April 2012

A Fast Haar Classifier based Gesture Recognition using camShift algorithm and Curve Fitting Method

Bhavish Sushiel Agarwal

PES School of Engineering, Bhavish.agarwal@gmail.com

Jyoti R Desai

PES School of Engineering, Jyothirdesai@pes.edu

Snehanshu Saha

PES School of Engineering, snehanshusaha@pes.edu

Follow this and additional works at: <https://www.interscience.in/ijcsi>

Part of the [Computer Engineering Commons](#), [Information Security Commons](#), and the [Systems and Communications Commons](#)

Recommended Citation

Agarwal, Bhavish Sushiel; Desai, Jyoti R; and Saha, Snehanshu (2012) "A Fast Haar Classifier based Gesture Recognition using camShift algorithm and Curve Fitting Method," *International Journal of Computer Science and Informatics*: Vol. 1 : Iss. 4 , Article 14.

DOI: 10.47893/IJCSI.2012.1056

Available at: <https://www.interscience.in/ijcsi/vol1/iss4/14>

This Article is brought to you for free and open access by the Interscience Journals at Interscience Research Network. It has been accepted for inclusion in International Journal of Computer Science and Informatics by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

A Fast Haar Classifier based Gesture Recognition using camShift algorithm and Curve Fitting Method

Bhavish Sushiel Agarwal, Jyoti R Desai & Snehanshu Saha

PES School of Engineering

E-mail : Bhavish.agarwal@gmail.com, Jyothirdesai@pes.edu, snehanshusaha@pes.edu

Abstract - The use of hand gestures opens a wide range of application for human computer interaction. The paper makes use of haar classifiers and camShift algorithm to track the movement of hand. Parallelism is introduced at every step by segmenting the data from camshaft into an NxN grid. Every block of the grid now represents a lead point which is calculated from mean of all the points belonging to the particular grid. Now we have only N^2 points to recognize the curve that was performed by the user in his action. Finally the fit that was found is compared to pre-defined curve fit data to find out the curve using Mahalanobis equation. Parallelism used in reducing the number of points to be fitted allows the recognition to be faster.

Keywords - Gesture Recognition; Multimedia Application; Hand Tracking; Interpolation; Mean Algorithm; camShift Algorithm; Parallel processing; Video Processing; Point segmentation.

I. INTRODUCTION

Gesture is a movement usually of the body or limbs that expresses or emphasizes an idea, sentiment, or attitude (Merriam-Webster Online). Recognition of human gesture by the way of a machine is a complex scientific problem and throws up a lot of open ended questions. There is ample evidence that gesturing does not simply enhance spoken language, but is part of the language generation process [6]. Research around gesture recognition is geared to build a system able to identify certain human gestures and decode the information conveyed through such an activity. Engineering research also strives to create a system of device control using gesture recognition. In the present paper the authors aim to elucidate a novel method of gesture recognition.

II. RELATED WORKS

[10] is a survey which speaks about the various gesture recognition techniques with an emphasis on hand gestures and facial recognition. [12] is a paper which speaks on gesture recognition using Bezier curves from registered 3 D data. [2] talks on a very novel method of gestures by using quadratic curves to recognize the gesture. [15] talks about dynamic hand gestures via the use of kalman filter based hand tracking and a HMM based temporal characterization scheme.[13] another HMM based algorithm where time-sequential images is transformed into an image

feature vector sequence, and the sequence is converted into a symbol sequence by vector quantization.

III. FRAMEWORK

A. Pre-Processing

The Input Image has a lot of noise so the image has to be pre-filtered so that the hand in the image can be tracked with a much higher accuracy. The input image is converted into HSV space image. This is done according to the paper [2], which allows the proper color recognition without the lighting conditions affecting the overall algorithm. Once the Image is in the HSV space then the image is subject to a HAAR-classifier.

B. Hand Recognition

Before the user starts performing gesture on the screen the hand must be recognized. For this the user must place the hand before the camera and be detected, this allows only the hand to be recognized and no other object, hence noise is eliminated from input. We use the HAAR classifier to perform the hand recognition.

