IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 1, January 2015.
www.ijiset.com
ISSN 2348 – 7968

Hand Gesture recognition Using Neural Network

Bhushan Bhokse[1], Dr. A.R.Karwankar[2]
[1] Department of Electronics Engineering, Government College of Engineering, Aurangabad, Aurangabad, Maharashtra, India.

[2] Department of Electronics Engineering, Government College of Engineering, Aurangabad, Aurangabad, Maharashtra, India.

Abstract
A primary goal of hand gesture recognition research is to create a system which can identify specific human hand gestures and use them to convey information or for device control. Gestures are physical positions or movements of a person's fingers, hands, arms or body used to convey information. Gesture recognition is the process by which gestures formed by a user are made known to the system. There are different methods of Gesture Recognition. Here we are using Neural Network for hand Gesture recognition.

When speaking about image recognition or sign classification, the most widespread solution is the neural network. A program is developed in MATLAB for neural network for recognizing the number of the fingers in front of web camera. It's a highly efficient method that has been proven able to distinguish and classify with an amazing rate of performances. Then, it will be possible to train the network with a full set of examples and finally to use in real time conditions.

Keywords: *Hand gesture recognition, image processing, neural network, sign classification, Matlab*

1. Introduction

Since the introduction of the most common input computer devices not a lot have changed. This is probably because the existing devices are adequate. It is also now that computers have been so tightly integrated with everyday life, that new applications and hardware are constantly introduced. The means of communicating with computers at the moment are limited to keyboards, mice, light pen, trackball, keypads etc. These devices have grown to be familiar but inherently limit the speed and naturalness with which we interact with the computer.

In recent years there has been a great deal of studies aimed at the inconvenience of human computer intercommunication tools such as keyboard & mouse. As one of the alternative gesture recognition methods have been developed by which a variety of commands can be used naturally. Since conventional input devices need a great deal of technical education, many researches feel a great interest in & attach importance to hand gesture recognition. In the present day, framework of interactive, intelligent computing, an efficient human –computer interaction is assuming utmost importance in our daily life.

Gestures are physical positions or movements of a person's fingers, hands, arms or body used to convey information. Hand gestures, i.e., gestures performed by hand. Gesture recognition is the process by which gestures formed by a user are made known to the system.

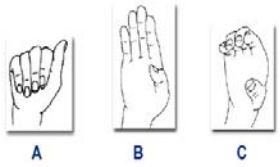


Fig 1. Hand Gestures

A pattern recognition system will be using a transform that converts an image into a feature vector, which will then be compared with the feature vectors of a training set of gestures. The final system will be implemented with a neural network.

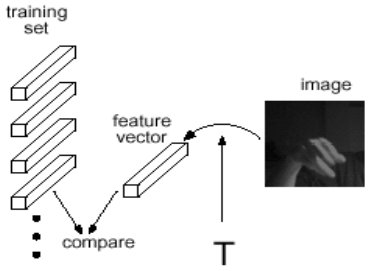


Fig 2 Pattern recognition

As the computer industry follows Moore's Law since middle 1960s, powerful machines are built equipped with more peripherals. Vision based interfaces are feasible and at the present moment the computer is able to "see". Hence users are allowed for richer and user-friendlier man-machine interaction. This can lead to new interfaces that will allow the deployment of new commands that are not possible with the current input devices. Plenty of time will be saved as well.

Recently, there has been a surge in interest in recognizing human hand gestures. Hand gesture

ISSN 2348 – 7968

recognition has various applications like computer games, machinery control (e.g. crane), and thorough mouse replacement. One of the most structured sets of gestures belongs to sign language. In sign language, each gesture has an assigned meaning (or meanings).

Computer recognition of hand gestures may provide a more natural-computer interface, allowing people to point, or rotate a CAD model by rotating their hands. Hand gestures can be classified in two categories: static and dynamic. A static gesture is a particular hand configuration and pose, represented by a single image. A dynamic gesture is a moving gesture, represented by a sequence of images. We will focus on the recognition of static images. Interactive applications pose particular challenges. The response time should be very fast. The user should sense no appreciable delay between when he or she makes a gesture or motion and when the computer responds. The computer vision algorithms should be reliable and work for different people. There are also economic constraints: the vision-based interfaces will be replacing existing ones, which are often very low cost. A hand-held video game controller and a television remote control each cost about $40. Even for added functionality, consumers may not want to spend more. When additional hardware is needed the cost is considerable higher. Academic and industrial researchers have recently been focusing on analyzing images of people. While researchers are making progress, the problem is hard and many present day algorithms are complex, slow or unreliable. The algorithms that do run near real-time on computers those are very expensive relative to the existing hand-held interface devices.

