ROBUST HAND DETECTION IN LOW-RESOLUTION IMAGES

A Thesis Presented

by

Nyan Bo Bo

Master of Science

Information Technology Program

Sirindhorn International Institute of Technology

Thammasat University

October 2007

ROBUST HAND DETECTION IN LOW-RESOLUTION IMAGES

A Thesis Presented

By

Nyan Bo Bo

Submitted to

Sirindhorn International Institute of Technology

Thammasat University

In partial fulfillment of the requirement for the degree of

MASTER OF SCIENCE IN TECHNOLOGY

Approved as to style and content by the Thesis Committee:

Chair and Advisor	Asst. Prof. Bunyarit Uyyanonvara, Ph.D.
Co-Advisor	Asst. Prof. Matthew N. Dailey, Ph.D.
Member	Assoc. Prof. Stanislav S. Makhanov, Ph.D.
Member	Asst. Prof. Toshiaki Kondo, Ph.D.

October 2007

Acknowledgment

Firstly, I would like to give my greatest gratitude to my former advisor Asst. Prof. Dr. Matthew N. Dailey and current advisor Asst. Prof. Dr. Bunyarit Uyyanonvara for their invaluable contribution to my research and thesis. I have successfully completed my research, this thesis and all requirements for my masters degree because of their guidance, support and encouragement. They are my source of knowledge and wisdom.

I also would like to express my appreciation to my thesis committee members Assoc. Prof. Dr. Stanislav S. Makhanov and Dr. Toshiaki Kondo for their helpful support, suggestions and comments on my research. Their suggestions and recommendations helped me to have good progress on my thesis.

I am very grateful to Assoc. Prof. Dr. Thanaruk Theeramunkong, head of the School of Information and Computer Technology (ICT), for his kind advice, the knowledge and experiences he shared with. Throughout my study, the staff at SIIT, especially the secretaries of the School of ICT, have helped me with all official paperwork. So, I would like to appreciate their contributions here.

Then, I also would like to thank Sirindhorn International Institute of Technology (SIIT) for providing me with a teaching assistant scholarship, not only waiving all of my tuition fees but also giving me a chance to earn my living expenses.

During my study here in SIIT, my colleagues have created a pleasant atmosphere for me with their friendly company, helped me to pass over hardship with my study, and shared experiences they had in doing research. Therefore, I would like to thanks my colleagues for everything they have done for me. Here, special thanks is given to Ms. Akara Sopharak, Mr. Sakib Jalil and Mr. Yoichi Nakaguro for their great help during data collecting.

Last, but not least, I would like to give enormous thanks to my father, mother and all family members for supporting me with their love and care.

Abstract

Robust real-time hand detection and tracking in video sequences would enable many applications in areas as diverse as human-computer interaction, robotics, security and surveillance, and sign language-based systems. The first contribution in this thesis is the introduction of a new approach for detecting human hands that works on single, cluttered, low-resolution images. My prototype system, which is primarily intended for security applications in which the images are noisy and low-resolution, is able to detect hands as small as 24$\times$24 pixels in cluttered scenes. The system uses grayscale appearance information to classify image sub-windows as either containing or not containing a human hand very rapidly at the cost of a high false positive rate. To improve on the false positive rate of the main classifier without affecting its detection rate, I introduce a post-processor system that utilizes the geometric properties of skin color blobs. The four features, which help discriminating false positives from real hand, are second contribution of this thesis. When my detector is tested on a test image set containing 106 hands, 92 of those hands are detected (86.8% detection rate), with an average false positive rate of 1.19 false positive detections per image. The rapid detection speed, the high detection rate of 86.8%, and the low false positive rate together ensure that my proposed system is usable as the main detector in a diverse variety of applications requiring robust hand detection and tracking in low-resolution, cluttered scenes.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

If it were possible to detect and track human hands in video sequences, a variety of useful applications would be possible. These applications include human-computer interaction, human-robot interaction, gesture and sign-language recognition, intelligent security systems and more.

Over the last 15 years, the problem of hand tracking has become an attractive area for research in the field of computer vision. Many early hand tracking systems relied on uncluttered static backgrounds, high resolution imagery, and manual initialization. Most of the modern hand tracking systems are oriented towards sign language recognition, human-computer interaction, and human-robot interaction. In these applications, it is possible to make the very useful assumption that only hands are moving while the rest of the scene is stationary. The problem can be further simplified by assuming that there will be only two hands, since there should be only one person performing sign language or gestures in the scene. Nowadays, the systems are becoming more robust, but they generally still require high resolution imagery.

1.1 Problem Statement

We are primarily interested in hand detection and tracking because monitoring peoples' hands could be a key to predict what that person is doing. In security applications, it would be very useful to detect and track hands of people in the scene and perform automated analysis of their actions, e.g., by determining if they are walking, running, punching someone, or even identifying any object they are holding. Detecting and tracking hands in security applications is more challenging than in other applications such as human-computer interaction because most surveillance cameras provide noisy images, with human figures quite far away and therefore appearing at a fairly low resolution. The resolution of hands in those images may be as small as 24×24 pixels or even smaller; detecting such small hands in static images is a very challenging task. Another difficulty is that motion information is possible in the security application which aim is to locate human hands and identify the object in the hand from single image. The problem of detecting hands in single image becomes more difficult if the hands in the image are in low resolution and noisy.

To track hands in video sequences, we must first *detect* hands, then use some tracking algorithm to link detected hands from frame to frame. The majority of the hand trackers rely mostly on relatively simple detection algorithms to initialize the tracker, then fairly powerful tracking algorithms maintain an estimate of the state of that hand. For example, some systems simply use a skin detector to detect skin blobs. Systems using such simple detection algorithms will be less robust when video sequences are cluttered with many potential hand blobs. A system that is able to robustly detect hands in static images will be a major contribution to the development of robust hand tracking systems. The hand bounded by the square in Figure 1.1 is indeed the size of 24×24 pixels and is corrupted by noise. When the bounded region is cropped out and zoomed as show in lower right corner of Figure 1.1, it is difficult even for humans to recognize what it is. The ultimate goal of my research is to construct a system which is able to detect multiple small hands like this in cluttered noisy single images.

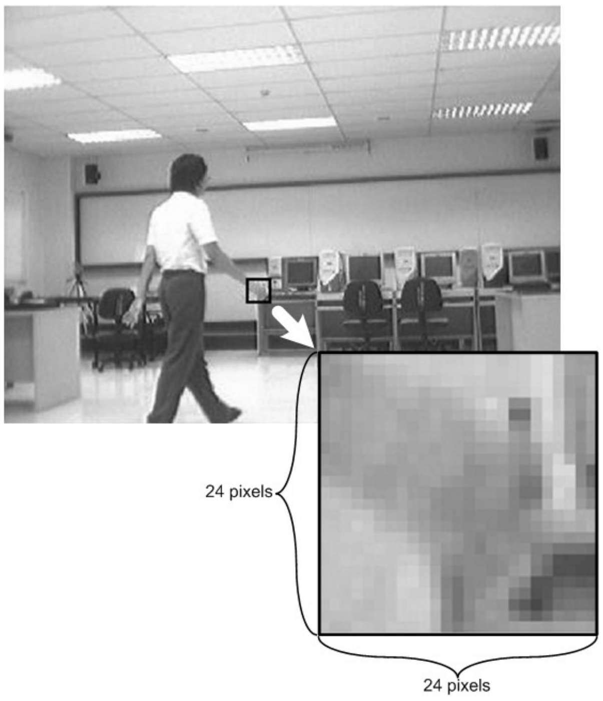


Figure 1.1: Low resolution and relatively noisy image (640×488) captured inexpensive IEEE1394 web cam. By looking at zoomed hand, it is obvious that detecting hands in this image is very challenging.

1.2 Approach In This Thesis

Recently, several face detectors [1, 2, 3] have been introduced and some face detectors are now in commercial applications such as focusing on faces in digital cameras. Among those face detectors, I hypothesized that the Viola and Jones robust real-time face detection cascade [3] would be useful for detecting hands in static images. But detecting low-resolution hands is much more challenging than detecting faces since faces have are well structured, with mouth, eyes, eyebrows, and nose in predictable positions. This structural information is available even on faces in low-resolution noisy images. For low-resolution hands, in contrast to faces, there is little useful structural information except the contour of hand, as shown in Figure 1.1. I found that a straightforward detector based on the Viola and Jones cascade [3] is not sufficient but does help to eliminate more than 95% of false positives at very high speed. To provide better performance, I utilize other useful features of hands like skin color and geometric properties to eliminate the remaining 5% of false positives.

I have conducted a thorough evaluation on our proposed system and found that its detection rate was 86.8% and that its false positive rate was 1.19 false detections per image on average. The system's speed and accuracy will enable many useful applications that are based on hand detection and tracking.

1.3 Organization Of This Thesis

This thesis is organized with six chapters:

Chapter 1 Brief introduction to hand detection and tracking, its applications, and an overview of the thesis.

Chapter 2 Literature review on hand detection and tracking not only in modern days but also in early time.

Chapter 3 Detailed explanation of the system architecture and each building block in my hand detection system.

Chapter 4 Thorough step-by-step description of how data acquisition, training and testing are carried out.

Chapter 5 Presentation and discussion of the results of the experiments described in Chapter 4.

Chapter 6 Conclusion and recommendations for those who would like to use or improve upon this work.

Chapter 2

Literature Review

Due to the limitations in computing technology and lack of powerful computer vision techniques in early days, tracking of of single hand in less cluttered or non-cluttered scene with high resolution imagery had been relatively difficult problem during that time. However, a lot of robust, real-time and useful hand trackers [4, 5, 6] had been introduced. Visual hand tracker [4], called DigitEyes can track a single hand with 27 degree of freedom in real-time from gray scale images at the speed of 10 frames per second. Early hand trackers are not limited only to 2D and some are capable of tracking hand in 3D space. One of the good example of 3D hand tracker is the real-time 3D hand tracker of Ahmad [5], which can operate at very fast speed of 30 frames per second. However those systems are not suitable for detecting and tracking hand in security application since they need high resolution imagery and detail of hand must be visible in the image.

Some early hand tracking systems like Pfinder [7] would be applicable to the hand detection problem for security. Pfinder is quite distinct in the fact that it attempts to follow the way humans look for the hand in images. Instead of directly detecting hands in an image, Pfinder looks for human bodies first and then easily segments out hands from the rest of the body by using skin color. However, since detecting humans in a cluttered video sequence is itself a very difficult problem, and the human body could easily be partially occluded in the scene, I try to bypass the human detection problem in our work by finding hands directly, without any attempt to find the entire human body first.

Over past 10 years, computer hardware and software technology is becoming more advanced, high computational power is available for researchers to implement powerful, sophisticated and computational costly algorithms in machine learning and image processing. Those powerful algorithms enable many robust and practically usable hand detectors and trackers. Most of the modern hand detectors and trackers are intended for human-computer action applications and very few researches had been done for security applications.

