct gesture. The program also tracks and follows the hand. The center of mass coordinates of the hand are presented and also of the index finger if it is raised. The whole work is done in matlab environment because it has a large variety of tools which we used to achieve our goal.

The basic idea for this project comes from works done in previous years. One of them used a color labeled glove to help recognizing the gestures, in our case the hand is unarmed with any artificial device. So it is more convenient for the user but makes the recognition task harder. Another project which related to us also tried to recognize gestures but it only counted the number of fingers, in our project a base for a simple
language was established with the gestures as words and some words have the same letters (same amount of fingers), we also used completely different ways to recognize the gestures. For example when counting the fingers we used a simple circle that passes across the fingers and counts them. We feel that the solutions we offer are easier to understand and to implement.
Working environment and utilized tools

The video capturing was done by a simple web cam that can be found at any home. We analyzed the video using the matlab program. The surroundings resembled a working area of a person sitting near a computer in his room:

Figure 1: Working area
General work

Before getting into the inner parts of our algorithm and software. We would like to describe the general scheme and principals of our work. A good representation of the system that has been built can be seen in figure 2. We think that a system of such sort, can suit a great part of all hand gesture recognition and tracking algorithms. The input data is a video, in the analysis stage we use color segmentation, and other common techniques used in image processing and analysis. Our model parameters and space classes are the hand binary mask, and the skeleton image of the hand. From that stage we move on to the recognition stage, where with the use of model templates (you can think of it as grammar) we compare and find the correct gesture. We feel that in that stage, we gave our major contribution and improvement compare to pervious projects. Finally the gesture position and recognition, were sent as an output frame picture, with all the collected data. This is only a brief description of our software, now we will proceed with a thoroughly explanation for each and every stage.

![General flow chart](image-url)

Figure 2: General flow chart
Algorithm explanation main subjects

Part 1 – Creation of hand mask

Our main objective at first was to locate and isolate the hand from all the other objects in the frame picture. We tried to eliminate the other objects in the image such as the face of the person, the background, and anything else which is not the hand. To do so, two possible ways were thought of. The first was using the shape and size of the hand. However, these criteria change a lot based on the user location and during hand movement, so we couldn’t rely on them. The second obvious way is using the color of the human hand. But before we could start working with video, we had to understand how the computer works with images, since the video is actually a series of images

1. A: Color segmentation

The image is stored in the computer as a grid of pixels. Each pixel has a value in the RGB color space, giving us the dimensions of the image as 320x240x3 (for example). The pixel has a (x, y) coordinates which are the position of the pixel in the picture. However, we have found that the RGB color space doesn't separate well the hand from the background.

For the segmentation and the creation of a silhouette of the hand called “mask”, the HSV color space was chosen, because it has proved to be very effective distinguishing the hand in the frame. During our work, an attempt with the YCBCR color space has also been made. After a few trials we concluded that the HSV is better, because it produced better results in the detection and segmentation of the hand (More information and explanation about the color spaces is given in the appendix).
Let us see this work in progress: first, a picture of one of us has been taken –

![Sample picture used for color segmentation](image)

**Figure 3: Sample picture used for color segmentation**

Then it was translated to the HSV color space:

![The picture in the HSV color space](image)

**Figure 4: The picture in the HSV color space**
As it is shown it the figure above the values of the human skin (hand and face) are very different from anything else. This fact was used to determine which pixel belongs to the hand and which doesn’t. A threshold for the hue and saturation were set so the hand pixels would remain white while the other pixels would be black. The value space didn't help us determine whether the pixel is human or not, so we didn't use it in the threshold. So, the actual mask was created:

![Initial mask after color segmentation](image)

**Figure 5: Initial mask after color segmentation**

Notice: the thresholds must be set once at the beginning. They vary from one surrounding to the other, depending on the colors appearing in the room, light settings (whether it is full sun light or florescence light) and even on the color of each individual hand. So in order to make the correct segmentation, the proper thresholds should be found. It is a fairly simple procedure that needs to be done once at each new place. Incorrect thresholds affect the results.

As shown in figure 5, the mask is not complete. The head has not been removed yet. To separate the hand we also used segmentation based on edge detection. In later stages we also used the fact that when sitting in front the computer/webcam, the hand takes a bigger area of the screen than the head.
Small holes in the hand were filled with the `imfill` function. The median filter (`medfilt2`) was also used to lower the noise levels.