2. Literature Review

In recent years there have been great deals of studies aimed at the inconvenience of human computer intercommunication tools such as keyboard & mouse. As one of the alternative gesture recognition methods have been developed by which a variety of commands can be used naturally. Since conventional input devices need a great deal of technical education, many researches feel a great interest in & attach importance to hand gesture recognition. In the present day, framework of interactive, intelligent computing, an efficient human –computer interaction is assuming utmost importance in our daily life. Gesture recognition can be termed as an approach in this direction [1]. The major tools used for this purpose includes HMMs [2] & ANNs [3].HMMs tool basically deals with the dynamic aspects of the gestures. Gestures are extracted from a sequence of video images. But the major disadvantage of HMMs is that it is based on probabilistic framework [4]. For large data sets ANNs have been used for representing & learning the gesture information. ANNs is mostly used for recognizing a static posture.

3. Objective of the Work

The goal of this research work is to develop a program implementing gesture recognition. At any time, a user can exhibit his hand doing a specific gesture in front of a video camera linked to a computer.

The program has to collect pictures of this gesture, analyze it and to identify the sign. In order to lighten the project, it has been decided that the identification would consist in counting the number of fingers and recognition of American Sign Language that are shown by the user in the input picture.

4. System Development

4.1 Experimental set-up

The experimental setup consists of:

1. Digital camera: To take the images. The camera is interfaced to computer.
2. Computer: Computer is used to create the database & analysis of the images. The computer consists of a program prepared in MATLAB for the various operations on the images. Using Neural Network tool box, analysis of the images is done.

4.2 Block Diagram

The Conceptual block diagram of the work is shown below.

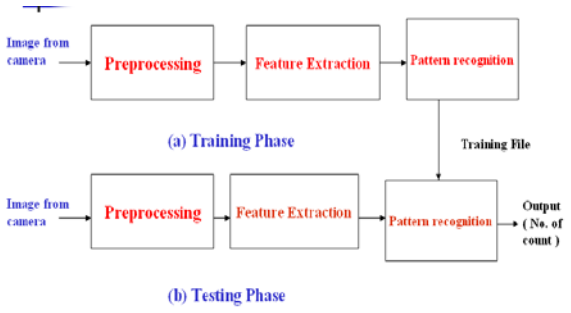


Fig 3. Functional block diagram

a) Image creation: The images are taken by the camera. Camera is interfaced to the computer.
b) Preprocessing: It is used to filter out noise from the image. Image captured by the camera contains noise like impulse noise & Salt-pepper type noise. Salt –pepper noise is one of the types of the

ISSN 2348 – 7968

saturated impulse noise. This type of noise is removed by using Median filter.

c) **Feature Extraction:** In this step the required features are extracted from the image. The following steps are used:

- ➔ Subtraction: In background subtraction method, the image of background is captured as a reference image. The plain background image is subtracted from the image having hand with background that is the Target image. The resulting image is only hand. This is the simple & easiest method of extraction.
- ➔ Segmentation: Thresholding is used for clarity. Thresholding by Hysterisis method is used. The hysterisis will give the information regarding which point is to selected as the threshold. Also it will convert the image into the binary form. So that it will be easy for further processing.
- ➔ Thinning: Thinning is a operation that is used to remove selected foreground pixels from binary images. In this object is converted into the set of digital arcs. These arcs are lying roughly along the medial axis. That is it will give reduced amount of the data, reducing the time required for processing. The border pixel having more than one neighbor is removed, converting into the thin line. It can be used for several applications, but is particularly useful for skeletonization. In this mode it is commonly used to tidy up the output of edge detectors by reducing all lines to single pixel thickness. Thinning is normally only applied to binary images, and produces another binary image as output.

d) **Pattern Recognition:**

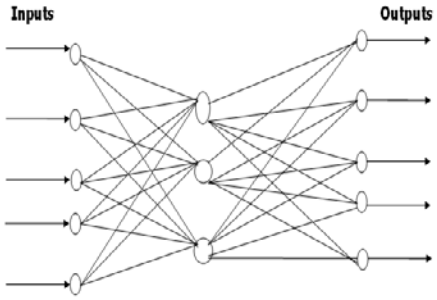
- ➔ Creating Training sets: The pattern recognition block consists of creating the training sets from the images. Here the input & its expected output is known & according to that the Neural network is Trained, which calculates the weight & bias? This creates the Training sets.

