MM803 Final Project Report

Real-Time Hand Gesture Detection Using Haar-Like Features

Xinyao(Alvin) Sun

***Abstract****:*

*In this project,* *I describe a vision-based hand gesture classification. The low level of the implementation is using the Haar-like features and the Adaptive Boosting (AdaBoost) machine learning algorithm for a certain hand gesture recognition. This algorithm achieves the real-time performance and high recognition accuracy and has been approved on a published paper [1]. Moreover, based on this technique, I present high-level improvements as an extension work to increase the classifying accuracy. During the real-time tracking, the system pre-processes the current frame image to extract the skin area with a skin model. The next stage is using the classifier to predict the bounding boxes of the hand. Then the post-processing through the Kanade-Lucas-Tomasi feature tracker to track the features points within the detected bounding boxes. The demonstration of this* *improved system, I tested a short video with a pre-trained model for “fist” hand gesture the overall accuracy obtained more than 95% that is significantly better than the original system. Moreover, the system performance is validated in real video sequences.*

1. **Introduction**

Hand gestures are a powerful human to human communication modality that can be used to Human Computer Interface (HCI) applications[1]. Using hand gesture to interact with computer interface is an effective way to instead original touch-screen interfaces [2]. For example, in some clinical condition, such as ultrasound test that often uses fluids, where medical personnel have to take off the gloves to interact the physical system [3]. Therefore, vision-based hand tracking has become an important problem in the field of HCI that meets requirements for real-time, accuracy and robustness. For vison-based hand recognition techniques, there are two main categories: appearance-based and 3D hand model-based approaches [4].

Due to the topic of this course is image processing, my project focus on the domain of appearance-bases approach. Where using image features to model a visual appearance of the hand pose and compare these parameters with the extracted images’ feature from the input video. In this context, I propose a real-time hand gesture recognition system based on the existing proved booster classifier algorithm that allows the robust and fast detection of the hands [1]. Through combining the using of skin segmentation and feature points tracking procedures to increase the system’s overall accuracy. This report is organized as follows: In Section 2, I introduce some previous related work. In section 3 the algorithm base on the Adaboost algorithm is presented. In section 4 presents the high-level improvements of the tracking system for this course project. Moreover, the details of implementation and demonstration results for a real-time video sequence are presented and analyze in Section 5. The conclusion and discussion are given in Section 6.

1. **Related work**

Some traditional vision-based techniques that used to track hands based on color segmentation [5] that need users to wear colored gloves or contour-based methods [6] that only work on restricted backgrounds or infrared segmentation based methods [7] that require expensive infrared cameras. Therefore, recently shape-based and skin-based methods are the two primary research direction.

1. **Cascade of Boosted Classifiers**

Viola & Jones introduces a cascade detector built from Harr features [8] achieve a robust detection of face detection. They employed a statistical approach to handling the large variety of human faces. In [1] author uses the same methodology that combining Haar-Like Features and the AdaBoost algorithm to detect a certain hand gesture. The overall framework of Cascade Classified based on Adaboost algorithm as shown in Figure 1.

Figure : Cascade of AdaBoost classifiers framework

Using Integral image to increase the computation effeciecy

Haar-Like features

Construct a strong classifier

Construct multiple weak classifiers

Cascade Classifier

AdaBoost

*3.1 Haar Features*

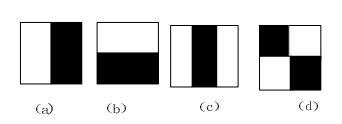
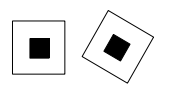
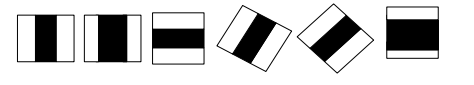
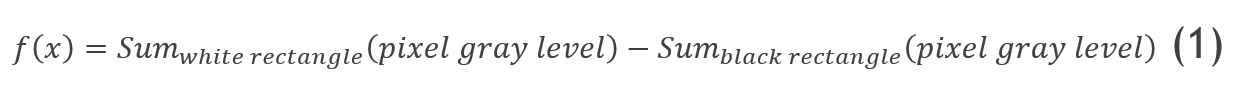
To present the feature of a certain hand gesture, using a set of Haar-like feature that focus more on the information within a certain area of the image rather than each raw single pixel. The simple Haar-like features as shown in Figure 2 which are used in [8] where (a) and (b) is edge features and (c) is line features and (d) is opposite. (so-called Harr-like because it computed similarly to the coefficients in the Haar wavelet transform). There is also some extended Haar-like features set proposed by Lienhart and Maydt [9] as shown in Figure 3 & Figure 4.