There are several approaches to hand detection and tracking. The first approach uses skin color information to segment hands from the background and then tracks segmented hands between frames using a tracking algorithm. The face and hand tracking system for sign language recognition [8] first segments the image into skin and non-skin regions using an elliptical model for skin pixels in C_bC_r space. Then face detection is used to locate the face skin blob ideally leaving only the skin blobs of hands. The system constructs a template for each hand then in subsequent frames, finds the region best matching that template using a minimum mean-squared error cost function. A similar approach is used by a real-time hand gesture system based on evolutionary search [9] to detect and track hands for human-robot interaction. The hands and face tracking system for VR application [10] also follows this approach but this system is extended to track hands and faces in 3D for a virtual reality application.

Some hand trackers used slightly different approach. The hand tracker of Shamaie and Sutherland [11] does not use skin color information, so it works on monochrome video sequences but slightly high resolution imagery and less clutter background seems to be required. Hands are extracted from the background using a blob analysis algorithm then tracked using a dynamic model from control theory. Unfortunately, the techniques used in these systems to locate hand is just a image segmentation based on regional properties and relying more on tracking. Since tracking algorithms normally require more than two consecutive frames, these approaches are not suitable when the goal is to extract hands from single images.

The approach used in open hand detection in a cluttered single image using finger primitives [12] is quite distinct from previously mentioned approaches. This system makes use of the geometric properties of the hand, such as parallel edges of fingers, without the use of skin color or motion information. Their proposed system is robust to the size and the orientation of hands with the limitation that one or more fingers must be visible. So, it is not applicable to my case due to its needs for high resolution imagery, although this system can detect hand in cluttered single image robustly.

However, there is another approach, where a detection window is scanned over the image and each of the scanned image patches are classified as hand or non-hand. In this approach, various object detection techniques are used for classification of scanned image patches and detector is able to locate hands in static images. This approach is used by robust hand detector [13], which is able to detect upright hand in pretty high resolution image. Their system utilized boosted classifier cascade object detector [14, 3] and poses of detectable hands are limited to six fixed postures. Because of limitation on the postures of hand, their system is not directly usable for my desired system where hand is assumed to be in arbitrary posture. A boosted classifier tree for hand shape detection [15] uses similar approach, but it constructs classifier tree instead of normal classifier cascade. Their system is able to not only detect hand but also classify the shape and posture of hand. But their system experienced the limitation of the constraints on the shape and orientation of hand making less applicable for my problem.

Very robust object detector, boosted classifier cascade [14, 3] is made even more powerful by introducing new set of Haar-like filters [16] and several variations of AdaBoost learning algorithm [17]. This improved system was demonstrated as a hand detector, which is able to detect a hand in infinite number deformations and poses, in research work for human-robot interaction based on Haar-like features and eigenfaces [18]. According to the illustration of hand detector output sequences, it seems that the detector was tested on the image sequence, in which only a single high resolution hand is present, in contrast to our case, in which multiples hands and entire humans' body may present in the scene. This makes system less suitable for applying to my problem without any modification and improvement.

The system intended for real-time hand tracking in crowded scenes [19] tried to utilized the motion information between two adjacent frames addition to the appearance information. The main idea behind this system is from pedestrian detector [20], in which AdaBoost learning algorithm learn to recognize not only in appearance pattern in a single image but also the pattern of motion between two consecutive frames. Unfortunately, this system does not work as good as in the case of detecting pedestrian and is not practically usable due to its high false positive rate. The main reasons for having high false positive rate is that the speed of hand motion is varied widely and the learning process is not able to generalize the motion pattern of hand.

From above literature review, some approaches can not be used for hand detection problem in security applications, although they are very efficient for their intended applications. However, approach based on general object detector of Viola and Jones [14, 3] and improved version of it [16, 17] are suitable starting point for my aimed system. Hand detector, mentioned in this thesis, is based mainly on this system with additional modules to improve the performance.

Chapter 3

Hand Detection System Architecture

A block diagram of my hand detection system is shown in Figure 3.1. A scan window is scanned over the input image at different scales and each of the resulting image patches is fed into a classifier cascade which rapidly determines whether the image patch is a hand. The classifier cascade eliminates more than 95% of the non-hand regions in a given image. However, due to the large number of candidate regions in one image, to be practical, the false positive rate must be further reduced. To serve this need, I add a post-processor to the system, denoted by dash-line block in Figure 3.1, to further reduce false positive detections. The post-processor takes advantage of a prior knowledge of hands color and geometry. Skin detection, feature extraction, and Mahalanobis classification are the essential building blocks of the post-processor.

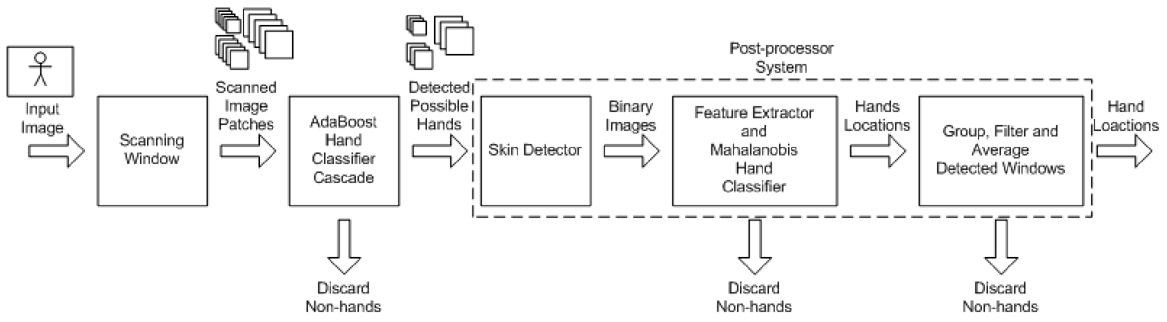


Figure 3.1: Hand detection system architecture. Scanned image pages are classified as hand or non-hand by a cascade based on Haar-like features then further filtered by a post-processor based on skin pixel blob analysis.

3.1 Scanning Window

When an image is presented to hand detection system (Figure 3.1), a detection window is scanned over the image at multiple scales, and each resulting image patch is passed to the boosted classifier cascade. The scanning process is to begin with a detection at the images original scale with the fixed scanning window size 24×24. After every possible image patch at that scale has run through the classifier cascade, the image is scaled down by 90% and the process is repeated until a minimum image size (maximum detection window size) is reached. The scanning window moves a pixel at a time in either x or y axis, while making sure that it scanned thorough out the whole image at every scale. The scanning process is shown as sudo-code in Figure 3.2.

```
Given : $Img$, an image
Return : $ImgSet$, a set of image patches

  1. Initialize $w_i = 1/N$ where $i = 1, \ldots, N$.
  2. Repeat while smallest side of the image $Img$ is equal or grater than
     24.
       (a) Scan $24 \times 24$ pixel window over $Img$ with a step size of one pixel.
       (b) Add scanned image patches to the image set $ImgSet$.
       (c) Downscale $Img$ to 90% of its current size.
  3. Return the image set $ImgSet$.
```

Figure 3.2: Explanation of how the scanning window extracts image patches from the image.

3.2 Boosted Classifier Cascade

The research work on rapid object detection using boosted cascade of simple features [14] originally proposed the cascade of boosted classifiers as a real-time general object detector and applied it to face detection [3]. Their work showed that the system can robustly detect faces in static images independent of the background. The system runs in real-time since the feature detector is limited to a class of Haar-like filters that can be computed in constant time with the help of integral images, regardless of the spatial extent of the filters. The speed of the system is increased even further by arranging the classifiers in a cascaded fashion, so that the early stages reject most of the image patches unlikely to contain the object of interest. The cascade therefore only spends significant compute time on the image patches most likely to contain the object of interest. The illustration shown in Figure 3.3 will give a clear insight of how cascading helps faster classification.

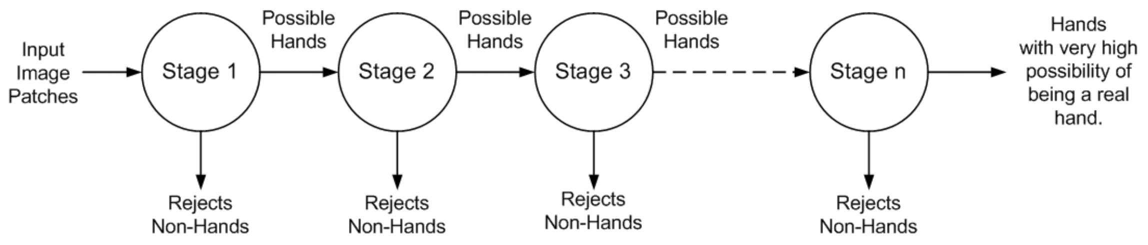


Figure 3.3: Classifiers in cascade arrangement speeds up the classification by rejecting most non-hand image patches

Each stage in the cascade is constructed from a set of simple Haar-like filters using AdaBoost algorithm [21]. AdaBoost builds a strong nonlinear classifier from multiple weak threshold classifiers, in this case each using a Haar-like filter, a threshold, and a weight, all of which are selected by AdaBoost to minimize the weighted error for the whole stage over the training set, while maintaining the desired detection rate. Original detector [14, 3] used the four types of Haar-like filters shown in Figure 3.4 (a). The filters can take on arbitrary positions and sizes within an image patch. The output of each filter is simply the difference between the

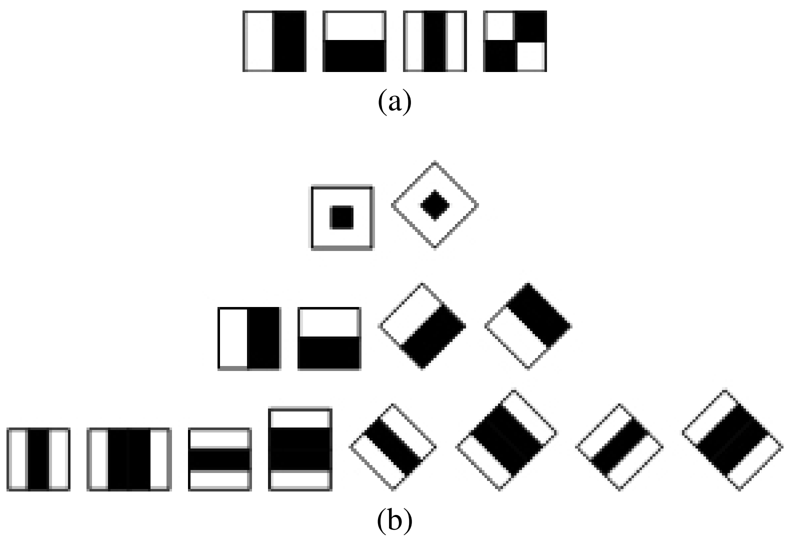


Figure 3.4: Haar-like features used to construct weak classifiers in the boosted classifier cascade. (a) Viola and Jones features. (b) Lienhart and Maydt features.

average pixel value within the clear rectangular regions and the shaded rectangular regions. They applied integral image technique to compute the value of Haar-like filters in constant time regardless of the type and size of filter.