All those actions are performed by the `get_hand` function which was written by us. The code for it appears in the end. The function receives as an input the colored image, and returns the output which is a black and white mask of the hand.
1. B: Noise filtering

After we got the hand mask, we noticed that the image came with some noise. In some cases the hand was distorted especially near the edges. Due to the aggressive cut by the thresholds some pixels were identified incorrectly, giving us white spots where there was hand and vice versa. As mentioned before, some filtering methods like filling the image and median filter were used. Apparently it wasn’t enough. In order to give the mask a more hand like appearance, morphological operations were applied. Those operations included erosion and dilation of the white object, using a small disk. In matlab they came to implementation in the form of the \texttt{imopen} and \texttt{imclose} functions.

After this, a filtering using a low pass Gaussian filter was made. It really helped to smooth the edges, and to eradicate a lot of the noise which came from the camera and the segmentation process. The size and the width of the filter were determined based on trial and error until we managed to get satisfying results.

At first we thought that a band pass filter would be required, but its results were similar to the low pass, so we preferred to stay only with the low pass because it is easier to make and use.

A final, more brutal operation was made: the elimination objects whose area was below a certain limit. We did this because of the understanding that by now the hand area should be large, and the other small objects can not be candidates for the hand. So they had to be removed before any future processing and analyzing of the hand image could be done. The function we used to do so was bwareaopen.

To summarize all the noise filtering process, we show the figures below. First the original picture is shown and it's noisy mask (figure 8), Then the final mask of the hand (figure 9):
Figure 8: The original picture with its noisy mask

Figure 9: The filtered mask
Our next objective was to start recognizing the gestures appeared in the image. Before any evaluation could start, a model of the hand and its parameters had to be decided. Several options stood before us. One was to create and use a 3D model of the hand, which is the most advanced method and the one we could extract the most information regarding the hand state. However, this model is difficult to produce from one camera and a simple image. The 3D model was far above our needs and requirements of tracking the movement and recognizing predefined postures.

The second option for the hand model was the binary silhouette- which already been created. From this model we get information about from the image geometry parameters. The regionprops function was used to find out the features of the hand, like: size (area), angle (orientation), shape (eccentricity, solidity), position (centroid). The binary model did help us and we used it to count fingers, but for the purpose of full recognition it was not sufficient. For that goal we chose the 2D skeleton model for the hand from which we found the fingertips positions and other parameters. We created templates of all the gestures. Then a comparison between the input skeleton and the templates was made using a certain distance map. The gesture with the minimum distance was chosen.

**Figure 10: Different hand models**
Here we can see the mask and its skeleton output:

Figure 11: The hand silhouette

Figure 12: Skeleton of the hand silhouette
2. A: Creation of a model template

In order to make a good comparison, a template must be made. We could not use the image coordinates as they were, because the hand can move in almost every direction and also come near or far from the camera. First of all, we rotated the hand so the hand axis would be parallel to x and y. By this way the palm of the hand would always be in a straight position. We used the function `imrotate` to do so:

![Figure 13: Hand at a certain angle](image)

![Figure 14: The vertical hand after rotation](image)
Secondly, we wanted the center of mass in the center of the palm. So we had to exclude the forearm:

![Figure 15: Hand palm after forearm removal](image)

After that we took the bounding box of the image, and resized it to 200x150, the size we chose for the template. That way all the hand masks would be centralized and normalized:

![Figure 16: Normalized hand](image)
Now the skeleton model was produced by using `bwmorph` function. Basically, this function thins object into lines. Applied on a circle, the output will be a single dot – the circle center. In our case, it thinned the mask until the round palm and fingers were turned into lines.

![Skeleton template](image)

**Figure 17: Skeleton template**

Following that we were able to find the end points of the skeleton, which are the fingertips. We used the `find_skel_ends` function made by someone.

![Skeleton with found end points](image)

**Figure 18: Skeleton with found end points**

Every frame picture went through the same procedure, which was described above. For all the 9 gestures which we intended to recognize, a model template was constructed and the end points were found. This data was saved in the `tp_all` file.

