論文 / 著書情報
Article / Book Information

題目(和文)	ジェスチャーコミュニケーションにおけるAdaBoostに基づく手と顔の検出の新たな手法に関する研究
Title(English)	Human Parts Detection in Gesture Communication Using A Novel Method Based on AdaBoost
著者(和文)	ブシュクケイ
Author(English)	Shuqiong Wu
出典(和文)	学位:博士（工学）, 学位授与機関:東京工業大学, 報告番号:甲第9944号, 授与年月日:2015年6月30日, 学位の種別:課程博士, 審査員:長橋　宏,佐藤　誠,新田　克己,渡邊　澄夫,長谷川　修
Citation(English)	Degree:, Conferring organization: Tokyo Institute of Technology, Report number:甲第9944号, Conferred date:2015/6/30, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Human Parts Detection in Gesture Communication Using A Novel Method Based on AdaBoost

東京工業大学
Tokyo Institute of Technology

Shuqiong Wu

This dissertation is submitted for the degree of

Doctor of Engineering

March 2015

Abstract

With the rapid development of information technology, gesture communication systems become more and more popular in the last two decades. In gesture communication systems, detecting face and hands is particularly important. Compared with face detection, hand detection is far more difficult because hands are highly deformable.

There are many approaches for detecting hands. These approaches can be roughly divided into two categories. One is device-based and another is vision-based. Device-based methods detect hands based on the data obtained from some specific devices such as data gloves, sensors, Kinect, and so on. In this case, hands positions can be detected correctly. However, wearing specific devices may cause troublesome and uncomfortableness. Furthermore, processing massive data from these devices could lead to a large amount of calculation. By contrast, vision-based techniques process 2D images captured by cameras. They are more natural and suitable for real-time applications compared with device-based methods. Nevertheless, it is difficult to detect hands positions precisely by utilizing only 2D information.

Our research focuses on improving the accuracy of vision-based hand detection systems. In our work, we detect the face first, and then use the face information to improve the performance of hand detector. We train the face and hand detectors based on Haar-like features by AdaBoost, which is a machine learning method. To reduce the false positive rate and the number of hand training instances, we propose a novel technique called "background-masking" which applies the face information to the training of hand detector. In addition, we devise a new AdaBoost variant which we call "Penalized AdaBoost" to

improve the performance of the face and hand detectors. Experiments show that our detection system based on background-masking and Penalized AdaBoost not only achieves high accuracy, but also saves a lot training and detection time compared with other vision-based approaches.

To my parents, husband, and child ...

Acknowledgements

I would like to express my sincere gratitude to Professor Hiroshi Nagahashi, my supervisor for his kindly academic support, valuable comments and constructive guidance throughout my doctoral period. I could not achieve the great progress in both my research ability and writing technique without the support from my supervisor.

I also wish to thank Kota Aoki, our assistant professor for his kindly support and guidance during my doctoral study. Similarly, I owe my sincere gratitude to Dr. Chamidu Atupelage and Dr. Min Yao for their assist in my academic study. I also want to express my gratitude to all members in my lab for their sincere help.

I genuinely thank JASSO and Kobayashi Scholarship Foundations for their financial support during my doctoral life. Owing to their kindly support, I could focus all my energy on my research, and accomplish my objectives on schedule.

Last, I want to express my gratitude to my parents and husband for their enduring love and support. Thanks to their understanding and encouragement, I could put a great source of energy into my research and overcome any difficulty during my doctoral period.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Objective

Face and hands detection is crucial to gesture communication systems because they carry a lot important information that signers are conveying. Nowadays, vision-based methods are preferable since device-based approaches may lead to troublesome and tremendous calculation. Among many vision-based object detection techniques, Viola-Jones method has gained its popularity rapidly due to its high performance on both the speed and accuracy. It combines Haar-like features with AdaBoost. Haar-like features are proved robust against illumination and scale changes whereas AdaBoost is a machine learning method for classification. Viola-Jones method achieved considerable success in face detection [1]. However, it is difficult to apply this approach to hand detection due to the high deformability of hands. If we train one detector for one kind of hand shape, we can not handle many hand shapes in real time because many detectors are required simultaneously. If we train one detector for many hand shapes, the detection system can run in real-time. However, the false positive rate will be high due to the diversity of training instances.

To solve the above problem, we propose a novel hand detection system which trains one detector for multiple hand shapes. Our detection system combines a new AdaBoost variant (Penalized AdaBoost), a novel training technique (background-masking), and motion filtering together. The main objective of our research is accomplished by addressing the following three sub-goals.

- Decreasing the false positive rate of our trained hand detector.
- Reducing the number of hand training instances to speed up the training process of hand detector.
- Improving the classification performance of AdaBoost to increase the recall of our detection system.