An integral image is a special representation of image, on which rectangle Haar-like features can be computed very rapidly. The integral image is very similar to summed-area tables used in texture mapping [22] and it is one of the contributions to achieve object detector to operate in real-time. During detection, integral image for each input image is computed and then the classifier computes all required features with a few simple mathematical manipulations, only addition and subtraction.

Recently, an extended set of Haar-like features [16] are added to the original features [14, 3] for better performance. Their additional rotated Haar-like filter types are shown in Figure 3.4 (b). On a particular test set, they found that their modified system gave 10% fewer false positives than the original system for certain detection rates. The empirical analysis of detection cascade of boosted classifier [17] compared Discrete AdaBoost (the algorithm used by [14, 3]), Real AdaBoost [23], and Gentle AdaBoost[23]. The results from this comparison analysis showed that classifiers trained with Gentle AdaBoost performed the best.

Since the results from the work of [17] are very promising, I decided to used their system as a core boosted classifier cascade in my hand detection system. Due to outstanding performance over other boosting algorithms, Gentle AdaBoost is used as booting algorithm for training process with all full features set, shown in Figure 3.4 (b). The Gentle AdaBoost is very similar to AdaBoost algorithm but different in the way weights are updated. The brief idea of Gentle AdaBoost is shown in Figure 3.5 and please refer to the original work [23] for more detail.

As boosting algorithms are supervised learning algorithms, a large number of labeled positive and negative examples must be input to the training process. Positive examples manually located hands in the images and negative examples are extracted from are *background images*. The process to get positives examples and background images will be discuss in

```
Given :  $\alpha_{tp}$ the desired true positive rate
         $\alpha_{fp}$ the desired false positive rate
         $N$ examples $(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)$ where $x \in \Re^k$ and $y \in \{-1,1\}$
Return : A boosted classifier $H(x)$

  1. Initialize $w_i = 1/N$ where $i = 1,\ldots,N$.

  2. Repeat until required $\alpha_{tp}$ and $\alpha_{fp}$ is met, while increasing $M$ by 1 at
     each time.

       (a) Fit the regression function $f_M(x)$ by weighted least-squares of $y_i$
           to $x_i$ with weights $w_i$.
       (b) Set new weights $w_i \leftarrow w_i e^{-y_i f_M(x_i)}$ where $i = 1,\ldots,N$.
       (c) Renormalize new weights sum of all weight is equal to 1,
           $\sum_{i=1}^{N} w_i = 1$.
       (d) Find the $\alpha_{tp}$ and $\alpha_{fp}$ of current classifier $H(x) =$
           $sign\left[\sum_{m=1}^{M} f_m(x)\right]$ on the given $N$ examples.

  3. Output the classifier, $H(x) = sign\left[\sum_{m=1}^{M} f_m(x)\right]$.
```

Figure 3.5: Gentle AdaBoost algorithm used to train the classifier cascade of my hand detector system.

Chapter 4, Section 4.1. Besides those examples, two important learning parameters must be specified. Those parameters are the desired true positive and false positive rate for each stage of the cascade, and one stage is trained at a time until that stage achieves the specified true positive and false positive rates. Then a new stage is begun, and the process continues until some stopping criterion is reached. Only the positive examples that are correctly classified by the previous stages and the negative examples that are incorrectly classified by the previous stages are used to train each new stage. Figure 3.6 clearly shows how the training process train and arrange boosted classifiers in cascade order to improve detection speed.

Once we have trained the classifier cascade, image patches can be fed to the cascade for classification. Each stage in the classifier cascade classifies incoming image patch and passes to the next stage only if it is classified as a hand until the end of the cascade. The image patches that are positively classified by the last stage of cascade do have very high possibility to be true hands but high percentage of false positives are present. So, further post-processors are needed to reduce those false positives making the whole system to be practically usable.

The complete object detection system techniques mention above [14, 3, 16, 17] are implemented in Open Computer Vision Library [24]. The boosted classifier cascade training program allows users to train cascade with several choices of Haar-like feature sets and various boosting algorithms. I use this utility to construct boosted classifier cascade for my prototype hand detector system.

```
Given : $N_{stage}$ the desired number of stage in the cascade
        $\alpha_{tp}$ the desired true positive rate for each stage of cascade
        $\alpha_{fp}$ the desired false positive rate for each stage of cascade
        $NP_{stage}$ number of positive example to train each stage of cascade
        $NN_{stage}$ number of negative example to train each stage of cascade
        $P_{total}$ a set of $NP_{total}$ positive examples
            where $NP_{total} \geq [NP_{stage} + (N_{stage} \times \alpha_{tp} \times NP_{stage})]$
        $B_{image}$ back ground images from which negative examples $N_{examples}$
            will be extracted
Return : A cascade of boosted classifier $H_{cascade}$

    1. Extract $NP_{stage}$ of positive examples $P_{train}$ from given positive exam-
       ple set $P_{total}$.
    2. Extract $NN_{stage}$ of negative examples $N_{train}$ from background images
       $B_{image}$.
    3. Repeat until desired number stage $N_{stage}$ is exceeded in the cascade.
        (a) Train single boosted classifier $H$ that meets desired $\alpha_{tp}$ and $\alpha_{fp}$
            with positive examples $P_{train}$ and negative examples $N_{train}$ as de-
            scribed in Figure 3.5.
        (b) Add resulting boosted classifier $H$ to the cascade $H_{cascade}$.
        (c) Test resulting boosted classifier $H$ on training examples $P_{train}$
            and $N_{train}$.
        (d) Positive examples from $P_{train}$, misclassified by $H$, are replaced
            with new positive examples from positive example set $P_{total}$.
        (e) Negative examples from $N_{train}$, correctly classified by $H$, are re-
            placed with new negative examples extracted from background
            images $B_{image}$.
    4. Output the boosted classifier cascade $H_{cascade}$.
```

Figure 3.6: Cascading algorithm arranges boosted classifiers in cascade order to improve detection speed.

3.3 Skin Detector

There are many approaches to segmenting regions with similar color and texture from other regions. To extract skin color blobs from images, color information is the obvious choice. Nowadays, there are many skin detectors [25, 26, 27, 28, 29] available and some of those are very robust. The skin detector for my hand detection system need not be extremely robust but it should be fast.

The Bayesian maximum likelihood classifier based on color histograms [28], meets all of these needs. Based on their results and my own follow-up study, I selected the *HS* (*hue* and *saturation*) color model. The main reason to use only *H* and S while ignoring *I* (*intensity*) from *HSI* color space is to eliminate non-uniform illumination problem. Using only *HS* color model also saves computational cost and helps skin detector to work on different tone, i.e. different skin tones over Caucasian, Asian, African, and so on. Histograms used in the skin detector have two dimensions, namely hue and saturation. Each axis of the plane is quantized into 16 bins, so that each histogram will have $16\times16=256$ bins. I selected 16-bin quantization based on comparison experiments with different bins counts of 8, 16, 32, and 64. I found that 16 bins along each axis gave the best performance for my application.

I construct histograms for skin and non-skin pixels from a large training set. The normalized histogram counts are used to construct a discrete class-conditional likelihood for a Bayesian maximum likelihood classifier which I then use to determine whether a given pixel is most likely skin or not skin. To classify whether a given pixel $\overline{p}$ is skin pixel or not, *H* and *S* values are computed and quantized into 16 bins for each values. Then probability of being skin pixel $P(s|\overline{p})$ is found from skin histogram using those quantized *H* and *S*. From non-skin histogram, the probability of being non-skin pixel $P(\neg s|\overline{p})$ is found. According to the Bayesian maximum likelihood classification, pixel $\overline{p}$ is classified as skin if:

$$\frac{P(s|\overline{p})}{P(\neg s|\overline{p})} > 1.$$

Otherwise, $\overline{p}$ is considered as non-skin pixel.

Each image patch which is classified as a hand by the cascade is scaled to a standard size 24×24 pixels and then fed to the skin detector, which produces a binary image, in which the value 1 represents a putative skin pixel and the value 0 represents a non-skin pixel.

3.4 Features Extractor

The shape and relative size of the skin blob within the detection window give useful information for discriminating image patches containing hand from those not containing hand. There are a many popular and classical shape descriptors like shape signatures, Fourier descriptors and curvatures [30]. Even more powerful shape analysis and classification technique called shape context [31] is available. But those techniques demand high computational cost and are not suitable for classification pretty noisy shape in a small 24x24 pixels binary image. I did research to find the efficient features best suited to my application.

Finally, I got four simple features which can be used to discriminate between hand image patches and on-hand image patches. The binary images produced by skin detector may be

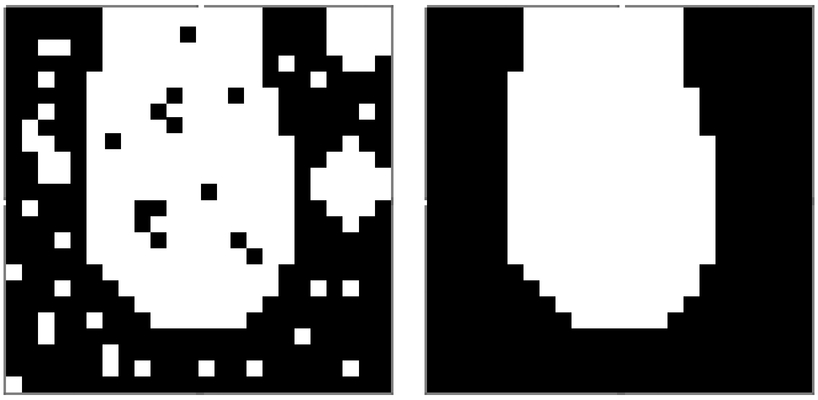

(a) (b)

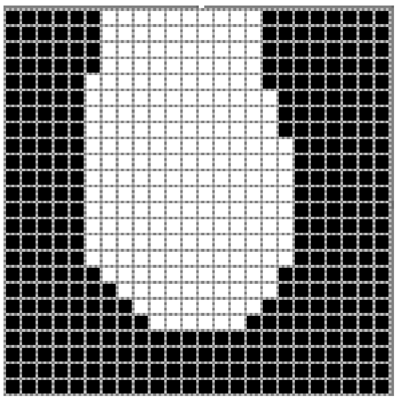
Figure 3.7: (a) Ideal skin detected image of proper hand, which may be produced by skin detector. (b)Largest connected skin blob of image (a), on which holes are filled. (Images are synthesized to model problem.)

noisy and probably contain multiple connect skin blobs. So, the largest connected blob is selected and fill the holes on it. The ideal skin detected image of proper hand is is shown in Figure 3.7 (a) and image (b) shows the hole filled largest connected skin blob. Then, feature extractor computes four simple features from the largest skin blob that are surprisingly useful for accurate classification:

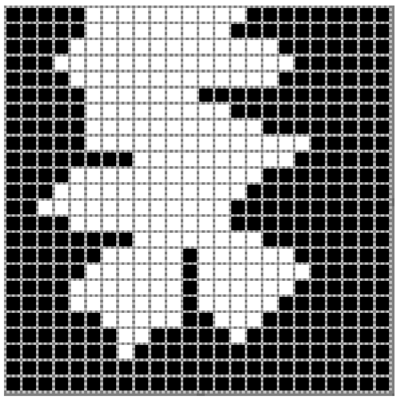
1. The area of the blob f_{area}.
2. The length of the perimeter of blob $f_{perimeter}$.
3. The eccentricity of the blob $f_{eccentricity}$.
4. The number of pixels on the boundary of blob that intersects the detection window boundary $f_{boundary}$.