Viola and Jones devised an algorithm, called Haar Classifiers, to rapidly detect any object, including human faces, using AdaBoost classifier cascades that are based on Haar-like features and not pixels [7].

HAAR classifier [3] is used to accurately recognize facial features. Detecting human facial features, face and such as the hands requires the Haar classifier cascades to

be trained first. In order to train the classifiers, these gentle AdaBoost algorithm and Haar feature algorithms must be implemented.

To train the classifiers, two set of images are needed. One set contains an image or scene that does not contain the object i.e. the hand that is to be detected. This set of images is referred to as the negative images. The other set of images, the positive images, i.e. the hand. The location of the objects within the positive images is specified by: image name, the upper left pixel and the height, and width of the object [1]. The negative images could be anything ranging from different people to walls colors and so on. Examples of negative are shown in the below images.

Fig.1 : Hand Recognition using Haar classifier.

Intel developed an open source library devoted to easing the implementation of computer vision related programs called Open Computer Vision Library (OpenCV). The OpenCV library is designed to be used in conjunction with applications that pertain to the field of HCI, robotics, biometrics, image processing, and other areas where visualization is important and includes an implementation of Haar classifier detection and training [4]. Fig.1 show the recognition of hand when the hand is brought in the front of the screen using the Haar classifier.

As shown in the figure, initially the hand is kept still and then the hand is recognized. The blue circle marks the centroid of the frame in which hand is recognized.

C. Hand tracking using CamShift

Once the hand is been identified, all the other objects in the screen (including face) should be eliminated so that the gesture recognized is only of user hand. For the gesture to be recognized we use the camShift algorithm implementation [1]. The algorithm is as follows:

1. The region of interest is acquired from the previous step, where the hand was identified in the frame.

2. The above image is selected as the initial location of the mean shift search. The hand is the target distribution to be tracked.
3. We calculate the color probability distribution of the hand region centered at the mean shift search window.
4. Iterate mean shift algorithm to find the centroid of the probability image. We store the centroid location and the zero moment.
5. Now center the search window at the centroid location that was found and set the window size to the zero moment that was found in the previous step.
6. Save the value of the centroid in a vector.

D. Computing Mean of Points in every grid block

The amount of centroid generated when a gesture is performed by the user can be quite big which can be denoted as M . Fitting all the M points in real time may not be welcome.

Hence as mentioned earlier we can define the grid of size $N \times N$ which will be used to classify the data points that are generated. For example we assume that $N=8$. With this now

Fig.2 : 8X8 grid

we consider the image to be the grid of size 8×8 and every element of the grid to be a block. All the center of the frames from camshaft must belong to one of the 64 blocks that are present in the grid as shown below.

The complete image size is divided into an $N \times N$ grid. In the above image in particular N corresponds to 8. Every block represents one point which will be used to fit the data hence forth, and every block is independent from other block.

Assuming if each point needed to be classified into each one of these blocks in the Fig.2 would take t seconds to complete classifying the points in their blocks, so with 64 blocks it would take about $O(64t)$ to

perform the classification and then finds out the lead points from every block.

A lead point is formed by taking mean of all the points that belong to the particular block. Since calculating lead points in block is independent of all the blocks, we can find the lead points asynchronously. Now we can spawn a threads equivalent to number of blocks and perform our calculation for lead points, which is now of order $O(n)$. Further this can be combined during the calculation of mean by reducing the amount of computation. We made use of CUDATM for its implementation.

CUDATM [6] is a parallel computing platform and programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing the power of the graphics-processing unit (GPU).

We can use CUDATM [6] to use parallelism to calculate the mean of the point, which belong to every block in our given grid.

The mean of the points is calculated as follows:

1. The vector containing the centroid of frames is passed to separate threads for every block in the grid.
2. No of threads spawned is equal to the total number of blocks in the grid.
3. If there is no point belonging to a certain block that block can be ignored i.e. is on the edge of block.
4. The mean of the points in the X and the Y direction for only the points that belong to the certain block in the grid using the following equation.

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i \quad (1)$$

5. Now the above points stored in a feature vector are used in the least-square curve fitting method.

The below mentioned figure shows a straight vertical movement that was performed by the user and the corresponding points are as plotted in the grid. Each point belongs to a certain block of the grid.