In the next pages we bring you a table of those 9 gestures, their names, pictures and skeletons. Those skeletons are meant to be the perfect representation, for each hand gesture in our space model.
<table>
<thead>
<tr>
<th>Gesture name</th>
<th>Picture of the hand posture</th>
<th>Skeleton of the gesture</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Image 1</th>
<th>Image 2</th>
<th>Image 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open palm</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td>Four fingers</td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td>Three fingers</td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
</tr>
<tr>
<td>Ninja turtle</td>
<td><img src="image" alt="Ninja turtle" /></td>
<td><img src="image" alt="Ninja turtle" /></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------</td>
<td>-----------------------</td>
<td></td>
</tr>
<tr>
<td>Alright</td>
<td><img src="image" alt="Alright" /></td>
<td><img src="image" alt="Alright" /></td>
<td></td>
</tr>
<tr>
<td>V sign</td>
<td><img src="image" alt="V sign" /></td>
<td><img src="image" alt="V sign" /></td>
<td></td>
</tr>
<tr>
<td>Gesture</td>
<td>Image</td>
<td>Animation</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>-------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>Rock</td>
<td>![Rock Image]</td>
<td>![Rock Animation]</td>
<td></td>
</tr>
<tr>
<td>Index finger</td>
<td>![Index Finger Image]</td>
<td>![Index Finger Animation]</td>
<td></td>
</tr>
<tr>
<td>Fist</td>
<td>![Fist Image]</td>
<td>![Fist Animation]</td>
<td></td>
</tr>
</tbody>
</table>

Figure 19: Table of gestures
Part 3 – Gesture recognition

Finally, at this stage we actually began extracting gestures from the image. The procedure was divided into two functions. One counts the number of fingers in the silhouette hand image called `count_fingers`. The second matches between the templates, which were previously constructed using the skeleton and its end points, and gives us the template which resembles the most to the input image. This function is called `match_temp`. Although the two functions do not depend one on another, it was decided to combine the outcomes to achieve better identification results. That way, by knowing how many fingers are in the current image, we can reduce the number of templates to choose from in the `match_temp` part. By doing that, we avoid unnecessary calculations, reduce computational time, and decrease the chance for error. For example: when the function counts 5 fingers, it is obviously the open palm gesture. However, since the algorithm is not 100 percent fool proof, we compared it with the template of 5 fingers and 4 fingers, but not with the rest, thus achieving more accurate results. In gestures with identical number of fingers, it is absolutely necessary to use both methods. We shall now explain about the two functions, why they were chosen and how do they work.

In the next figure you can see an example of correct recognition:

Figure 20: Example of a correct 3 fingers recognition
3. A: Finger counting

In this section we will explain about the `count_fingers` function. This function receives the processed hand mask (silhouette), with its geometric parameters. The function returns the number of raised fingers in this image. We chose to implement this function by drawing a circle over the mask, and counting the number of passages between black and white, which gives us an indication of how many fingers are raised. It may be not the best way to count fingers but it is definitely very simple and effective.

The circle equation is as known \((x-a)^2 + (y-b)^2 = R^2\). In our case, \((a,b)\) are the coordinates of the palm center, and \(R\) is the radius. After several trials, it was decided to be about 2/3 of the minor axis length. For convenience, we worked in the spherical coordinates, meaning:

\[
\begin{align*}
x &= R \cos \theta + a \\
y &= R \sin \theta + b
\end{align*}
\]

Now, if the circle is over the hand area, its value is 1 and otherwise 0. Then the number of changes in the circle is calculated and from it subsequently the number of raised fingers. As mentioned before this way of counting is not the most robust, because it is affected from noise and other distortions in the mask. However, we were able to overcome these problems. For example, a passage which is less than the average width of a finger was not taken into the sum. The boundaries of the image were also dealt with, so the circle will not go out of bounds. A more specific treatment was made to deal with the fist image since it sometimes returned poor results. So it this case we relayed on the geometric features meaning the width and shape of the gesture to determine when zero fingers were raised.
Here we can see an example of correct finger count:

**Figure 21: Counted fingers in different gestures**

As you can see in the figure above, a "false passage" which is less than the width of a finger is not taken into consideration, so the overall count is still correct.