The area feature f_{area} is simply the number of pixels in the largest connected skin component; it is normalized by the total number of skin pixels 24×24=576 in the image patch. It is very obvious that the given image patch is unlikely to contain a hand if the area feature is very large or very small. As shown in Figure 3.8, image patch containing very large (c) or small (d) can be easily discriminate from the image patch containing real hand (a) based only area feature. But (b) can not be discriminated from (a) only using area feature since the area value for both images are the same. Fortunately, there are other features which are able to handle this kind of problem.

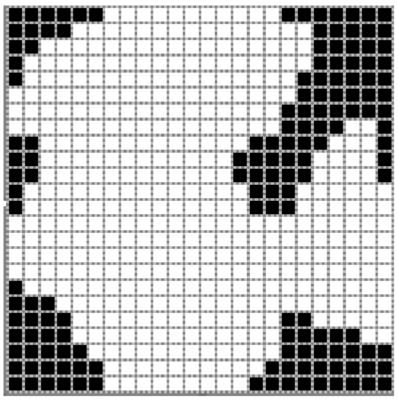
The perimeter feature $f_{perimeter}$ is the total number of pixels on the perimeter of the largest connected skin component; it is normalized in the same way as the area feature. I selected perimeter feature based on the fact that although area and other region properties of two shape may be the same, the perimeter can be different. Figure 3.8 (a) and (b) clearly demonstrate this idea. Both images are having slightly same area, eccentricity and boundary but perimeter feature helps to eliminate image containing non-hand (b) from the positive image patch (a). However, this feature is usable only for detecting hand in low-resolution condition where detail of hand is not visible. If detail of hand is visible the range of perimeter for



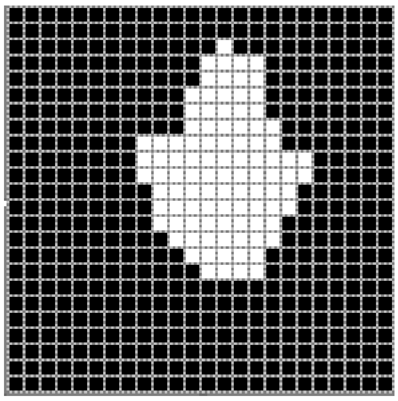
(a) Ideal Hand
f_{area} - 226
$f_{perimeter}$ - 62
$f_{eccentricity}$ - 0.7783
$f_{boundary}$ - 10



(b) Hand-like skin color blob
f_{area} - 226
$f_{perimeter}$ - 120
$f_{eccentricity}$ - 0.7755
$f_{boundary}$ - 10



(c) Non-hand very large skin blob
f_{area} - 424
$f_{perimeter}$ - 122
$f_{eccentricity}$ - 0.4885
$f_{boundary}$ - 37



(d) Non-hand small skin blob
f_{area} - 98
$f_{perimeter}$ - 47
$f_{eccentricity}$ - 0.7357
$f_{boundary}$ - 0

Figure 3.8: Illustration to demonstrate how useful four features are in the discrimination between image patches contain hand and those do not contain proper hand. Here, feature values for each image are absolute values, i.e. they are not normalized. (Images are synthesized to model problem.)

hand will be very large due to unlimited deformations and poses of hand in high-resolution imagery.

The eccentricity feature $f_{eccentricity}$ is basically the eccentricity of the ellipse having the same second moments as the largest connected skin component, i.e., the ratio of the distance between the foci of the ellipse and its major axis length. Eccentricity is computed with technique used by function *regionprops* in Image Processing Toolbox of MATLAB 6.5 [32]. We know x_i and y_i, locations of $x-y$ location of n skin pixels, from the skin detected binary image. Then, the second order moments relative to x axis μ_{xx}, y axis μ_{yy} and both axis μ_{xy} can computed as

$$\mu_{xx} = \sum_{i=1}^{n} \frac{(x_i - \overline{x})^2}{n} + \frac{1}{12}$$

$$\mu_{yy} = \sum_{i=1}^{n} \frac{(y_i - \overline{y})^2}{n} + \frac{1}{12}$$

$$\mu_{xy} = \sum_{i=1}^{n} \frac{(x_i - \overline{x}) \times (y_i - \overline{y})}{n}$$

where $\overline{x}$ is the mean value of x_i

$$\overline{x} = \sum_{i=1}^{n} \frac{x_i}{n}$$

and $\overline{y}$ is the mean value of y_i

$$\overline{y} = \sum_{i=1}^{n} \frac{y_i}{n}.$$

From those second order moments, the eccentricity is computed by using following mathematical formula.

$$f_{eccentricity} = \sqrt{\frac{2\sqrt{(\mu_{xx} - \mu_{yy})^2 + \left(4\mu_{xy}^2\right)}}{\mu_{xx} + \mu_{yy} + \sqrt{(\mu_{xx} - \mu_{yy})^2 + \left(4\mu_{xy}^2\right)}}}$$

The eccentricity value range is between 0 and 1, with 0 indicating a circle and 1 indicating a line segment. This feature helps to discriminate face skin regions and other round shape skin colored objects, which tend to be quite round, from true hand skin regions, which tend to be more eccentric.

Finally, the boundary feature $f_{boundary}$ helps to discriminate between arm skin regions, which tend to intersect the boundary of the detection window in two places, from true hand skin regions, which only intersect the detection window at the wrist. The boundary feature value is normalized by the total number of pixel on the perimeter of the detection window, $(24\times2)+[(24-2)\times2]=92$ in our case.The boundary feature provides information about how wrist-like the boundary is. Boundary feature also helps eliminate negative image patches containing skin color blob that intersects with detection window at several places or does not intersect at all. This can be seen clearly in Figure 3.8 (a), (c) and (d). Since skin-color blob in (c) intersects with detection windows at several places its boundary value is very much larger than that of image patch containing real hand (a). Moreover, boundary value of negative image patch (d) is zero, very much lesser than that of positive image patch (a) because skin-color blob in image patch (d) does not intersect with detection window boundary.

3.5 Mahalanobis Classifier

No matter how good those four features are, they will not be efficiently utilized for classification without a suitable classifier. There are many classification techniques available ranging from very simple statistical Gaussian classifier to complex state of art classifiers such as support vector machine (SVM). Among those, classical artificial neural networks are very popular choices.

I prefer classifiers that are simple with few parameters to tune and training process of the classifier to be simple. The computation cost of should be low so that there will be some possibility that hand tracking system or higher level applications based on my hand detector will be able to run in real-time. I found that a simple classifier based on Mahalanobis distance is a reasonable choice. The advantage of using Mahalanobis distance over Euclidean distance is that it take account the patterns of covariance that exist in the data during computation [33].

Each image patch can be represented by a feature vector consisting of the area, perimeter, eccentricity, and boundary features. To classify a given feature vector $\vec{f}$ as a true hand or not a hand, we calculate the Mahalanobis distance

$$d_{Mahalanobis} = (\vec{f}-\vec{\mu})^T \Sigma^{-1} (\vec{f}-\vec{\mu})$$

between the feature vector $\vec{f}$ and the mean feature vector μ, then we classify $\vec{f}$ as a hand if $d_{Mahalanobis}$ is less than some threshold θ. Here the mean hand feature vector $\vec{\mu}$, the covariance matrix Σ and they are computed from the four features of given sample positive examples matrix:

$$f_{sample} = \begin{bmatrix} \vec{f_1} \\ \vec{f_2} \\ \vdots \\ \vec{f_n} \end{bmatrix} = \begin{bmatrix} f_{area_1} & f_{perimeter_1} & f_{eccentricity_1} & f_{boundary_1} \\ f_{area_2} & f_{perimeter_2} & f_{eccentricity_2} & f_{boundary_1} \\ \vdots & \vdots & \vdots & \vdots \\ f_{area_n} & f_{perimeter_n} & f_{eccentricity_n} & f_{boundary_n} \end{bmatrix}.$$

So, f_{sample} is n$\times$4 matrix, where n is the total number of *sample positive image patches*. The procedure to get *sample positive image patches* will be described in Chapter 4, Subsection 4.2.3. The mean hand feature vector is simple computed from f_{sample} as:

$$\vec{\mu} = \begin{bmatrix} \mu_{area} & \mu_{perimeter} & \mu_{eccentricity} & \mu_{boundary} \end{bmatrix} = \frac{1}{n}\sum_{i=1}^{n} \vec{f_i}.$$

Then, covariance matrix Σ is computed by using following formula:

$$\Sigma = \frac{\chi^T \times \chi}{n}$$

where χ is simply computed by subtracting $\vec{\mu}$ from each row of the matrix f_{sample}:

$$\chi = \begin{bmatrix} \vec{f_1}-\vec{\mu} \\ \vec{f_2}-\vec{\mu} \\ \vdots \\ \vec{f_n}-\vec{\mu} \end{bmatrix}$$

and dimensions is same as dimensions of matrix f_{sample}. The resulting covariance matrix σ is 4×4 matrix. The distance threshold θ are estimated from the training set and will be discussed more in Chapter 4, Subsection 4.2.3.

Once classification for each possible detection windows is done, the positively detected hands are fed to the final module, the grouping, filtering, and averaging module. Further reduction of false positives is done there.

3.6 Grouping, Filtering, and Averaging Module

The Mahalanobis classifier produces a few very sparsely distributed false positives and densely distributed true detections around the actual targets. Since it produces several true detections around each of the actual detections, grouping and averaging is necessary to ensure only one detection for each target. I use the existing implementation of this technique in the OpenCV.

Grouping process groups the nearby detections together. Given square shape detection D_1 and D_2 are taken to be in the same group if the following requirements are met.

