ORIGINAL RESEARCH PAPER

Volume-7 | Issue-8 | August-2018 | PRINT ISSN No 2277 - 8179

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH

HAND GESTURE RECOGNITION SYSTEM USING NEURAL NETWORKS

Engineering

Brajesh Kumar Asst. Prof., Women's Institute of Technology, Darbhanga

Dr. R.K. Singh* Retd. Prof. ,M.I.T. Muzaffarpur *Corresponding Author

ABSTRACT

Using orientation histograms a simple and fast algorithm will be developed to work on a workstation. It will recognize static hand gestures, namely, a subset of American Sign Language (ASL). Previous systems have used data gloves or markers for input in the system. 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 Perceptron network.

KEYWORDS

American sign language, neural computing, perceptron, matlab, image database, recognition.

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.

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 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.

American Sign Language

American Sign Language is the language of choice for most deaf people in the United States. It is part of the "deaf culture" and includes its own system of puns, inside jokes, etc. However, ASL is one of the many sign languages of the world. As an English speaker would have trouble understanding someone speaking Japanese, a speaker of ASL would have trouble understanding the Sign Language of Sweden. ASL also has its own grammar that is different from English. ASL consists of approximately 6000 gestures of common words with finger spelling used to communicate obscure words or proper nouns. Finger spelling uses one hand and 26 gestures to communicate the 26 letters of the alphabet. Some of the signs can be seen in Fig(1) below.

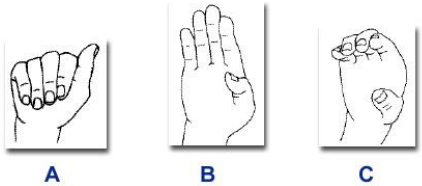

Fig 1: ASL Examples

Another interesting characteristic that will be ignored by this project is the ability that ASL offers to describe a person, place or thing and then point to a place in space to temporarily store for later reference.

ASL uses facial expressions to distinguish between statements, questions and directives. The eyebrows are raised for a question, held normal for a statement, and furrowed for a directive. There has been considerable work and research in facial feature recognition, they will not be used to aid recognition in the task addressed. This would be feasible in a full real-time ASL dictionary.

METHODOLOGY

The procedure can be 'divided' in 6 steps. Let's examine them one by one.

Step1: The first thing for the program to do is to read the image database. A *for* loop is used to read an entire folder of images and store them in MATLAB's memory. The folder is selected by the user from menus. A menu will firstly pop-up asking you whether you want to run the algorithm on test or train sets. Then a second menu wills pop-up for the user to choose which ASL sign he wants to use.

Step2: Resize all the images that were read in Step1 to 150x140 pixels. This size seems the optimal for offering enough detail while keeping the processing time low.

Step3: Next thing to do is to find the edges. As mentioned before 2 filters were used.

Step 4: Dividing the two resulting matrices (images) *dx* and *dy* element by element and then taking the a tan ($\tan^{-1}$). This will give the gradient orientation.

Step 5: Then the MATLAB function *im2col* is called to rearrange the image blocks into columns. This is not a necessary step but it has to be done if we want to display the orientation histogram. *Rose* creates an angle histogram, which is a polar plot showing the distribution of values grouped according to their numeric range. Each group is shown as one bin. Below we can see some examples. While developing the algorithm those histograms are the fastest way of getting a good idea how good the detection is done.

Step 6: Converting the column matrix with the radian values to degrees. This way we can scan the vector for values ranging from 0° to 90°. This is because for real elements of X,

A tan(X) is in the range $[-\pi/2, \pi/2]$

This can also be seen from the orientation histograms where values come up only on the first and last quarter.

Determining the number of the histogram bins was another issue that was solved by experimenting with various values. I have tried with 18 20 24 and 36 bins. What I was looking for was the differentiation (or not) among the images. At the same time I was thinking of the neural network itself as this vector would be the input to the network. The

smaller the vector the faster the processing. Finally, the actual resolution of each bin was set to 10°, which means 19 bins.

The algorithms development was organized having in mind MATLAB weaknesses. The major one is speed. MATLAB is perfect for speeding up the development process but it can be very slow on execution when bad programming practices have been employed.