Figure 2:A set of basic Haar0lie features

Figure 3: Center-surround features

Figure 4: Line features

 As shown in above figures, each Haar-lie features contains two or three connected “black” and “white” rectangles. The value of one certain Haar-like feature is the difference between the sums of the pixel values (grayscale value in our case) within the black and white rectangles Equation 1. Hence, at the same position of the hand and non-hand images, the feature value is different. The motivation for using the Haar-like features rather than raw pixel values is: Haar-like features can describe ad-hoc domain knowledge that is hard to represent by a finite quantity of training data. On the other hand, the Haar-like features efficiently reduce/increase the in-class/out-of-class variability [9], hence it makes the classification easier. Haar-like features describe the difference between the “black” and “white” areas within a kernel. Therefore, these features are also relatively robust to noise and lighting changes because they compute the gray level difference between two areas within a kernel. The noise and lighting variations affect the pixel values within the whole feature area. This kind of influence can be counteracted during Haar-like feature calculation.

*3.2 Integral Image*

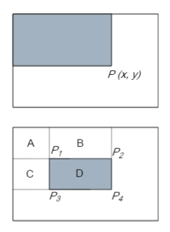
The general base subwindow used for Haar-like kernels is 24x24 in any given image. Considering all possible parameters of the Haar features like position, scale, and type within a 24x24 base window, the end up calculating about more than 160,000 features. Since it is clear that a huge number of these rectangular Haar-like features have to be evaluated each time. Therefore, the integral image is a neat technique to reduce the computation rather than summing up all pixel values under the black and white rectangles every time. The concept of integral image is to find the sum of all pixels under a rectangle with just four corner values of the integral image. The pixel value at the location of x,y in original image *i* contains the sum of the pixel values (grayscale value in our case) of the left of x,y Equation 2. As shown in Figure 5, The Sum of pixel value within “D” is - (+. Therefore, whatever the changing of the kernel, the calculation costs a constant time. This makes the rectangle Haar-like features can be computed rapidly.

Figure 5: Integral Image

*3.3 Adaboost Algorithm*

*3.3.1 Weak Classifier*

As mentioned before, using the Haar-like features to detect the hand gesture within an image, the image is scanned by a sub-window contains a certain Haar-like feature. As shown in Figure 6. Based on each Haar-life feature a weak classifier can be defined as: Equation 3.

Figure 6: Detecting a fist gesture with a Haar-like feature

is a subwindow, and is the direciton of the inequality sign.  is the treshold. For each feature, we need to train a weak classifer which generates the lowest classfication error on training data with optimal . The trained process is

1. For each feature *f*, calculate all feature value for all training data and sort the feature values.
2. For each sorted elements, calculate all positive weight (with hand) and all negative weight (without hand) . Then calucate all posive weight before this elemtne and negative weight before this element
3. The final error for each element is
4. The threshold is an element with minimal e.

*3.3.2 Strong Classifier*

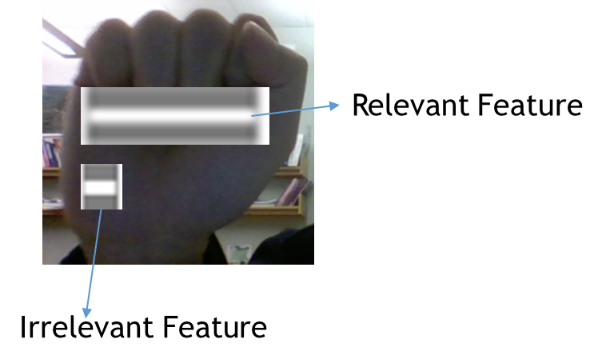
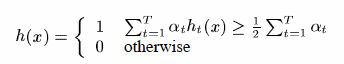
As stated above there can be approximately more than 160,000 features when a detector at 24x24 based resolution that need to be calculated. However, it is to be understood that only a few set of features will be useful among all these features to identify a hand gesture. As shown in Figure 7 one of the Haar-like features is relevant, and the other is irrelevant. Then Adaboost is a machine learning algorithm that helps in finding only the best features among all these 160,000 features. After these features are found a weighted combination of all these features in used in evaluating and deciding any given window contains a certain hand gesture or not. Each of the selected features is trained as a weak classifier in 3.3.1. To obtain a strong classifier we need preset the number of iteration T then:

Figure 7: Relevant and irrelevant Haar-like feature

1. Given the sample positive samples X and negative samples Y.
2. Initialize each sample’s weight as 1/(2X) for positive samples and +1/(2Y) for negative samples.
3. Using X+Y sample to train the first best weak classifier as in 3.3.1 which has the lowest classification error. Then re-weight the samples that have been misclassified.
4. Then using the new samples and misclassified samples to train the next term
5. Loop T times to get T best weak classifiers.