1. D_{2x} is in the range of $[D_{1x} \pm (D_1 w \times 0.2)]$,
2. D_{2y} is in the range of $[D_{1y} \pm (D_1 w \times 0.2)]$,
3. $D_{2w} \leq (D_{1w} \times 1.2)$, and
4. $D_{1w} \leq (D_{2w} \times 1.2)$.

Where D_{1x} and D_{1y} is the x, y location of D_1
D_{2x} and D_{2y} is the x, y location of D_2
D_{1w} is the width of D_1
D_{2w} is the width of D_2.

Once grouping is done, a group which contains less than some number of detections can be disposed of on the assumption it is a false positive. Then the averaging process takes the average of detection windows in each remaining group resulting only one detection window for each group. The positively detected hands output from this module could then be forwarded to another component in an integrated application, for example a gesture recognition module. But, in this thesis, I simply evaluate the performance and efficiency of the proposed algorithm on a series of video sequences. The following chapter describes my experiments in detail.

Chapter 4

Experimental Procedures

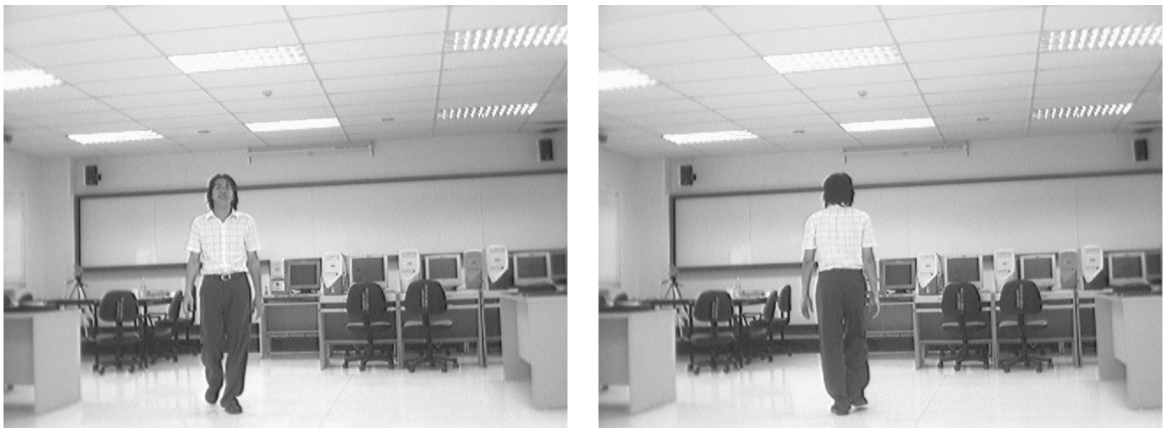
4.1 Data Acquisition

For the purpose of training, testing and evaluation of the proposed hand detection system, I captured 12 video sequences in a moderately cluttered laboratory environment. Four people volunteered to be models, and I captured three video sequences for each person. In the first sequence, each model walked away from the camera then came back to the starting position, in a direction parallel to the camera angle (Sequence 1, Figure 4.1). In the second sequence, each model walked back and forth across the field of view in a direction perpendicular to the camera angle, at three different distances from the camera(Sequence 2, Figure 4.1). In the last sequence, each model walked diagonally across the field of view, starting from a position to the right or left of the camera then returned to the start position, and repeated the procedure beginning from the other side of the camera (Sequence 3, Figure 4.1).

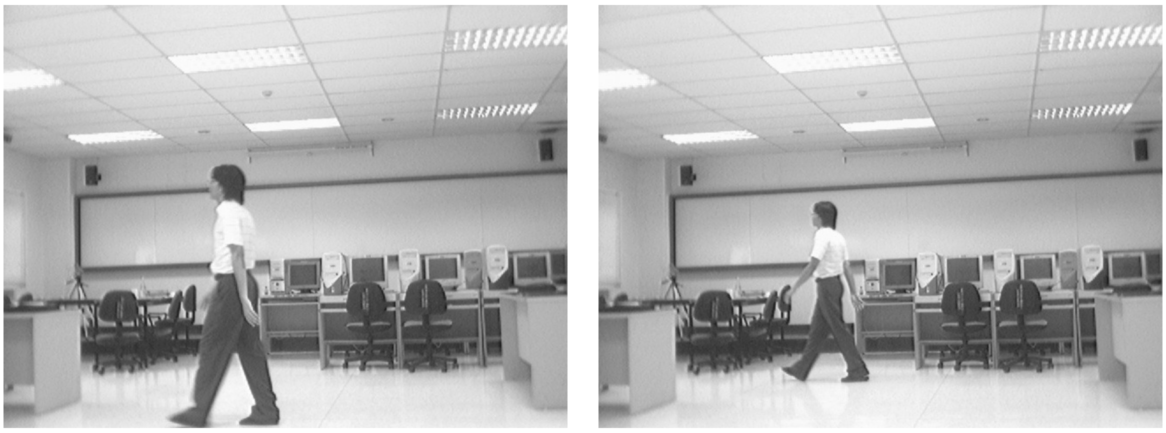
I captured the video sequences at 15 frames per second with an inexpensive *Fire-i* IEEE1394 web camera at a resolution of 640×480 pixels. Each sequence lasted approximately 30 seconds. After video capture, all visible hands not smaller than the standard size of 24×24 pixels in every image of all 12 sequences were manually located. A total of 2,246 hand locations were obtained. Our criteria for locating the selection window on the hand was that the hand should be roughly at the center of the window while taking up about 50% of the pixel area of the selection window. Some examples are shown in Figure 4.2.

Of the 12 video sequences, 11 were used to train the system and the remaining sequence was reserved for testing and evaluating the complete hand detection system. To train the boosted classifier cascade, I used 2,000 hands as positive examples, and negative examples were automatically extracted from a set of background images. As background images, I used four randomly selected images from the video sequences that did not contain any human. I created an additional set of background images using six randomly selected images containing humans. From each image, I cut out two large regions that did not containing hands but did contain other body parts such as faces and arms. From the test image sequence, I selected 99 images, each of containing at least one hand not smaller than 24×24 pixels. These 99 images contained a total of 106 proper hands. All of our test evaluation calculations are based on those 106 proper hands. I also prepared a holdout set by randomly selecting 100 images from the 11 training video sequences. This holdout set was used to monitor system performance as well as to tune system parameters.

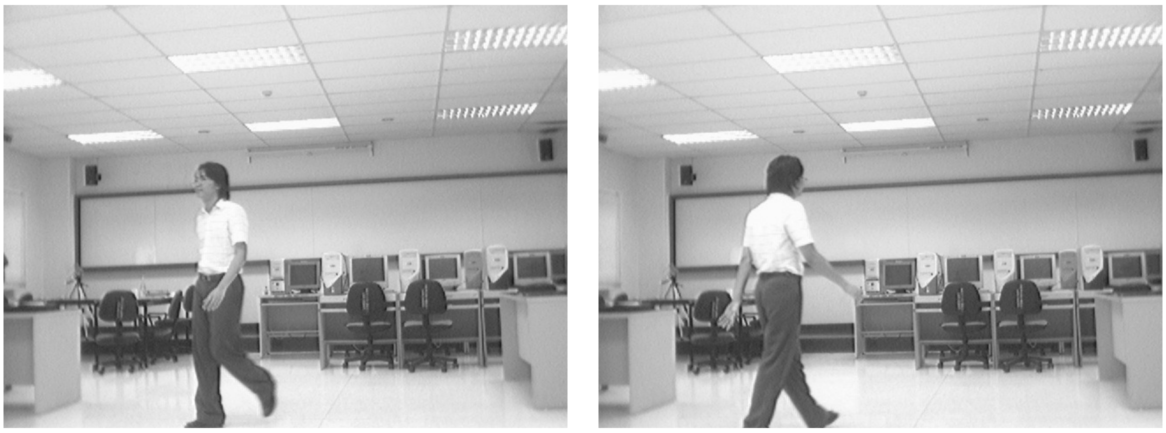
I also captured 10 images at two different locations under different lighting condition to train skin and non-skin histograms for skin detector. Skin pixels are manually marked on those images by creating binary mask images where 1's represent skin pixels and 0's represent non-



Sequence 1



Sequence 2



Sequence 3

Figure 4.1: Example images from three video sequences taken while person is walking. The walking path of the model is parallel to the direction of camera and it is perpendicular to camera's focal axis in sequence 2. In sequence 3, model is walking diagonally across the field.

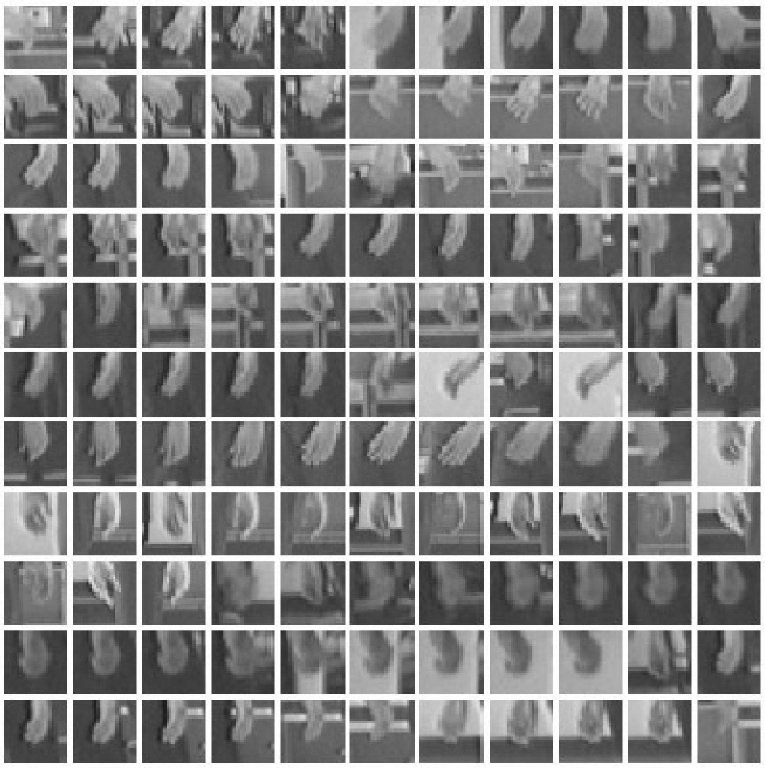


Figure 4.2 Example positive training images scaled to 24×24 for better visualization.

skin pixels as show in Figure 4.3. From 10 images, I obtained 70,475 skin and 1,203,094 non-skin pixels for the training of two histograms required for skin detector.