Fig. 3: Swipe up gesture

The user performed a swipe up gesture as shown in the above figure. But the line that was performed is not straight, but the variation is allowed in our algorithm.

Fig. 4: lead point applied to every block

After finding the mean of the points along the x and y direction, we get the set of points which need to be fitted as shown in the figure. All the points that lie on the edge of the blocks are discarded. The new curve that is formed is very similar to the existing one, hence preserving the input gesture that was drawn from the user. Above set of points reduce the time taken to fit the data in the curve. In real time analysis this has a considerable impact on the performance of the system.

E. Curve fitting

The size of the vector V_c that is created can be a maximum of 64 in length corresponding to the total number of blocks that are present in the grid. All the 64 blocks in the grid may or may not have the lead points present in them. Hence only the blocks which contain the lead point, are henceforth considered to be useful for the data fit in the above curve.

Least square is a standard approach to calculate the approximate solution for the set of equations in which

they are more equations than unknowns. “Least square” means that the overall solution minimizes the sum of the squares of errors made in every single equation.

General linear least squares is used to fit a set of data points (x_i, y_i) to a model that is not just a linear combination of 1 and x , but rather a linear combination of any order specified function of x . From [2] it is told that an order $n = 6$ is sufficient to fit most of the curves that will be performed by the user.

Now we use the following standard equation of the order 6 (3) to fit our curve.

$$y(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6 \quad (2)$$

To recognize the gestures we pre-compute the fit for the gestures. This can be done theoretically to get the appropriate actions that were performed by the user. Let's call this vector V_p .

Find the curve fit coefficients $(a_0 \dots a_6)$ using the least square for the action that is performed from the lead points vector V_c that was computed in the previous section.

With the above calculated values, compare the coefficients that are found with the normal stored curve values using Mahalanobis algorithm as cited [2]. Mahalanobis distance is given as follows:

$$d(V_c, V_p) = \sqrt{(V_c - V_p)^T S^{-1} (V_c - V_p)} \quad (3)$$

Where S is the covariance matrix which is derived from the vectors above. A first order covariance matrix is defined by [17]

$$S_{ij} = \text{COV}(V_i, V_j) = E[(V_i - \mu_i)(V_j - \mu_j)] \quad (4)$$

Where $V_i = V_c$, $V_j = V_p$

E is the Expected value.

$\mu_i = E(V_i)$, $\mu_j = E(V_j)$

i, j is the length of V_c and V_p

If the calculated value of the distance is small then the user has performed the required gesture action.

F. Algorithm

Following shows the overall algorithm.

Input: A frame from the video or camera as per the users need

Output: The appropriate gesture that will be performed by the user or no action performed if no gesture was found

Step 1. Hand recognition using Haar-classifier

Step 2. If hand is identified in step1 then proceed with step 3

Step 3. Track hand motion using camShift algorithm

Step 4. Store the centroid of frames in V_c that are found in Step 3

Step 5. If sufficient centroid of frames is obtained in V_c then proceed with step 6

Step 6. Using CUDA find out the lead point for every block in the 8x8 grid simultaneously

Step 7. For the lead points, fit the data to the curve using the least square algorithm (2)

Step 8. For the curve that was fitted use the Mahalanobis (3) to find the distance.

Step 9. If the distance is small from the required curves, then we have found the required curve.

In step 6 of the above algorithm if all the *lead* points of $N \times N$ grid is found sequentially then it would take $T(N)$. Taking advantage of CUDATM we take the 10 series GPU's [16] which provides 8 cores of processor to process the data. Now we have $T_p(N)$ which solves step6 in parallel, where P is the number of cores of processor.

Now the speed up $S_p(N)$ is given by

$$S_p(N) = T(N) / T_p(N) \quad (5)$$

$$S_p(N) \leq P \quad (6)$$

So P times the algorithm is speed up at that step.

The efficiency of the algorithm at step 6 also improves as

$$E_p(N) = T_1(N) / (P T_p(N)) \quad (7)$$

Where $T_1(N)$ is the time taken with one processor in step 6.