Fig 4. Pattern of Hand Gestures

- ➔ Testing: For testing, the unknown image is given as the input to network, which is then compared with the training set. Depending upon the match it gives the output.

5. Performance Analysis

5.1 Feed forward Neural Network



Input Layer Hidden Layer Output Layer

Fig 5. Feed Forward Network

Inputs that are used to create the Training sets are given at the input layer. In the feed forward network the output of the input node is connected to each of the input of hidden layer node. Each input is having weight which is updated according to the error generated by nodes associated with mismatch between the actual output and the desired output. When there is perfect match the error is zero and the weights are fixed. In this way, the network is trained for a particular input and output combination.

$E = \sum(Y-D)2$

Y=Actual Output

D=Desired Output

$Wnew = Wold - \varepsilon (\delta E/\delta W)$

ε =Learning Constant

The images to be tested are given as input to the trained network. Depending on the comparison we are getting output for a given image. The output is depends on training system. i.e. how we train the system.

5.2 Choosing the Neural Network

Now let us see how we can choose the neural network dimension, how many layers should be there and how many neurons should be there in each layer.

a) The input layer

In the input layer, there will be the same number of neurons as the number of elements in each column vector of the "relevant input vectors matrix". It has already been explained that the size of this vector

ISSN 2348 – 7968

depend on the standard variation of each pixel in the whole set, and that if the set gets bigger, there is more chance to add a picture in which a pixel that used to be constant get a new value, because of noise. That's why, when the training set gets larger, the size of this column vector also gets bigger and finally the number of neural network inputs too. It has been observed that a 100 examples-per-sign set is optimal (see next paragraph). According to previous paragraphs, in such case, more than 150 pixels are irrelevant; the minimum number of neurons in the input layer will be 900-150 = 750.

b) Hidden layers:

The problem of classification that has to be solved is simple enough: only five shapes have to be distinguished, and all these examples are easily distinguishable, because these different shapes are all different enough: there is always one finger more. In such cases, only one hidden layer is enough for classification. The issue is now to find the number of neurons in this single layer. For exactly the same reasons, one can guess there will not be hundreds of neurons in this layer. Then, the only way to determine the optimal size of the hidden layer is to compare the performances for different configurations. The following plotting represents the evolution of the rate of successful classification for different number of neurons in the hidden layer:

This plotting has been built by simulating neural networks with successive different hidden layers. Each time, it has been necessary to start the whole training. The value on this plotting corresponds to 1 minus the final rate of false classification.

When the hidden layer contains no neurons, the neural network is completely inexistent. One can think that the rate of correct classification should be 0. In fact, given that the neural network does not works, the final classification output will be the default one, that is to say the first one, which is '1': For any input, the neural network will answer '1'. And given that it has been supposed that the five signs have the same probability, hence the same frequency in the training set, so one time in five, the constant output of the network will be the real expected value. That's why, without any hidden neuron, the rate of successful classification is not 0!

c) Output layer:

The network has to classify between 5 kinds of signs. As a consequence, it will necessarily include 5 output neurons, so that the whole system will provide five output values, which should be considered as a 5x1 vector.

5.3 Advantages of Neural Network

There are a variety of benefits that an analyst realizes from using neural networks in their work.

- Pattern recognition is a powerful technique for harnessing the information in the data and generalizing about it. Neural nets learn to recognize the patterns which exist in the data set.
- The system is developed through learning rather than programming. Programming is much more time consuming for the analyst and requires the analyst to specify the exact behavior of the model. Neural nets teach themselves the patterns in the data freeing the analyst for more interesting work.
- Neural networks are flexible in a changing environment. Rule based systems or programmed systems are limited to the situation for which they were designed--when conditions change, they are no longer valid. Although neural networks may take some time to learn a sudden drastic change, they are excellent at adapting to constantly changing information.
- Neural networks can build informative models where more conventional approaches fail. Because neural networks can handle very complex interactions they can easily model data which is too difficult to model with traditional approaches such as inferential statistics or programming logic.
- Performance of neural networks is at least as good as classical statistical modeling, and better on most problems. The neural networks build models that are more reflective of the structure of the data in significantly less time.
- Neural networks now operate well with modest computer hardware. Although neural networks are computationally intensive, the routines have been optimized to the point that they can now run in reasonable time on personal computers. They do not require supercomputers as they did in the early days of neural network research.