**Figure 22: Improvement of circle counting**
3. B: Comparison using minimum distance function

As we have seen before, knowledge about the number of fingers is not enough to determine the correct gesture. Therefore a way of correlating between the templates of the gestures, and the skeleton points we received from each frame were thought of. Since simple subtraction of the image minus the template does not yield good results, a distance function was calculated, giving us the likelihood that the current skeleton is the template we compared it with. We used the quadratic chamfer distance function: when we have a set of template points \( A = \{a_i\}_{i=1}^{N_A} \), and a set of points coming from the current frame skeleton \( B = \{b_i\}_{i=1}^{N_B} \), the quadratic chamfer distance is given by the average of the squared distances between each point of \( A \) and its closest point in \( B \).

This likelihood in mathematical term is: 

\[
d(A,B) = \frac{1}{N_A} \sum_{a \in A} \min_{b \in B} \|a - b\|^2.
\]

The function `match_temp` receives a set of both points and returns this likelihood by calculating the Euclidean distance between the points. The smaller this “distance” is, the more it means that the picture is like the template, and since we know what gesture this template stands for we can easily recognize the gesture.

![Figure 23: Example of distance calculation](image)

As shown in figure 23 only the red line will be taken to the sum calculation, because when the templates are on the same grid it is the minimal distance. The same is done for all the other points.

Now we have a method to compare different gesture to one another. It was used to distinguish between gestures that had the same number of fingers, like in the case of 3 fingers, ninja turtle, alright (which all have 3 fingers raised).
**Special treatments:** Some gestures were easier to recognize than others. The "v sign" got mixed up with the "rock" gesture more than once, which is not surprising due to the fact that they have a very similar template. We tried to determine the right gesture by measuring the distance between the two raised fingers in the frame. We set a certain threshold; above it the gesture was detected as "rock" otherwise "v sign".

![Figure 24: V sign and rock gestures](image)

As shown above the distance between the index finger and the pinky in the rock gesture, blue line on the left, is always larger than the distance of the index finger and the middle one in the v sign, green line on the right.

The "fist" gesture required a special treatment in the `count_fingers` function. We noticed that more than once, when the gesture was "fist", the function returned 1 finger up. Because "fist" and "index finger" has practically identical template, it was almost impossible to distinguish between them. Therefore, we checked the "finger" length above the palm area. If it was small or zero, we determined that it was "fist" and so we returned zero fingers rose. It has also been noticed that the fist gesture had a very high solidity value compared to the rest, because of its round shape with no fingers.
Figure 25: Fist recognition

Despite the skeleton not being much of a help in this case, and the difficulties of finger counting in this case, the gesture is still recognized correctly.
**Part 4 – Tracking hand movement**

Hopefully, at this stage we already recognized the correct hand gesture. Now a tracking of the hand movement can take place. By tracking we mean to give the location of the hand compared to other objects in the background, and also show the direction in which the hand moved. Since we already isolated the hand from its surroundings in the color segmentation stage, its location is shown simply by pin pointing its center of mass in green. Also, its contour is made visible so you can really see the gesture that was recognized, as we can see in the next figure:

![Image of hand tracking](image)

**Figure 26: The hand in blue, center mass in green**

Another feature is the finding of the position of the index finger in gestures in which it appears. It was done by calculating the angle between the coordinates of the fingertips and the center of mass, based on the fact that the arm is rotated and now straight, and also remembering that the movement of the finger is constrained. After a few experiments we have found that the index finger is tilted about 20º from the center mass. It was pretty easy to identify which point belonged to the index finger and so it was also marked by red. The idea behind this was that the index finger will simulate some sort of an indicator like a mouse cursor.
Here we can see an example with the index finger marked:

![Image of hand with marked finger]

Figure 27: The hand in blue, center mass in green, index finger in red

As mentioned before, another goal was to follow the hand and show its direction of movement. Here we no longer use a single image but rather the difference between two sequential frames. Since in each frame the center of mass was found, we calculated the difference in its position, resulting in the vector:

\[
\begin{align*}
X &= x_{new} - x_{old} \\
Y &= y_{new} - y_{old}
\end{align*}
\]

This vector is actually the movement direction. By calculation its angle we know which direction the hand is headed. We sorted the general movement into 8 specific ones according to the wind rose, and one state of no movement if the had moved less than a previously set threshold.