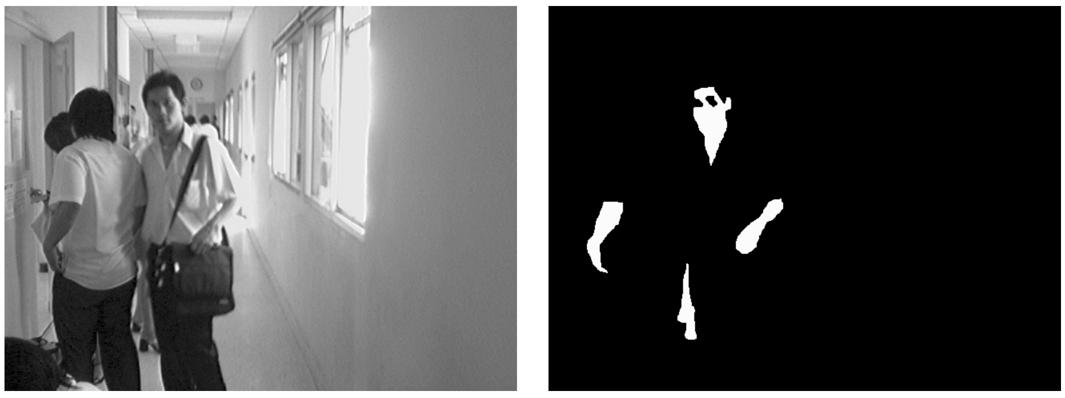
4.2 Training and Testing

To construct and test the hand classification system, I begin by training of a boosted classifier cascade on positive and negative examples of hands. Then I train a skin detector using positive and negative examples of skin pixels. Based on the positive detections by the classifier cascade and the output of the skin detector, I train a Mahalanobis classifier to filter out false positives. To complete the classifier, I determine the optimal minimum number of positive detections in a region of the image to be classified as a hand. For these steps I use a training set of 11 video sequences. Finally, after all system parameters have been determined, I test on a never-seen twelfth video sequence. The details of each step are described in the following subsections.

4.2.1 Boosted Classifier Training

To train the classifier cascade, I used the previously described 2000 manually located hands from the eleven training video sequences as positive examples and the 16 previously described background images.

The important parameters of the training process are the minimum hit rate (true positive rate α_{tp}) and maximum false alarm rate (false positive rateα_{fp}). Every stage in the cascade must



(a) (b)

Figure 4.3: Example images used for skin detector training. (a) Original color image. (b) Binary mask image, in which white pixels represent skin pixels and black pixels represent non-skin pixels on the image (a).

satisfy these criteria on the training set. I used 100% for the hit rate α_{tp} and 60% for the false alarm rate α_{fp}. This means when adding a new stage to the classifier, the training system keeps adding additional weak classifiers to that stage until it correctly classifies all of the positive training examples with at most a 60% false alarm rate. The training system extracts the desired number of negative examples, 4,000 for my experiment, by scanning a window with different scales over the background images. After training one stage of the classifier, the negative examples which are correctly classified are disposed of and the system extracts a sufficient number of new negative examples. As mentioned in section 3.2, I used the Gentle AdaBoost [23] variant of AdaBoost and the full Haar-like feature set [16].

During the training process, I monitored the performance of the cascade on the holdout set by examining both true positive rate and false positive rate every time new stage is added to the cascade. By examining true positive and false positive rates displayed in Table 4.1, it is obvious that cascade containing 12 stages of strong classifiers gave the optimum performance. The 12-stage classifier had a 97.5% detection rate on the holdout set, while having a reasonably low false positive rate of about 0.3% on the holdout set. A false positive rate of 0.3% may seem quite low but in fact this means I had an average of 1,033 false positive detections per image because one image contains more than 300,000 possible image patches. Clearly, these results indicate that a post processor is necessary to further eliminate false positives if the system is to be usable in practical applications.

The training was done on Pentium 4 desktop personal computer with the processor speed of 3.00 GHz, 1.00 GB of RAM, and took about seven days to complete training of 14 stages.

4.2.2 Skin Detector Training

To train our skin detector, previously mentioned 10 training images and their mask images were fed to the skin detector training process. The training process computes the hue (H) and saturation (S) for each pixel and quantizes each value into one of 16 bins. From the quantized values of skin pixels that is represent by value 1 on mask image, 2D histogram for skin was constructed. Another histogram is constructed from the quantized values of the non-skin pixels. Both histograms were constructed by simply counting the number of pixels

Table 4.1: Table showing the performance of the cascade on the holdout set for each time new stage is added.

Cascade Size	Number of True Positive (Total 121 hands)	Number of False Positive (for 100 holdout images)
1	121	More than 10^7
2	121	More than 10^7
3	121	More than 10^7
4	121	6,067,366
5	121	3,353,965
6	121	1,893,092
7	121	1,032,368
8	120	613,260
9	119	389,034
10	118	219,463
11	118	145,372
12	118	103,393
13	115	73,085
14	115	54,466

which belong to same bin, and they were normalized by the total number of pixels used to construct the histogram.

4.2.3 Mahalanobis Classifier Training

The purpose of the Mahalanobis classifier is to eliminate the false detections made by the boosted classifier cascade while still maintaining a high detection rate. As the detection window is scanned over every image in the training set, the boosted classifier outputs both true positive and false positive image patches. I found 78,658 true positives on our training set then randomly selected 6,000 true positives as *sample positive image patches* for computing the mean feature vector μ and covariance matrix Σ for the Mahalanobis classifier.

After I obtained μ and Σ for the Mahalanobis classifier, I need to find the optimum threshold. To do so, I scanned a detection window over every image in the holdout set and separated the detected image patches into false positives and true positives using the known hand locations for the holdout set. I extracted the Mahalanobis classifiers four features from each detected image patches and calculated the Mahalanobis distance between the feature vector of each image patch and the mean feature vector . As the class for each image patch is known, I plotted the ROC curve as shown in Figure 4.4. At this point, a detection rate of less than 100% is acceptable because the classifier cascade typically produces multiple true detections around each hand. Examining the ROC curve, I found that a Mahalanobis distance of 2.9 is a reasonable threshold since this threshold gives a very low false positive rate (6%) while giving an acceptable true positive rate (60%) on the image patches output by the classifier cascade.

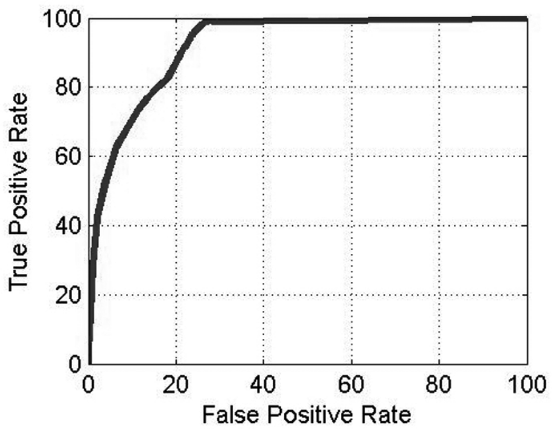


Figure 4.4: ROC cure between true positive and false positive for different threshold on Mahalanobis distance. True positive and false positive rate are calculated based on number of true detection and false detection input to the Mahalanobis Classifier.

4.2.4 Parameter Tuning for the System

Once all the required building blocks for hand detection are in place, I need to specify one last parameter, i.e., the minimum number of nearby positive image patches required for the Group, Filter, and Average block. In practice, this parameter must be tuned to achieve a good detection rate. To tune this parameter, I assembled all of the building blocks into a complete system then tested it on the holdout set with various values for neither parameter. I found that a minimum of 4 neighboring patches produced the optimal result: 81.8% of the hands in the holdout set were detected and the false positive rate was also relatively low, an average of 1.55 false positives per image.

4.2.5 Testing the Complete System

I tested the complete hand detection system on the test set that was never used in any part of the training process. As previously described I used 99 images containing 106 hands in known locations. The detailed results of the test are discussed in the next chapter.

4.3 Summary of Experimental Specifications

The experimental specifications for the experiments conducted in my study are clearly summarized in Table 4.2.

Table 4.2 Summary of the experimental parameters.

Data Acquisition	Total Number of video sequence	12
	Frame size	640×480
	Frame rate	15 fps
Boosted classifier cascade training	Number of video sequence used	11
	Number of positive examples	2000
	Number of negative examples for each stage	4000
	Desired true positive rate α_{tp} for each stage	100%
	Desired false positive rate α_{fp} for each stage	60%
	Total number of stage	12
	Total number of weak classifier	391
Skin detector training	Number of skin pixel	70,475
	Number of non-skin pixel	1,203,094
	Number of bin	16×16=256
Mahalanobis classifier training	Number of positive sample	3000
	Threshold on Mahalanobis distance	2.9
Group, Average and Filter	Minimum number of member in eligible group	4
Testing	Number of video sequence used	1
	Total number of image tested	99
	Total number of eligible hand	106

Chapter 5

Results and Discussion

5.1 Experimental Results

When system was tested on the test set, results turned out very satisfactory. The final hand detector detects 92 hands (86.8%) of the 106 hands in the test set, with an acceptable false positive rate of 1.19 false detections per image on average. A detection rate of 86.8% will enable many applications based on hand detection. All hands detected are shown in Figure 5.1. Although all detected hand image patches are scaled down to 24×24 pixels for easy visualization, actually size image patched varied from 25×25 to 65×65 pixels. By examining the result detections, I found that ratio between the size of the hand and detection windows is almost constant. So, it is possible to predict the size of hand from the size of the detection window.

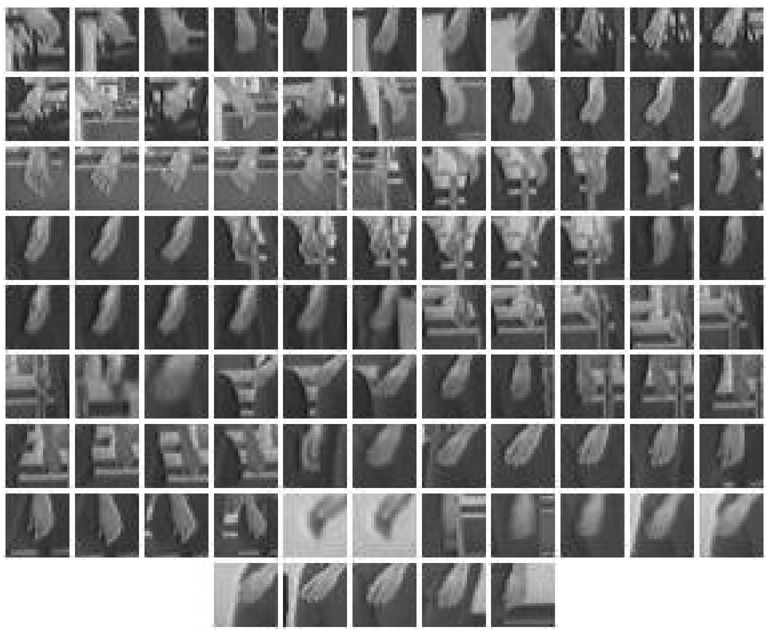


Figure 5.1: Hands detected by our complete hand detector system. All detections are scaled down to standard size 24x24 pixels for easy visualization.