Then final strong classifier is defined as Equation (4), where is the classification error of t’s weak classifier.

*3.4 Cascading*

In a practical implementation, using one strong classifier that based on the AdBoost algorithm, due to the detection requests scan every location of the original image. Moreover for the detection, we need to apply multi-scales and multi-areas detection. Multi-areas means we need to cut the original image with multiple areas by sub-window. There are two main methods for multi-scale detection: change the size of original image, however because the feature value is calculated based on the integral image if the size of the original image is changed. The integral image has to be regenerated, and it costs extra computations. The second method is feature scaled that keep the size of the original image but keeps changing the size of the sub-window. Therefore, the second way is better than the first one. However, both of these methods causes an excessively large amount of the evaluated sub-windows. Moreover, most of the sub-windows would be the negative that without any certain hand gesture. Furthermore, according to [8], the AdaBoost strong classifier has a low false positive rate at the same time has a small detection rate. Using one strong classifier, increasing the detection and decreasing the false positive rate is conflictive.

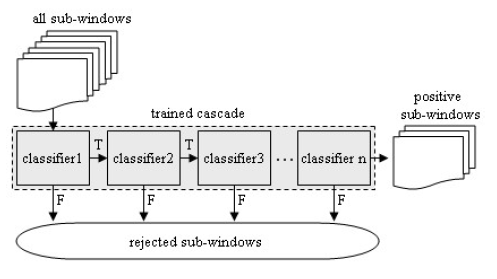
Due to these situations, the attention cascade is employed to increase the number of the strong classifier that has been approved in [8] to deal with the above problems. To train a Cascade classifier: the first stage the threshold of the AdBoost classifier is adjusted low enough to almost 100% of the objects can be detected while keeping the false negative rate close to zero [8]. Then it would have high false positive detection rate. Then a positive result from the first classifier triggers the evaluation of the a second stage classifier, which has also been adjusting to achieve very high detection rates. A positive result from the second stage classifier triggers a third stage classifier and so on. The detection of the trained cascade, all the subwindows must pass each stage of the cascade, if the subwindow leads rejection by any point of the stage, it will be treated as a negative outcome as shown in Figure 8. The advantage of using the cascade classifier is keeping the small true positive rate and still achieve an acceptable detection rate. Furthermore, the detector concentrates on discarding non-hand subwindows quickly and spend more on time on probable hand regions to speed up the performance.

Figure 8: Detection of positive sub-windows using the cascade

1. **High Level Improvements**

The performance of the statistical classifier depends strongly on how representative the training sets are. The harr-like feature has the limitation on if there are some objects has a similar Haar-like feature on the background, this causes false positive detection as shown in Figure 9. Moreover, the classifier based on the Haar-like feature is sensitive to the In-plane and out-of-plane orientation of the object, if the object has the orientation, the detector would be unable to detect the hand as shown in Figure 10.

Figure 10



Figure 9

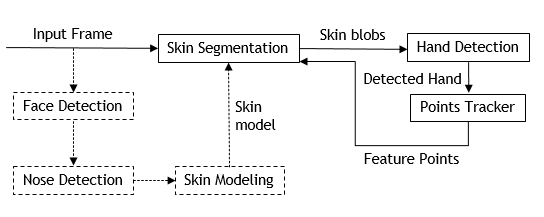
In order to cover these limitations, I present the pre-processing and the post-processing procedures between the detection part to build a more robust detection system for real-time video. The overall framework of the system is shown in Figure 11.

Figure 11: Proposed hand gesture detection system

*4.1 Pre-processing with Adaptive Skin Segmentation*

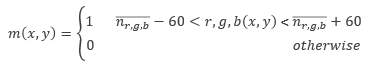
The adaptive skin segmentation is implemented based on the similar procedure in [10]. The main idea is to use the color information of face to build a specific skin model. The Cascade classifier for face and nose detection is robust enough and embed in Matlab. Moreover, the trained model for face and nose is also public on the website in XML format that can be directly used by Matlab and OpenCV. Get the advance of these existing works, I introduce a workflow that first using Face cascade classifier to detect the region of the face, and within the face region to detect the nose region through the Nose cascade classifier. The reason chose the nose region because the nose provides a more accurate measure of the skin tone, it does not contain any background pixels as shown in Figure 12. The proposed skin model is defined as follows:

Figure 12: Detect the region of nose

Where the binary mask *m* at *(x,y)*, 1 indicates the pixel is the skin, and 0 indicate the non-skin pixels. The model is defined by the mean value of Red, Green, Blue Channel of the region of nose as: , , . To avoid some noise cases, I apply a median filter with size 7x7, to remove the noise pixels within the binary mask to keep the integrity of the skin and none-skin areas as shown in Figure 13. The sample output after pre-processing of skin segmentation are shown in Figure 14. This procedure removes lots of complex objects from the background to reduce the false positive rate of the cascade classifier further.

Figure 13: Apply the mask to input image, the left using the un-filtered mask and right one using filtered mask

Figure 14: Sample output after skin segmentation

4.2 Post-processing with Detected Hand Features Tracking

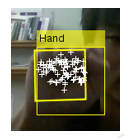
After one hand has been detected, I applied a hand-tracking module that is built using the Kanade-Lucas-Tomasi (KLT) [11] feature tracker to track the hands feature points from the detected hand region. After applying the minimum eigenvalue algorithm [13], I got a set of feature points locates within the hand region as shown in figure 15. Once the feature points have been identified in the current frame within bounding box of hand , for the next frame, the points tracker based on the KLT algorithm would attempt to find visibale points of these corresponding points in the next frame to get . Then using the M-estimator Sample Consensus (MSAC) algorithm [12] to estimate the transiation, rotation and scale between the and . After get the estimated transformation matrix, I apply to the original boundbox of the dected hand to get . Because the cascade filter still detects region(s) of hand in the current frame I extract all feature points from this/these region(s) to get . During the next step, the sytem filter the feature points to extract the points within in the box to get . Then unions the to get updated of points tracker for next frame processing. As shwon in Figure 16, the new detected region of hand updates the feature points of the origianl boundingbox.

Figure 16

Figure 15: KLT feature points

The advantage for this post-processing procedure is 1) Tracking the detected hands all the time to avoid the miss-classified condition caused by the orientation of the hand that is hard to be detected by my cascade classifier. 2) The KTL feature points tracking algorithm is sensitive to some slightly pixel value changes of the hand and rapidly large movements, because it lost most of feature points. Therefore updating on the feature points based on the detection results of the cascade classifier on each frame ameliorates this situation.

1. **Evaluation**

*5.1 Implementation*

*5.1.1 Train the Cascade classifier*

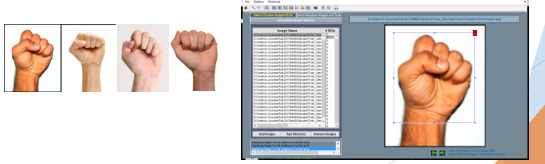
In my implementation, because the cascade training is very time-consuming and the accuracy of the classifier is very depends on how well the quality and quantity of training data. Therefore, I choose one public pre-trained model for “fist” hand gesture. The procedure for training this classifier is collecting the positive samples from a hand picture database with “fist” hand pose. Then using the tool to select the bounding box of the hand region manually as shown in Figure 17.

Figure 17: The sample positive samples for ‘fist’ gesture, and the tool for selecting region of the interest.

For negative samples, which are arbitrary images mush not contain the “fist” hand pose as shown in Figure 18.

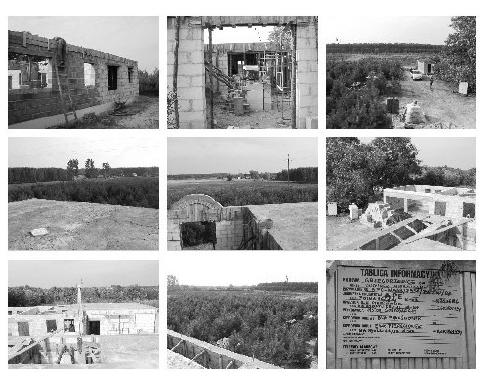


Figure 18: Sample of negative images

The pre-trained model I used for “fist” gesture is a 16-stage cascade classified what a true positive detection rate at 98%.

*5.1.2 Overall system*

The overall system is developed in MatLab, The camera used for the video input is the built-in USB2.0 UVC Camera with 640x480 resolution. Using GPU array to accelerate the processing for the median filter during pre-processing for skin segmentation part. Nose and face detection are using the built-in Harr-like cascade model in MatLab.