![Wind rose diagram]

Figure 28: The wind rose used for direction determination
Here we can see a series of 3 sequential figures with the index finger heading west:

Figure 29: Index finger heading west
Statistical results

We have come far along down the road, and now it is time to put our algorithm in test. We had two possibilities to do so. One is to do a live trial, using the matlab image acquisition toolbox and a simple webcam. The other option was to record a movie with a camera, copy it to the computer and then analyze it with matlab. We tried both ways, and got about the same results, but in order to properly analyze the data, we shall present the recorded video option.

At first, we used a digital camera and recorded a short movie containing all the gestures, right here in the lab. The video is in the VGA format, with 320*240 image resolution. It is about 40 seconds long with 15 frames per second captured, giving us 600 frames of hand gestures to recognize. In some of the frames there is no clear hand gesture to recognize, especially when the gesture is changed, so we excluded those frame from our results. The recording was done on a changing white background, and florescent light combined with sunlight coming from windows, when the head and some of the user's body was in the frame. These conditions were used to deliberately make the recognition process harder. Then we moved the video to the hard drive and used the mmreader function in matlab to read it. Now, all that was left is to check the number of correct/false recognitions in the video. We will summarize the results into a table:

<table>
<thead>
<tr>
<th>Gesture name</th>
<th>total appearances</th>
<th>correct recognition</th>
<th>false recognition</th>
<th>percentage of correct recognition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open palm</td>
<td>55</td>
<td>45</td>
<td>10</td>
<td>81.81818182</td>
</tr>
<tr>
<td>Four fingers</td>
<td>62</td>
<td>60</td>
<td>2</td>
<td>96.77419355</td>
</tr>
<tr>
<td>Three fingers</td>
<td>44</td>
<td>38</td>
<td>6</td>
<td>86.36363636</td>
</tr>
<tr>
<td>Ninja turtle</td>
<td>93</td>
<td>76</td>
<td>17</td>
<td>81.72043011</td>
</tr>
<tr>
<td>Alright</td>
<td>50</td>
<td>40</td>
<td>10</td>
<td>80</td>
</tr>
<tr>
<td>V sign</td>
<td>50</td>
<td>40</td>
<td>10</td>
<td>80</td>
</tr>
<tr>
<td>Rock</td>
<td>43</td>
<td>37</td>
<td>6</td>
<td>86.04651163</td>
</tr>
<tr>
<td>Index finger</td>
<td>75</td>
<td>68</td>
<td>7</td>
<td>90.66666667</td>
</tr>
<tr>
<td>Fist</td>
<td>47</td>
<td>39</td>
<td>8</td>
<td>82.9787234</td>
</tr>
<tr>
<td>Total</td>
<td>519</td>
<td>443</td>
<td>76</td>
<td>85.35645472</td>
</tr>
<tr>
<td>Index finger tip</td>
<td>422</td>
<td>390</td>
<td>32</td>
<td>92.41706161</td>
</tr>
<tr>
<td>Hand contour</td>
<td>519</td>
<td>495</td>
<td>24</td>
<td>95.37572254</td>
</tr>
</tbody>
</table>
As we can see, the overall results are very good. We have got about 85% correct recognitions in total, and let us not forget that the recording conditions weren’t easing. At first we were afraid that we will have problems with the head being in the frame, but the algorithm has managed to isolate the hand from the background almost in every frame.

Given here is an example for a correct recognition of each gesture:

Figure 30: Recognition
Conclusions

In this work we have showed a way to successfully track and recognize a known set of hand gestures previously defined. This was made by isolating the hand in the frame using skin color characteristics in the HSV color space. Then we constructed a normalized filtered mask of the hand silhouette. We counted the number of fingers in that silhouette, so we will know which templates to compare it with. The templates are the ideal masks of the gestures we worked with. The template with the minimum chamfer distance was chosen as the current gesture. Finally, we present a figure featuring the hand with its contour, with the center mass and the index finger tip (if present) marked, and we also specify the hand's movement direction, using the previous figure.

The algorithm was designed to work with live stream video using Matlab's image acquisition toolbox or with video files saved in the hard drive.

Overall, the results are satisfying, with over 85% of correct recognitions.