Nested loops slow down the program considerably. It is probably because MATLAB is built on loops. Therefore, unnecessary back-tracking was avoided and even some routines were written in full instead of using *for* loops.

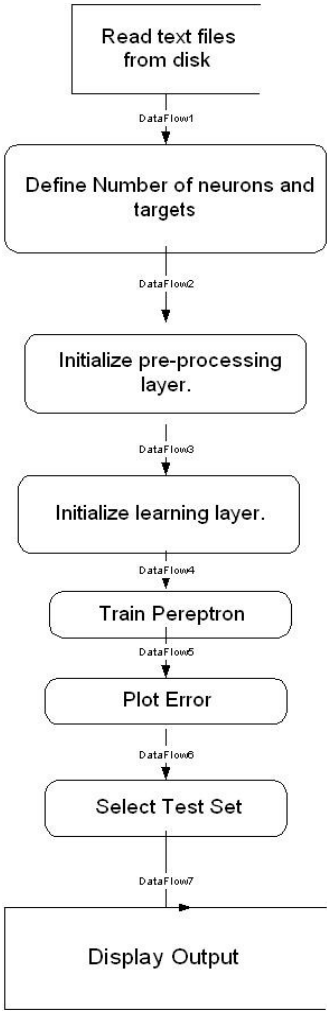

Fig: Flowchart of Pecrceptron

CONCLUSIONS

The idea of the project got started from a *McConnel's* idea of orientation histograms. Many researchers found the idea interesting and tried to use it in various applications. From hand recognition to cat recognition and geographical statistics, my supervisor and I had the idea of trying to use this technique in conjunction with Neural Networks. In other approaches of pattern recognition that orientation histograms have been used different ways of comparing and classifying were employed. Euclidean distance is a straight forward approach to it. It is efficient as long as the data sets are small and not further improvement is expected. Another advantage of using neural networks is that you can draw conclusions from the network output. If a vector is not classified correct we can check its output and work out a solution. As far as the orientation algorithm is concerned it can be further improved. The main problem is how good differentiation one can achieve. This of course is dependent upon the images but it comes down to the algorithm as well. Edge detection techniques are keep changing while line detection can solve some problems. One of the ideas that I had lately is the one of tangents but I don't know if it is feasible and there is not time of developing it.To say that I have come to robust conclusions at the end of the project is not safe. This is possible only for the first part of the project. Regardless of how many times you run the program the output vector will always be the same. This is not the case with the perceptron. Apart from not being 100% stable there are so many parameters (e.g. number of layers, number of nodes) that one can play with that finding the optimal settings is not that straight forward. As mentioned earlier it all comes down to the application. If there is a specific noise target for example you can work to fit these specifications.

REFERENCES:

[1] Christopher M. Bishop, "Neural networks for Pattern Recognition" Oxford, 2015.
[2] William T. Freeman, Michael Roth, "Orientation Histograms for Hand Gesture Recognition" IEEE Intl. Workshp. On Automatic Face and Gesture Recognition, Zurich, June, 2015.
[3] Maria Petrou, Panagiota Bosdogianni, "Image Processing, The Fundamentals", Wiley
[4] Vladimir I. Pavlovic, Rajeev Sharma, Thomas S Huang, "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A review" IEEE Transactions of pattern analysis and machine intelligence, Vol 19, NO 7, July 2017
[5] Srinivas Gutta, Ibraham F. Imam, Harry Wechsler, " Hand gesture Recognition Using Ensembles of Radial Basis Functions (RBF) Networks and Decision Trees" International Journal of Pattern Recognition and Artificial Intelligence, Vol 11 No.6 2016.
[6] Simon Haykin, "Neural Networks, A comprehensive Foundation", Prentice Hall Duane Hanselman, Bruce Littlefield, "Mastering MATLAB, A comprehensive tutorial and reference", Prentice Hall
[7] http://www.cs.rug.nl/~peterkr/FACE/face.html
[8] http://www.hav.com/
[9] http://www.dacs.dtic.mil/techs/neural/neural_ToC.html
[10] http://www.tk.uni-linz.ac.at/~schaber/ogr.html
[11] http://vismod.www.media.mit.edu/vismod/classes/mas622/projects/hands/
[12] http://aiintelligence.com/aii-info/techs/nn.html