5.4 Defining the different issues

- Collecting the pictures:First of all, and obviously, it will be necessary to collect pictures. There is a choice to do concerning the way we want to collect these pictures, given that it depends on how we implement the main program. Running in the MATLAB environment requires the pictures to be saved in memory and called back when running the program, because the Image Acquisition Toolbox is not available on the MATLAB version used for the design of the program.
- Finding the Hand: Now, let's suppose that a set of representative pictures is provided. We need then to analyze the picture, and to find the relevant part of the

ISSN 2348 – 7968

picture. Indeed the user will never put his hand in the same area of the picture. Here are given few examples of the same sign done in different areas, which have to lead to the same identification result, which should be '2'

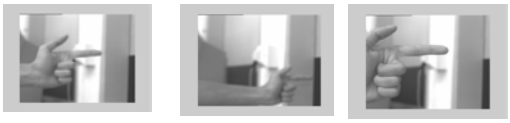


Fig 6. Finding the hand

In any cases, the user expects the program to count three fingers. So it is necessary to find the hand in the picture, that is to say to zoom on the hand, and to pick it up.

➔ Zooming on the Hand: According to the requirements, the video camera is not supposed to move. This piece of information gives a huge advantage that allows simplifying the zooming process. Indeed, it implies that the background is more or less always the same. In all what follows, it will be supposed that in the picture, we can just find the hand and the background: no other object should be present. As a consequence, it will be impossible to exhibit simultaneously the two hands and expect the program to process them both.

5.5 Picture of the difference with the background

The difference with the background can be done using the Matlab function "imabsdiff". After that, to make all the preprocessing easier, it is better to create a binary picture. To do so, it is necessary to choose a threshold: pixels with value lower than this threshold will be set to 0 (black) and others will be set to 1. The choice of this threshold depends on the video camera properties: if we consider that the camera provides pixels coded on bytes, pixel values will be from 0 to 255. Some measurements have proven that in this case, the presence of the hand will imply a variation of pixel values bigger than 20 units. Of course, the optimal threshold depends on the background; nevertheless, this threshold can be correct in most of cases. Then it is necessary to execute noise-removal functions, else every noisy pixel that its value is too high may be considered as part of the hand and will be included in the zoom-in picture. For example if we suppose that the hand is in one corner of the picture and that there is a noisy pixel in the opposite corner of the picture of the differences, so the zooming function will keep it and the resulting picture, after zooming, will not be very different of the initial picture! That's why it is necessary to use noise removal functions.

The noise removal is processed using the function bwmorph (open), that erodes then dilates the noisy picture. By this way, lonely pixels disappear during the erosion; other elements are restored to their initial shape thanks to the following dilation.

Here are given few examples of resulting pictures.

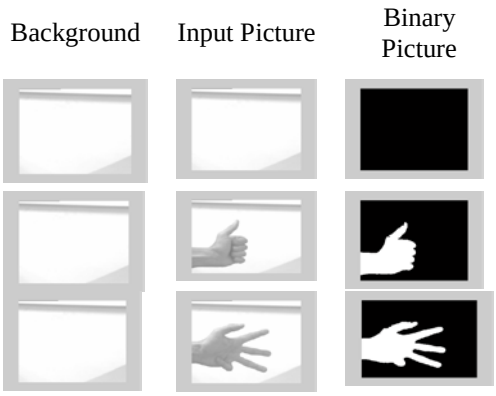


Fig 7. Resulting Pictures

5.6 Zooming the Binary Pictures

Let's suppose that the picture is completely black (0), except in the area of the hand in which it is completely white (1).The problem now is how to crop the area completely black. The fastest method is to compute a line vector in which the i-th element is the sum of the elements of the i-th column of the picture, and a column vector in which the i-th element is the sum of the elements of the i-th line of the picture. For example:

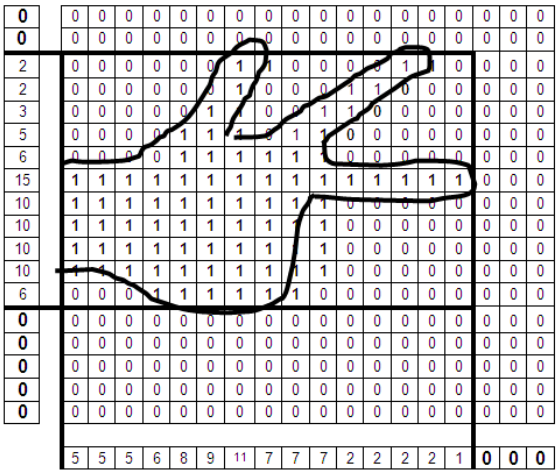


Fig 8. Zoom in on the hand

According to this table, in order to zoom in on the hand, we can just crop the two first lines, the five last lines and the three last columns. Here, the need of noise removal is more understandable: if few noisy pixels appear in the last lines, they may be considered as a part of the hand, and cause the picture not to respect the requirements for the analysis.

ISSN 2348 – 7968

This is an efficient way to find three limits of the hand, but concerning the fourth one, there is a problem. Indeed, we have seen it is quite easy to find the two horizontal limits and the vertical one that refers to the extremity of the fingers. But concerning the side the hand is linked to the arm, how to decide where does start the hand? We have to distinguish the two following schematic situations, but to identify them the same way:

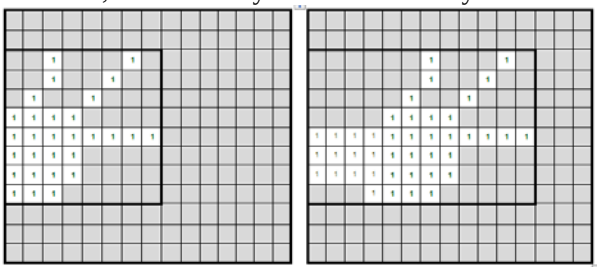


Fig 9. To find the two horizontal limits and the vertical one that refers to the extremity of the fingers

The way to differ this pictures that has been chosen is describe here: The basic idea is to find the thumb and to consider it indicates the left side limit. Using a such processing allows to treat get independent of the position of the hand within the picture:

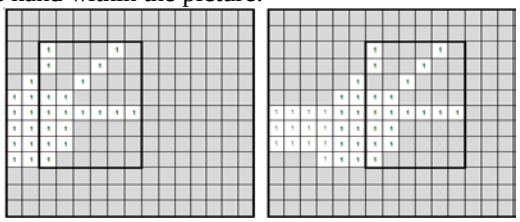


Fig 9. To find the thumb

The issue now is how to find the thumb. The way that has been implemented in the MATLAB files is the following: a new picture that corresponds to the edges of the initial one is created. Then a line vector receives the number of edges in each column of the picture. When we shift horizontally on the picture starting from the left, there is the arm, so the number of edges is 2. When we overpass the thumb, the number of edges suddenly jumps to 4. In order to find the thumb, we just have to find the index of the first column in which there are 4 edges

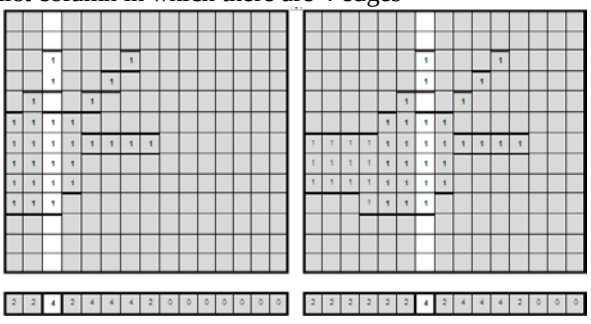


Fig 10. To find the index of the first column in which there are 4 Edges

Finally, it is possible to find the four limits of the square in which the hand is fully included in the initial picture with some very fast preprocessing. The only condition that has been added is that the user has to show his thumb finger, else the recognition cannot be efficient, because the zooming will not be reliable. In this example, the size of the initial picture was 15x15 and the size of the resulting one is 8x6.