1.2 Motivation

A gesture is a type of non-verbal communication language which consists of several visible body actions. Gestures usually include both face and hands movements. Differently from other physical body movements, gestures convey particular messages. A combination of hand trajectory, hand shape, face expression, the distance between face and hands represents various meaningful words and sentences. Gesture communication systems (also called gesture recognition systems) translate visible gestures into verbal language. Gesture communication systems provide a platform for an arbitrary communication between hearing impaired people and non-deaf people. Furthermore, they can be assistant tools for learning sign languages. For gesture communication systems, accuracy and speed are two crucial requirements.

Machine learning algorithms are widely used in gesture communication systems due to their rapidity and efficiency. Among these methods, HMM (Hidden Markov Model) is most popular because it builds a statistical model for temporal pattern recognition [2–4]. However, this technique requires a tremendous training data set [3]. For each sign language word which is a sequence of actions, many training instances are needed. Therefore, collecting training instances becomes a laborious and time-consuming task. To solve this problem, many researchers try to break down gestures into monotonous movements. Since many gestures share similar basic movements, recognition systems based on these basic movements require far less training instances than those based on gestures.

In our research, we want to break down monotonous movements into smaller parts such as hand shapes, hand positions, hand trajectories, relative distance between face and hands, and so on. Since a lot of monotonous movements share similar hand shapes and moving styles, our proposal can compress the training data set more than those movements based approaches. Our research is valuable because it can train a statistical gesture recognition model based on limited training data. Furthermore, our idea makes it possible to deal with a large vocabulary of sign language in real-time. Because our research is based on face and hand information, detecting face and hands position precisely and extracting face and hands patches in proper size are most important. In this dissertation, we focus on devising a novel face-hand detection system that can run in real-time with high accuracy.

1.3 Approach

As we discussed above, vision-based detection approaches are more natural and suitable for real-time applications. However, they are more difficult to achieve high accuracy than device-based methods especially in hand detection because hands are highly deformable. In this paper, we propose a novel vision-based face-hand detection system which has strong robustness against rotation, scale change, illumination change, and background change. Our detection system is based on Haar-like features and AdaBoost. Nevertheless, differently from Viola-Jones method [1], we propose background-masking [5], motion filtering [6] and Penalized AdaBoost [7] to improve the performance of our detection system.

As a general rule, training instances with diverse backgrounds are required to train a hand detector. However, using training instances in various backgrounds not only leads to a long training time, but also causes background noise which may degrade the performance of the trained hand detector. background-masking effectively solves this problem by a skin color segmentation technique. As we know, face detection is more advanced than hand detection because faces share similar patterns. Thus, we can detect the face first, and then use the face information to do a skin color segmentation. background-masking does the skin color segmentation for both training instances and test images. Thereby, it can train the hand detector based on training instances in only one background.

Motion filtering aims at improving the performance of background-masking. Motion filtering combined with background-masking excludes non-skin-color parts and non-moving part efficiently.

Penalized AdaBoost is proposed to improve the performance of our face and hand detectors. It is a variant of AdaBoost which utilizes the margin distribution to restrain the misclassification of small-margin instances. Moreover, it introduces an adaptive reweighting technique which can reduce the influence from noise-like instances. Our face-hand detection system is built by the following steps:

(1) To train the face detector, first we extract Haar-like features from face training instances. Then we utilize Penalized AdaBoost to train the face classifiers. Finally we combine these trained classifiers into one face detector.

(2) To train the hand detector, first we extract Haar-like features from the background-masked hand instances. Then we train hand classifiers by Penalized AdaBoost. Finally we combine the trained classifiers into one hand detector.

(3) First we utilize background-masking and motion filtering to process the test images. Then we use the trained face and hand detectors to detect face and hands in the processed images.

1.4 Contributions

The novel contributions of our research are listed as follow:

- We proposed background-masking which introduces a new training style for machine learning methods. It utilizes the background-masked hand instances instead of 2D hand images. background-masking removes the background part of training instances. Thus, it can avoid the influence of background noise. On the other hand, instead of using training instances in diverse backgrounds, we can create the background-masked instances from hand images in only one background. Therefore, background-masking decreases the number of training instances effectively [5].

- We proposed Penalized AdaBoost to improve the classification performance of our detection system. It penalizes the misclassification of small-margin instances in the current loop by analyzing the margin distribution in the previous loop. Furthermore, it reinitialize the weights of noise-like instances to reduce their influence on the training. Our experiments show that Penalized AdaBoost is more robust than other AdaBoost variants. As a new machine learning approach, Penalized AdaBoost not only achieves high performance in our detection system, it can also be applied to improve the classification performance of other systems [7].

- We proposed a novel face-hand detection system which is a combination of Haar-like features, background-masking, motion filtering, and Penalized AdaBoost. Our experiments show that this detection system can run in real-time, and is robust to scale change, illumination change, background change, and rotation [6].