For each detection, decision making of whether it is true positive or false positive is based on ground truth bounding square box previously put around hands as described in Section 4.1. For accepting a given detection as a true detection, the following criteria must be satisfied.

1. The Euclidean distance between a candidate detection and ground truth detection must

not exceed the size of the ground truth detection window.

2. The size of a candidate detection window must not be smaller the the half size of the ground truth detection window.

3. The size of a candidate detection window must not exceed the twice of the ground truth detection window size.

The summary results from my study is shown in Table 5.1.

Table 5.1: Summary of the result from the testing of complete hand detector system on the test set containing 99 images.

Number of image in test set	99
Number of eligible hand	106
Number of hand detected	92
Detection rate	86.8%
Total number of false positive	126
Average number of false positive per image	1.19

5.2 Discussion

The example detections on six consecutive images from the test video sequence by complete hand detector system are shown in Figure 5.2. All hands were detected in all six images but detection on the right hand in Image 2 is not count as positive due to its size. The false detection on the the desktop computer in the middle of scene is present in all six images and almost all the rest of the images in the test set. The main cause of this false positive is that the computers color and texture are in fact similar to that of a hand. So, any stage of the system could not reject this false detection. But this kind of false positive detection on a stationary object will be eliminated if we utilize motion information between two consecutive frames in the video sequence. The negative result of using motion information is that only moving hands will be detected and detector will miss the stationary hands.

I have also analyzed the performance of each module of the system. Over 10^7 false positives from all 99 images in test set are reject only by boosted classifier cascade very rapidly, while preserving 105 hands of total 106 hands. There are still 79,362 false positives for 99 images (about 800 false positives per image) and it is not practically usable. But 95.28% of those positives are surprisingly reduced by skin detector, feature extractor and Mahalanobis classifier, leaving only 3,748 false positives for whole test set, i.e. only average 37.85 false positives per image. At this point, total of three true hands are lost, one is wrongly rejected boosted classifier cascade and the other two are misclassified by Mahalanobis classifier. Finally, the group, filter and average module could get rid of all false positives, except false detection on the desktop computer as mentioned before. The reduction in average number false positive for one image at each important modules of the system can be clearly seen in Figure 5.3. Moreover, Table 5.2 shows false positives after each stage of system on six images shown in Figure 5.2.

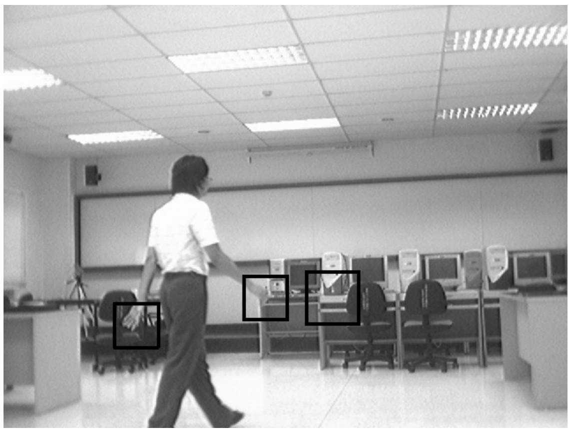


Image 1
(Hit - 2, Miss - 0, False Positive - 1)

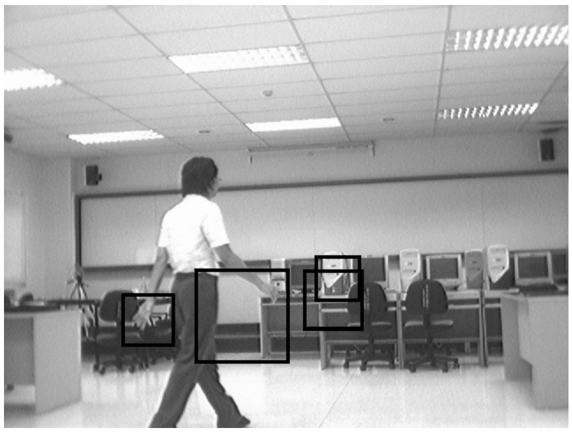


Image 2
(Hit - 1, Miss - 1, False Positive - 3)

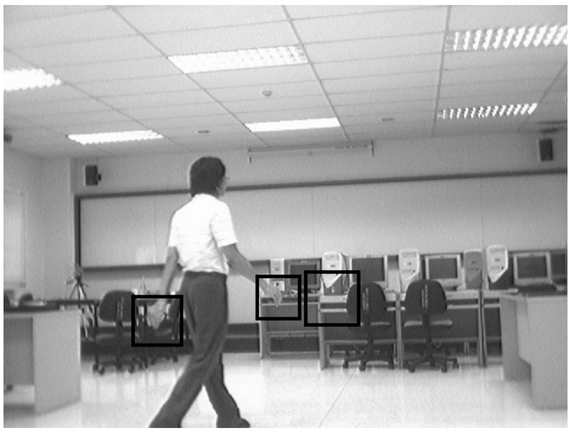


Image 3
(Hit - 2, Miss - 0, False Positive - 1)

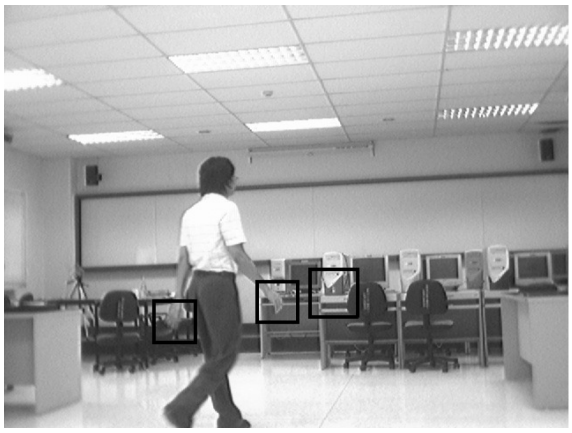


Image 4
(Hit - 2, Miss - 0, False Positive - 1)

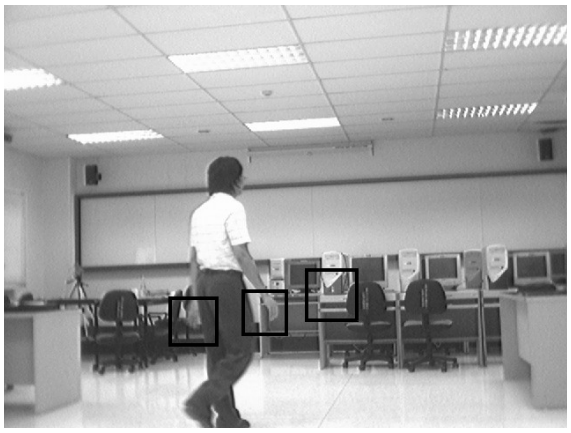


Image 5
(Hit - 2, Miss - 0, False Positive - 1)

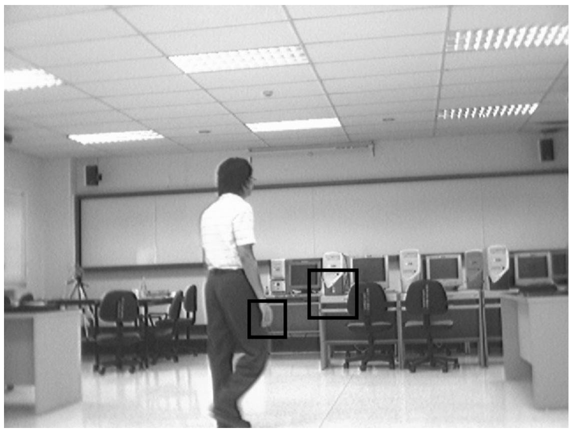


Image 6
(Hit - 1, Miss - 0, False Positive - 1)

Figure 5.2: Example detection results of our proposed hand detector system on six consecutive frames from test image sequence. Detection on right hand in image 2 is considered as false detection because of the relatively huge detection window size.

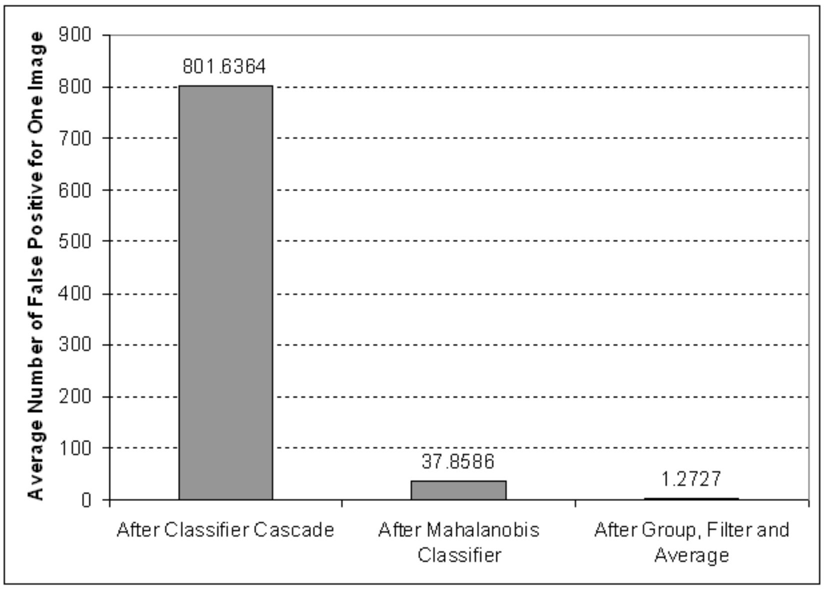


Figure 5.3: Bar graph showing the reduction of false positives at each important stage of the hand detector system.

Table 5.2: False positives after each important modules of the detector system. This table covers only six consecutive frames (shown in Figure 5.2) from test set and gives the idea of how each stage of the system reduce false positives on each image.

Image Name	Number of FP after Cascaded Classifier	Number of FP after Mahalanobis Classifier	Number of FP after Group, Filter and Average
Image 1	749	40	1
Image 2	762	56	3
Image 3	811	38	1
Image 4	933	20	1
Image 5	1047	96	1
Image 6	978	30	1

Most of the 14 missed hands from test set have relatively large detection windows around it, i.e. system really did not missed those hands. During evaluation of the performance of the system, I considered those large detection windows on hands as a false detections because those detection windows do not meet the criteria mention above. This kind of error is caused mainly by the groping and averaging operations. During grouping process, nearby false detections are taken in to the group of real detections because those false detections meet the criteria to be considered as a group member of real detections. Then group with widely spread member detections are formed and when those detections are averaged, the resulting detection window becomes relatively large to be considered as real detection. However, position and size of those missing hands can be estimated from two adjacent frames if detector is working on video sequence.