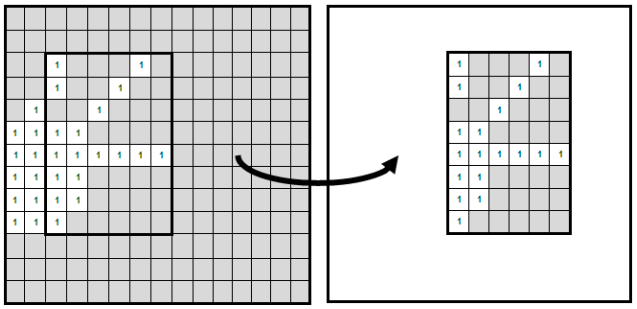


Fig 11. The size of the initial picture was 15x15 and the size of the resulting one is 8x6.

5.7 Standard Resizing

According to the requirements, the user is not supposed to be systematically at the same distance of the video camera. The consequences are obvious: if he is close to it, the hand will occupy a large part of the input picture. At the contrary, when he is far from it, the hand will appear small enough on the picture. So, the pictures of the hands after cropping may have some very different sizes. That's to say that it is necessary to resize all the pictures to a standard size so that we can process them all the same way.

It seems evident that it is not useful to resize it to a size larger than the original one given that it will not add information. Worse, it would be a serious drawback because it would increase the amount of massive calculus, and it is contrary to the constraint of real time processing. For these reasons, it is quite more interesting to reduce the size, but not too much. Indeed, in an excessively reduced picture, some fingers can disappear, and some spaces between two fingers way also disappear so that is seems there is only one finger.

After few tests and measurements, it has been decided that a size of 30x30 is quite small enough to make calculus fast, and large enough to avoid any major damage to the initial picture. In these conditions, the average dimensions of a finger are:

- width: 3~5 pixels
- length: 15~20 pixels

Of course, different users will all have different hands, hence different absolute measurements. Nevertheless, such standard re-sizing will provide relative measurements: if the size of the real thumb and ring fingers

ISSN 2348 – 7968

depend on the user, the ratio will be generally constant. For almost all users:

- $Width(thumb) \approx Width(ring) \approx ... \approx Width(atrial)$
- $\frac{Length(ring)_{User1}}{Length(atrial)_{User1}} \approx \frac{Length(ring)_{User2}}{Length(atrial)_{User2}} \approx ... \approx Cst$

That's why this re-sizing operation can be considered as a standardization process: for any user, the final re-sized image will have almost identical properties concerning the dimension of its elements.

Finally, the fact that the width of a finger is 4 to 5 pixels implies that in the resulting picture

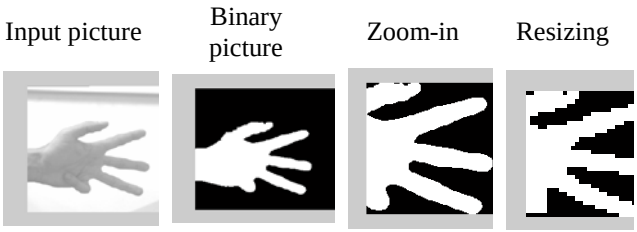


Fig 12. A real example

In these conditions, for any input picture, for any hand gesture that involve the thumb finger, the preprocessing algorithm provides a standard-sized binary picture that corresponds to a zoom on the hand. Once this preprocessing is finished, the real processing can start, that is to say, the identification process can be launched.

5.8 Counting the fingers: Simple Pixel Counting Analysis

The first immediate idea is the following: a picture that contains only the hand of the user is provided to the program. In this picture, if there are only one or two fingers that are exhibited, the numbers of pixels with value '1' will be small. If the five fingers are shown, there will be more pixels at '1'. So, there is a strong link between the number of fingers and the number of pixels set to '1'. The easiest way to classify an image is then to compute the sum of the pixels of the re-sized hand picture, and to compare to the resulting value to different ranges:

If sum < range_1
    Then No fingers
If range_1 < sum < range_2
    Then 1 finger
If range_2 < sum < range_3
    Then 2 fingers
If range_3 < sum < range_4
    Then 3 fingers
If range_4 < sum < range_5
    Then 4 fingers
If range_5 < sum
    Then 5 fingers

The advantage of this method is huge: Such programming is quite easy and very fast. However, it is not a very efficient way:

According to the previous sections, the width of a finger will generally be 4 to 5 pixels, and let's suppose its length is 15 to 20 pixels, according to the user. So let's consider that for User 1, each finger has a dimension:

$$4\,pixels(width)*15\,pixels(length)=60\,pixels/\ finger$$

For User 1, four fingers will lead to about 200 pixels. Let's suppose that for User 2, the width of a finger is 5 and its length is 20. Finger dimension is:

$$5\,pixels(width)*20\,pixels(length)=100\,pixels$$

For User 2, two fingers will also lead to 200 pixels. The Consequence is that the program will get confused and may tell the User 2 he is exhibiting five fingers (two fingers and the thumb) when he just shows three of them (two and the thumb)!