Hence the efficiency is improved by P times with parallelism, which is significant when real time implementation is concerned.

IV. RESULTS

Different gestures performed by the user that were recognized by the system using the above algorithm are as shown in the figure below.

Fig.5

The gesture performed can be used to perform any task further. The following table shows the accuracy with which the gestures were recognized by the system.

Table 1. Recognition accuracy of different gestures

Gesture Performed	Recognition accuracy
Curve right	89.3%
Curve left	90.1%
Swipe down	93.2%
Swipe left	95.5%

The accuracy of the system is good and the recognition rates are fast. When the accuracy rates need to be higher than the current needs, the amount of blocks i.e. the value of N in the grid can be increased which will allow the error in curve fitting to reduce. Hence increasing the accuracy but at the cost of time, since the time that will be taken to fit the curve will also increase as more points are fitted in the curve. Different types of curves apart from the once that are fitted above can be recognized with great speeds and accuracy.

V. CONCLUSION

Parallelism was introduced while reducing the number of points of the curve to be fit which improves the overall efficiency of the algorithm. The numbers of points were reduced by calculating a *lead* point by taking mean of all the points that belong to the particular block. Also the number of errors during the fit is reduced. This enables recognition of gesture on the

mobile which provide limited computing power to be faster. Pre-processing of the data is essential when gesture needs to be recognized in real-time. The above stated algorithm combines all the popular techniques.

REFERENCES

- [1] John G. Allen, Richard Y. D. Xu, Jesse S. Jin , Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces, Pages 3-7, IP '05 Proceedings of the Pan-Sydney area workshop on Visual information processing
- [2] Qiulei Dong, Yihong Wu, and Zhanyi Hu, Gesture Recognition Using Quadratic Curves. ACCV (1) 2006: 817-825
- [3] Phillip Ian Wilson , Dr. John Fernandez , Facial feature detection using HAAR classifiers, pages 127-133 , Journal of computing sciences in colleges Volume 21 Issue 4 , April 2006
- [4] Open Computer Vision Library Reference Manual. Intel Corporation, USA, 2001.
- [5] www.itl.nist.gov/div898/handbook/pmd/section1/pmd141.htm
- [6] D. McNeill and E. Levy. Conceptual Representations in Language Activity and Gesture, pages 271-295. John Wiley and Sons Ltd, 1982.
- [7] Viola, P. and Jones, M. Rapid object detection using boosted cascade of simple features. IEEE Conference on Computer Vision and Pattern Recognition, 2001.
- [8] Yaav Freund, Robert E. Schapire. "A decision-Theoretic Generalization of on-Lianer and an Application to boosting"
- [9] An algorithm for restricted least square regression , RL Dykstra - Journal of the American Statistical Association, 1983 – JSTOR
- [10] Gesture Recognition: A Survey ,SushmitaMitra, Senior Member, IEEE, and TinkuAcharya, Senior Member, IEEE,IEEE transactions on System , Man and Cybernetics.
- [11] Recognition System, MokhtarM.Hasan, Pramod K. Mishra, HSV Brightness Factor Matching for Gesture ,International Journal of Image Processing (IJIP), pages 457-467,Volume (4): Issue (5)
- [12] Min C. Shin, Leonid V. Tsap, Dmitry B. Goldgof, Gesture recognition using Bezier curves for visualizationnavigation from registered 3-D data, Pattern Recognition 37 (2004) 1011 – 1024
- [13] Yamato, J., Ohya, J., Ishii, K.: Recognizing human action in time-sequential images using hidden markov model. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition, champaign, IL (1992) 379-385
- [14] Kjeldsen, R., Kender, J.: Finding skin in color images. In: Proc. IEEE Workshop on Automatic Face and Gesture recognition, Killington, Vermont, USA (1996) 312-317
- [15] Aditya Ramamoorthy, Namrata Vaswani, Santanu Chaudhury, Subhashis Banerjee, Recognition of dynamic hand gestures, Pattern Recognition 36 (2003) 2069 – 2081
- [16] <http://developer.nvidia.com/>
- [17] <http://mathworld.wolfram.com/CovarianceMatrix.html>