Our work can be expanded to support more gestures in a simple way. If one wants another gesture to be taken into question, all one need to do is just add its template to the rest of the templates and modify a bit the code to make sure that the new gesture is chosen when it gets the minimum distance.
Future aspects and ideas

While working on the project we understood the huge potential of systems like ours. We believe that in the future, HCI components would be replaced with more natural operating systems, to ease the connection between man and machines. We know that our software is very basic, and there is room for improvement. Both by making our code more efficient, and by adding more complex gestures to the vocabulary. A crucial step must be implementing the system in real time. Meaning it has to be translated to C or C++ and see how it works. After that it could be implied as a mouse, which was the original idea for this system. For example the index gesture could be the cursor, and the open palm will mean open a file, if followed be fist it would mean close the file. Many other applications can come also in mind. As we see it our algorithm is pretty much straight forward and deterministic. Its evolution can be as a learning process, where it gets a series of pictures of hands (test samples), learns their parameters and classifies them automatically. We know that classification algorithms already exist, like K-means and SVM classifier. A combination of image recognition and those learning algorithms should be explored. In the department of tracking movement, we only found out where did the hand gone. We think it is also possible to estimate where the hand will move in coming frames, using practical filters like kalman filter. For conclusion our project is only the beginning for many applications to come, however we shown that what was thought in previous years as science fiction is the reality. In one sentence the future is now.

Figure 31: Future reality
**Acknowledgments**

We would like to thank Arie Nakhmani who guided us in this project. We also want to give our appreciation to the control and robotics lab staff- Koby Kohai and Orly Wigderson for all their help.

**Bibliography**

* Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review.
  By Vladimir I. Pavlovic, Rajeev Sharma and Thomas S. Huang.
  IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 19, NO. 7, JULY 1997
Appendix

The HSV color space

HSV is an alternative representation of points in an RGB color space, which attempt to describe perceptual color relationships more accurately than RGB, while remaining computationally simple. HSV stands for hue, saturation, value.

The HSV space describe colors as points in a cylinder whose central axis ranges from black at the bottom to white at the top with neutral colors between them, where angle around the axis corresponds to “hue”, distance from the axis corresponds to “saturation”, and distance along the axis corresponds to “value”.

The HSV can be thought of conceptually as an inverted cone of colors (with a black point at the bottom, and fully-saturated colors around a circle at the top).

HSV is a simple transformation of device-dependent RGB, the color defined by a (h, s, v) triplet depends on the particular color of red, green, and blue “primaries” used. Each unique RGB device therefore has unique HSV space to accompany it. An (h, s, v) triplet can however become definite when it is tied to a particular RGB color space, such as sRGB.

This model was first formally described in 1978 by Alvy Ray Smith (though the concept of describing colors in three dimensions dates to the 18th century).[1][2]

Why do we even use the HSV color space? It is sometimes preferable in working with art materials, digitized images, or other media, to use the HSV color model over alternative models such as RGB or CMYK, because of differences in the ways the models emulate how humans perceive color. RGB and CMYK are additive and subtractive models, respectively, modeling the way that primary color lights or pigments (respectively) combine to form new colors when mixed.
Figure 32: HSV color wheel allows the user to quickly select a multitude of colors

The HSV model is commonly used in computer graphics applications. In various application contexts, a user must choose a color to be applied to a particular graphical element. When used in this way, the HSV color wheel is often used. In it, the hue is represented by a circular region; a separate triangular region may be used to represent saturation and value. Typically, the vertical axis of the triangle indicates saturation, while the horizontal axis corresponds to value. In this way, a color can be chosen by first picking the hue from the circular region, then selecting the desired saturation and value from the triangular region. The conical representation of the HSV model is well-suited to visualizing the entire HSV color space in a single object.

Another visualization method of the HSV model is the cone. In this representation, the hue is depicted as a three-dimensional conical formation of the color wheel. The saturation is represented by the distance from the center of a circular cross-section of the cone, and the value is the distance from the pointed end of the cone. Some representations use a hexagonal cone, or hexcone, instead of a circular cone. This method is well-suited to visualizing the entire HSV color space in a single object; however, due to its three-dimensional nature, it is not well-suited to color selection in two-dimensional computer interfaces.

Figure 33: HSV cone