Another issue is that even if it is always the same use who do signs, and that the different ranges have been optimized for his average finger size, errors will probably occur if he doesn't open widely the hand: Indeed, if the hand is fully open, let's assume no error will occur, but if the fingers are a little bit cockled ("closed"), then for each finger, the sum of its relative pixels will be smaller, and if it is the case of several fingers, the global sum may lead to a mistake. An example of this phenomenon is given here:

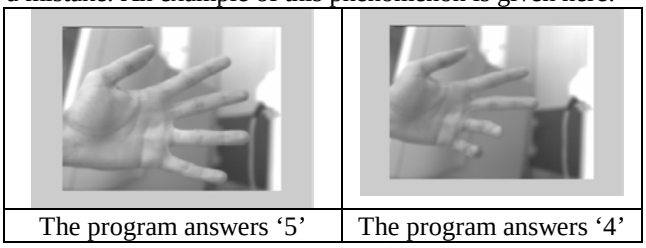


Fig 13. Counting the fingers

6. Conclusion

As a conclusion, it appears that there are plenty of ways to solve such problems. If the neural network method is the most efficient solution when considering the rate of errors, another method based on attributing weights to each column of the input picture, is quite easier to develop and manipulate, and has also been proven very efficient.

There are still some improvements to do, and a major problem to solve: find why does some Matlab files turned into files works perfectly when used in a Graphical User Interface, whereas other Matlab files turned into files using the same way lead to immediate run-time errors.

ISSN 2348 – 7968

Acknowledgments

I take this opportunity to thank my project guide Dr. A.R.Karwankar and Head of the Department Dr. A S. Deshpande for their valuable guidance and for providing all the necessary facilities.

References

[1] Sushmita Mitra, Tinku Acharya, 2007," Gesture Recognition: A Survey", IEEE Transactions on Systems, Man, And Cybernetics—Part C: Applications And views,Vol. 37, No. 3, pp. 311-323

[2] Sebastein Marcel, Oliver Bernier, 2005,"Hand gesture recognition using input-output Hidden Markov models", IEEE Transactions on Pattern recognition & Machine intelligence, Vol. 27, No. 8. pp-347-362.

[3] Yu Yuan, Ying Liu, Keneeth Barner, 2005,"Tactile gesture recognition for people with disabilities",International conference on Accoustic, speech & Signal Processing, Vol. 5, pp. 461-464.

[4] Kenji Oka , Yoichi Sato, Hideki Koike,2002, "Real-Time Fingertip Tracking and Gesture Recognition", IEEE trans. Computer Graphics & Applications, pp 64-71.

[5] Sylvie, C.W., Ong & Surendra Ranganath, 2005, " Automatic Sign Language analysis: A survey & the future beyond Lexical Meaning.", IEEE Transaction on Pattern recognition & Machine intelligence, Vol. 27, No. 6. pp-873-891.

[6] Jose Antonio Monteno, Luis Enrique Sucar, 2006, " A decision theoretic video conference system based on gesture recognition", Proceedings of the 7th Intrnational conference on Automatic face & gesture recognition, Issue 10, pp. 387-392.

[7] Asanterabi Malima, Erol Özgür, and Müjdat Çetin, 2006, " A Fast Algorithm for vision based hand gesture recognition for robot control ", IEEE transaction on signal processing & communication Applications. pp. 1 – 4 .

[8] Akio Ogihara, Hirishi Matsumoto & Akira Shiozaki, 2006, "Hand region extraction by background subtraction with renewable background for hand gesture recognition", IEEE International symposium on Intelligent signal processing & communication systems, pp. 227-230.

Bhushan Bhokse, student ME Electronics, Government college of Engineering, Aurangabad (An Autonomous Institute of Governmant of Maharashtra),Aurangabad,,Maharashtra State,India

Dr. A.R.KArwankar ,Assistant Professor, Department of Electronics & telecommunication Engineering, Government college of Engineering, Aurangabad (An Autonomous Institute of Governmant of Maharashtra),Aurangabad,, Maharashtra State, India