*5.2 Results and Discussion*

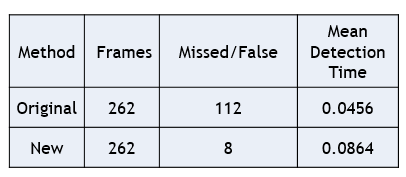
 In order to assess how well my the high-level improvements affect the performance of the whole detection system. I recorded one video and test with the improved system and original system. The results are shown in Table 1. For objective evaluation, I manually go through each output frame by frame, there are totally 262 frames for tested video. “Missed/False” section is the number of frames that the system miss detects the hand or generate false positive results. Detection time is the mean processing time for one frame of the system. Therefore, for this demonstration video, the accuracy of the original system is 1-(112)/262 = 53.82%, the system with high-level improvements is 96.95%. Almost frames are detected correctly under the improved system. The detection speed on my laptop is 10 fps for the improved system and 20 fps for original one. It is still reasonable for real time condition, and it is supposed to be faster if the program is implemented using C++ with OpenCV. The screenshot from the improved system as shown in Figure 19 indicates the system perform better detection on orientation cases. Moreover, Figure 20 shows the most of background pixels has been removed through the skin segmentation procedure.

Figure 19

Table 1: Sample results

Figure 20

1. **Conclusion and Future Work**

In this project, I proposed a Cascade of Haar-like feature based AdBoost classifier for a certain gesture “fist” detection in real time video. To increase the accuracy of the detector system, I present the pre-processing with skin segmentation and post-processing on extracted feature points tracking of the detected hand. Based on the demonstration results, the improved system achieves more than 95% accuracy on detection accuracy for the test video with significantly better performance rather than original work. In future work, because currently the system is just tested one hand gesture, furthermore I can apply more gestures with multiple cascade classifiers and applying the detection in a parallel structure to make the system able to detect multiple hand gestures in real time. Moreover, the detection speed for one frame just around ten frames per second under 630x480 resolution currently. It is still not efficiency enough. I plan to implement the current system using C++ with OpenCV to figure out how well the speed can be enhanced.

**References**

1. Fernandes B, Fernández J. Using haar-like feature classifiers for hand tracking in tabletop augmented reality[C]//XII Symposium on Virtual and Augmented Reality. 2010: 6-13.
2. Rossol, N.; Cheng, I.; Basu, A., "A Multisensor Technique for Gesture Recognition Through Intelligent Skeletal Pose Analysis," in *Human-Machine Systems, IEEE Transactions on* , vol.PP, no.99, pp.1-10 doi: 10.1109/THMS.2015.2467212
3. Carly E Glarner, Yue-Yung Hu, Chia-Hsiung Chen, Robert G Radwin, Qianqian Zhao, MarkWCraven, Douglas A Wiegmann, Carla M Pugh, Matthew J Carty, and Caprice C Greenberg. Quantifying technical skills during open operations using video-based motion analysis.Surgery, 156(3):729{734, 2014.
4. H. Zhou, T. S. Huang, “Tracking articulated hand motion with Eigen dynamics analysis,” Proc. of International Conference on Computer Vision, Vol. 2, pp. 1102-1109, 2003.
5. R. Y. Wang and J. Popovic, "Real-time hand-tracking with a color glove," in ACM SIGGRAPH 2009 papers New Orleans, Louisiana: ACM, 2009.
6. V. Gemignani, M. Paterni, A. Benassi, and M. Demi, "Real time contour tracking with a new edge detector," Real-Time Imaging, vol. 10, pp. 103-116, 2004.
7. Y. Sato, Y. Kobayashi, and H. Koike, "Fast tracking of hands and fingertips in infrared images for augmented desk interface," in Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition., 2000, pp. 462-467.
8. Viola P, Jones M. Rapid object detection using a boosted cascade of simple features[C]//Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. IEEE, 2001, 1: I-511-I-518 vol. 1.
9. R. Lienhart, J. Maydt, “An extended set of Haar-like features for rapid object detection,” Proc. IEEE International Conference on Image Processing ICIP 2002, Vol. 1, pp. 900-903, 2002.
10. Wimmer, M., Radig, B.: Adaptive Skin Color Classificator, Int. Journal on Graphics, Vision and Image Processing. Special Issue on Biometrics 2, 39–42 (2006)
11. Birchfield S. KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker[J]. 2007.
12. P. H. S. Torr and A. Zisserman. Mlesac: a new robust estimator with application to estimating image geometry. Comput. Vis. Image Underst., 78(1):138-156, 2000.
13. Shi, J., and C. Tomasi, "Good Features to Track," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June 1994, pp. 593–600.