From the through analysis, I also found that boosted classifier cascade sometimes classified arm as a hand, as shown in image patch 2 of Figure 5.4. For human, it is quite obvious that image patch 1 from Figure 5.4 is a hand and image patch is not. But my loosely trained cascade could not discriminate between those two images. It is even difficult problem for Mahalanobis classifier since area, perimeter and eccentricity values of image patch 2 are very similar to those of image patch containing actual hands as you can see in the skin detected images of image patch 1 and 2 in Figure 5.4. Fortunately, boundary feature is good enough eliminate this kind of false positives because number of boundary pixels in image patch 2 is almost twice of the number of boundary pixels in image path 1. However, most of the false positives produced by boosted classifier are sure to be eliminated mainly based on area feature by Mahalanobis classifier since most of the false positives do not contain any skin color pixels, or contain very less skin pixels.

[h]

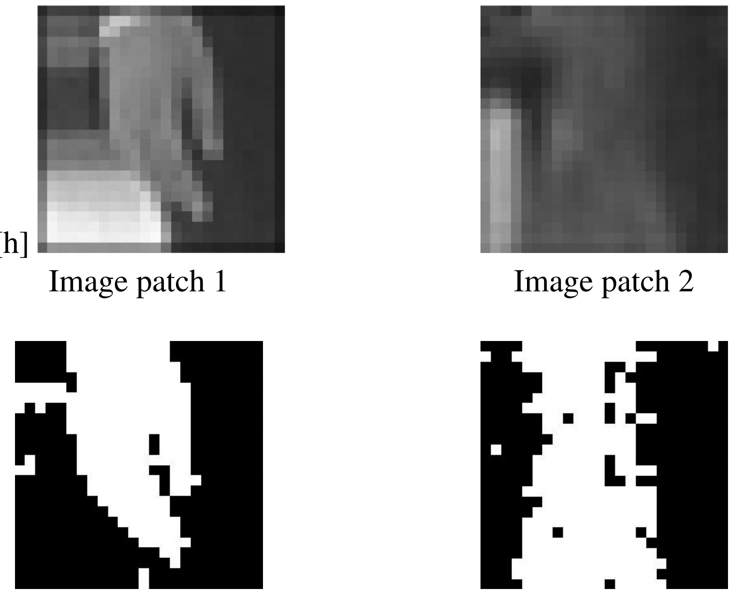


Image patch 1
Image patch 2
Image patch 1 after skin detection
Image Patch 2 after skin detection

Figure 5.4: Both image patch 1 and 2 are classified as hand by the boosted classifier cascade. Skin detected images of those two images are shown below each of the images. Boundary feature extracted from skin detected binary image helps in elimination of this kind of false detection.

Since, my hand detection system mentioned in this thesis is only a prototype system, the speed of the whole system could not be measured. However, the processing time for boosted classifier cascade may not exceed 150 ms on modern desktop computers. The approximated computing time for post-processor system is less than 100 ms. So, my hand detector is expected to be able to detect hands at 4 frames (640x480 resolution) per second.

Chapter 6

Conclusion and Recommendations

From the literature, we know that hand detectors incorporating boosting and Haar-like features perform quite well in applications like sign-language recognition, in which images are relatively high resolution with less cluttered background and constrained hand gesture. These approaches suffer from high false positive rates and low detection rates when applied to detect less constrained hands in low resolution and cluttered images.

However, I find that these limitations can be overcome with the help of a simple but efficient post processing system. In my experiments, the prototype hand detection system achieved excellent performance on its test set. One important limitation of this work is that both the training and testing image sequences were captured in the same environment. This means that the performance of our current system is likely background dependent; if so, the reported performance is optimistic. Another possible limitation is that the system may not work well on the image, taken while human is the motion beside normal walking. I had doubt about this limitation because the classifier cascade was trained only on the hands of working humans. So far, current results indicate that my hand detector can be efficiently used to locate hands for applications such as gesture recognition and human action recognition systems.

As recommendations for those who would like to make use of or improve my system, I would like to recommend finding new features, which help discriminate true hand from false hand more, from skin detected binary image. Then, more false positives can be eliminated. Although I used boosted classifier cascade based on the rapid object detector [14, 3] because of its rapid detection, it is not much robust to the rotation of hand according to the analysis on the rotational robustness of hand detection with Viola-Jones detector [34]. The performance of the system will be improved, especially in detection rate, if boosted classifier cascade can be replace with a high speed classifier that is more robust to rotation. The tuning of system parameters such as threshold of Mahalanobis classifier and number of neighbor may help improving detection rate while reducing false positive rate.

Bibliography

[1] H. Rowley, S. Baluja, and T. Kanade, “Neural network-based face detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 20, pp. 23–28, 1998.

[2] D. Roth, M. Yang, and N. Ahuja, “A SNoW-based face detector,” in *Advances in Neural Information Processing Systems (NIPS)*, pp. 855–861, 2000.

[3] P. A. Viola and M. J. Jones, “Robust real-time face detection,” *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[4] J. M. Rehg and T. Kanade, “Visual tracking of high DOF articulated structures: An application to human hand tracking,” in *Third European Conference on Computer Vision*, pp. 35–46, 1994.

[5] S. Ahmad, “A usable real-time 3D hand tracker,” in *Proceedings of the 28th IEEE Asilomar Conference on Signals, Systems and Computers*, pp. 1257–1261, 1995.

[6] J. Segen and S. Kumar, “Human-computer interaction using gesture recognition and 3D hand tracking,” in *Proceedings of the IEEE International Conference on Image Processing*, pp. 188–192, 1998.

[7] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, “Pfinder: Real-time tracking of the human body,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[8] N. Soontranon, S. Aramvith, and T. H. Chalidabhongse, “Face and hands localization and tracking for sign language recognition,” in *International Symposium on Communications and Information Technologies*, pp. 1246–1251, 2004.

[9] J. Wachs, H. Stern, Y. Edan, M. Gillam, C. Feied, M. Smith, and J. Handler, “A real-time hand gesture system based on evolutionary search,” in *Genetic and Evolutionary Computation Conference*, 2005.

[10] J. Varona, J. M. Buades, and F. J. Perales, “Hands and face tracking for VR applications,” *Computers & Graphics*, vol. 29, no. 2, pp. 179–187, 2005.

[11] A. Shamaie and A. Sutherland, “Hand tracking in bimanual movements,” *Image and Vision Computing*, vol. 23, no. 13, pp. 1131–1149, 2005.

[12] M. B. Caglar and N. Lobo, “Open hand detection in a cluttered single image using finger primitives,” in *Proceeding of the 2006 Computer Vision and Pattern Recognition Workshop*, 2006.

[13] M. Kölsch and M. Turk, “Robust hand detection,” in *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, 2004.

[14] P. A. Viola and M. J. Jones, “Rapid object detection using a boosted cascade of simple features,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 511–518, 2001.

[15] E.-J. Ong and R. Bowden, "A boosted classifier tree for hand shape detection," in *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 889–894, 2004.

[16] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 1, pp. 900–903, 2002.

[17] R. Lienhart, A. Kuranov, and V. Pisarevsky, "Empirical analysis of detection cascades of boosted classifiers for rapid object detection," tech. rep., Microprocessor Research Lab, Intel Labs, 2002.

[18] J. Barreto, P. Menezes, and J. Dias, "Human-robot interaction based on Haar-like features and eigenfaces," in *Proceedings of the 2004 IEEE Conference on Robotics and Automation*, pp. 1888–1893, 2004.

[19] M. N. Dailey and N. Bo Bo, "Toward real-time hand tracking in crowded scenes," in *The 2005 Asian Conference on Industrial Automation and Robotics*, 2005.

[20] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *IEEE International Conference on Computer Vision (ICCV)*, vol. 2, pp. 734–741, 2003.

[21] Y. Freund and R. E. Shapire, "A decision-theoretic generalization of online learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 5, no. 1, pp. 119–139, 1997.

[22] F. Crow, "Summed-area tables for texture mapping," in *Proceedings of SIGGRAPH*, vol. 18, pp. 207–212, 1984.

[23] J. Friedman, T. Hastie, and R. Tibshirani, "Hand tracking in bimanual movements," *Annals of Statistics*, vol. 28, no. 2, pp. 337–374, 2000.

[24] Intel Corporation, "OpenCV Computer Vision Library (software)." Open source software available at `http://sourceforge.net/projects/opencv/`.

[25] D. Saxe and R. Foulds, "Toward robust skin identificaiton in video images," in *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pp. 379–384, 1996.

[26] R. Kjeldsen and J. Kender, "Finding skin in color images," in *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pp. 312–317, 1996.

[27] M. J. Jones and J. M. Rehg, "Statistical color models with application to skin detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999.

[28] B. D. Zarit, B. J. Super, and F. K. H. Quek, "Comparison of five color models in skin pixel classification," in *International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems*, pp. 58–63, 1999.

[29] B. Jedynak, H. Zheng, and M. Daoudi, “Skin detection using pairwise models,” *Image and Vision Computing*, vol. 23, no. 13, pp. 1122–1130, 2005.

[30] L. da Fontoura Costa and R. M. Cesar Jr., *Shape Analysis and Classification, Theory and Practice*. CRC Press, 2000.

[31] S. Belongie, J. Malik, and J. Puzicha, “Shape matching and object recognition using shape context,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 24, pp. 509–522, 2002.

[32] The MathWorks, Inc, “MATLAB, The Language of Technical Computing (software).”

[33] J. Lattin, J. D. Carroll, and P. E. Green, *Analyzing Multivariate Data*. Thomson Learning, Inc., 2003.

[34] M. Kölsch and M. Turk, “Analysis of rotational robustness of hand detection with a viola-jones detector,” in *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 3, pp. 107–110, 2004.

Appendix A

List of Publications

National Journal

- Nyan Bo Bo, Matthew N. Dailey and Bunyarit Uyyanonvara. Natural-posed Hand Detection in Low-resolution Images. *Songklanakarin Journal of Science and Technology*. (Submitted Manuscript)

International Conferences

- Matthew N. Dailey and Nyan Bo Bo. Toward Real-Time Hand Tracking in Crowded Scenes. In *Proceedings of the 2005 Asian Conference on Industrial Automation and Robotics* (ACIAR 05). 11th – 13th May, 2005. Bangkok, Thailand.
- Nyan Bo Bo, Matthew N. Dailey and Bunyarit Uyyanonvara. Robust Hand Tracking in Low-resolution Video Sequences. In *Proceedings of the 3rd International Conference on Advances in Computer Science and Technology* (ACST 2007). 2nd – 4th April, 2007. Phuket, Thailand.