- We also proposed another AdaBoost variant which we call Parameterized AdaBoost. Although it is not related to our detection system, it proves a new theory that the decreasing of training error can be speed up by reducing the sum of sample weights explicitly [8].

- We analyzed the generalization abilities of many AdaBoost variants. Differently from other comparison studies, our work compare the generalization abilities of different AdaBoost variants by analyzing the weak hypotheses and margin distributions. Our comparison shows statistical proof to explain why this variant is better than that one. Our study is useful for researchers who want to improve the performance of their applications by switching to another AdaBoost variant [9].

In general, our research not only contributes to face and hand detection in gesture communication systems, but also contributes to the field of artificial intelligence.

1.5 Outline

This dissertation includes 8 chapters. This first chapter is the introduction. Chapter 2 describes the basic ideas related to our research, including the introduction of different AdaBoost variants, the meaning of face and hand detection in gesture communication systems, previous research in vision-based face detection, and previous research in vision-based hand detection. Chapter 3 explains our novel AdaBoost variant: Penalized AdaBoost. Also it shows the statistical proof that Penalized AdaBoost is more robust than other variants with respect to the generalization errors. Chapter 4 describes our proposed face-hand detection system. In this chapter, background-masking and motion filtering are introduced in great detail. Chapter 5 presents our another AdaBoost variant called Parameterized AdaBoost, and shows how Parameterized AdaBoost speeds up the training process. Chapter 6 shows experimental results with respect to Penalized AdaBoost, our face-hand detection system, and Parameterized AdaBoost. Chapter 7 is a comparison study of different AdaBoost variants and Chapter 8 concludes our entire research with values, limitations, and future work.

Chapter 2

Related Work

2.1 Training instances and weak classifiers

In this section, we briefly describe the training instances and weak classifiers used in our research. We suppose that S =$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ is a training set, where N is the number of training instances. We set y_i to be 1 if x_i is positive, otherwise, we set y_i to be -1. In this dissertation, we focus on solving binary classification problems. In our research, CART (Classification and Regression Tree) is utilized as weak classifiers. Figure 2.1 shows an example of CART. It is a decision tree whose leaves produce the classification results and inner nodes split the tree to minimize the error rate. In Fig.2.1, the feature vector of instance x_i is represented by $X_i = [x_{i,1}, x_{i,2}, ..., x_{i,m}]$.

2.2 AdaBoost and its variants

AdaBoost is a machine learning method for binary classification. At each iteration, it increases the weights of misclassified instances and decreases the weights of correctly classified instances. This type of weight adjustment is most important because AdaBoost can emphasize more on difficult-to-classify instances [10]. As the popularity of AdaBoost increases, many different variants were proposed for diverse purposes. In recent years, AdaBoost and its variant are widely utilized in data classification and object detection.

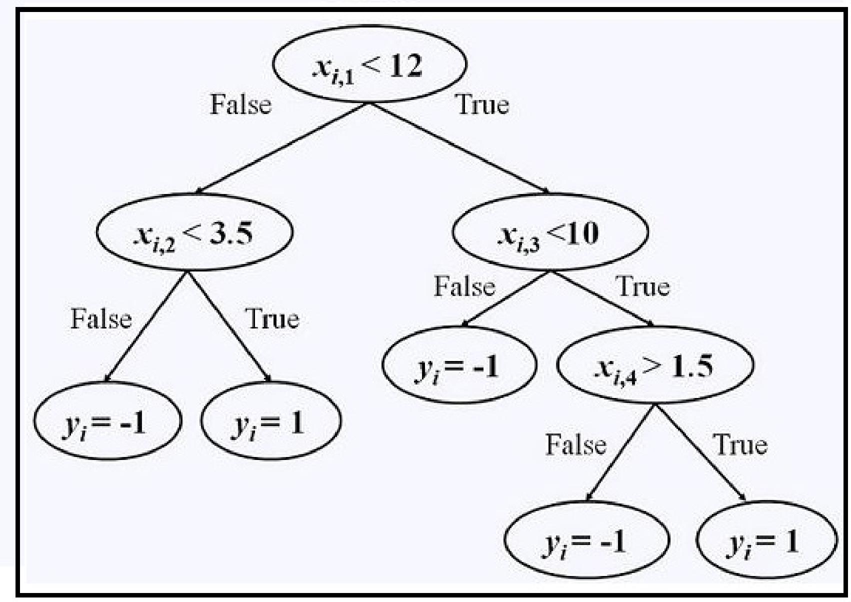


Figure 2.1: An example of CART

2.2.1 Real AdaBoost

The weak classifiers in AdaBoost (also called Discrete AdaBoost) output +1 or −1 to predict the class of instances. To analyze this method mathematically, a generalized version called Real AdaBoost is proposed by Schapire and Singer [11]. Differently from AdaBoost, each weak hypothesis in Real AdaBoost produces real numbers. The sign of the weak hypothesis predicts the class of the weighted majority of instances falls into the same partition divided by CART, and the absolute value of the weak hypothesis shows a predication confidence. Real AdaBoost is explained as follows:

Real AdaBoost

1. Assign the initial weights $w_{i,1} = 1/N$, where $i = 1, 2, ..., N$.

2. Repeat the following tasks for $t = 1, 2, ..., T$:

(a) Train a CART to classify the training set S into L partitions. Each leaf of the CART represents one partition. For each partition S_t^j where $j \in \{1, 2, ..., L\}$, compute W_{t+}^j and W_{t-}^j as follows:

$$W_{t+}^j = \sum\nolimits_{i:x_i \in S_t^j \wedge y_i = 1} w_{i,t}. \tag{2.1}$$

$$W_{t-}^{j} = \sum\nolimits_{i:x_i \in S_t^j \wedge y_i = -1} w_{i,t}. \tag{2.2}$$

(b) Compute the weak hypothesis for every partition S_t^j as

$$f_t^j(x) = 1/2 \times ln(W_{t+}^j / W_{t-}^j). \tag{2.3}$$

For each training instance x_i, its weak hypothesis equals to $f_t^p(x)$, where p is the index of partition which x_i falls into.
(c) Update the weights of all instances by

$$w_{i,t+1} = w_{i,t} exp[-y_i f_t(x_i)]/G_t. \tag{2.4}$$

$$G_t = \sum\nolimits_i w_{i,t} exp[-y_i f_t(x_i)]. \tag{2.5}$$

3. Set $F_T(x_i) = \sum_{t=1}^{T} f_t(x_i)$, and then output the strong classifier $H(x_i) = sign[F_T(x_i)]$.

2.2.2 Gentle AdaBoost

In order to improve the classification performance of AdaBoost, Friedman et al. analyzed the relationship between additive logistic models and AdaBoost, and then proposed Gentle AdaBoost that calculates weak hypotheses by optimizing the weighted least square errors [12]. Since Gentle AdaBoost tries to reduce the variance of its weak hypotheses, it is more robust and stable than AdaBoost and Real AdaBoost [12]. Next we describe Gentle AdaBoost as follows:

Gentle AdaBoost

1. Assign the initial weights $w_{i,1} = 1/N$, where $i = 1, 2, ..., N$.
2. Repeat the following tasks for $t = 1, 2, ..., T$:
(a) Train a CART, and then compute W_{t+}^j and W_{t-}^j for every partition S_t^j the same as in Step 2(a) of Real AdaBoost.

(b) Calculate the weak hypothesis for each partition S_t^j by

$$f_t^j(x) = [(W_{t+}^j - W_{t-}^j)/(W_{t+}^j + W_{t-}^j)]. \tag{2.6}$$

For any training instance x_i, Set its weak hypothesis to be $f_t^p(x)$. Here p is the index of partition which x_i belongs to.

(c) Update the weights for all instances by (2.4) and (2.5).

3. Set $F_T(x_i) = \sum_{t=1}^{T} f_t(x_i)$, and then output the strong classifier $H(x_i) = sign[F_T(x_i)]$.

2.2.3 Modest AdaBoost

In order to reduce the generalization error of Gentle AdaBoost, Vezhnevets and Vezhnevets introduced Modest AdaBoost which utilizes an "inverted" distribution to increase the contribution of weak hypotheses that work well on instances with large weights [13]. Modest AdaBoost obtains better generalization errors than Gentle AdaBoost in some data sets [14]. Nevertheless, it performs worse than Gentle AdaBoost occasionally with respect to the generalization errors. Modest AdaBoost is introduced by the following algorithm.

Modest AdaBoost

1. Assign the initial weights $w_{i,1} = 1/N$, where $i = 1, 2, ..., N$.

2. Repeat the following tasks for $t = 1, 2, ..., T$:

(a) Train a CART, and then compute W_{t+}^j and W_{t-}^j for every partition S_t^j the same as in Step 2(a) of Real AdaBoost.

(b) Compute an inverted weight $\overline{w}_{i,t}$ as

$$\overline{w}_{i,t} = (1 - w_{i,t})/U_t. \tag{2.7}$$

Here $U_t = \sum_i (1 - w_{i,t})$. Next calculate $\overline{W}_{t+}^j$ and $\overline{W}_{t-}^j$ for every partition S_t^j by

$$\overline{W}_{t+}^j = \sum\nolimits_{i:x_i \in S_t^j \wedge y_i = 1} \overline{w}_{i,t}. \tag{2.8}$$

$$\overline{W}^j_{t-} = \sum\nolimits_{i:x_i \in S^j_t \wedge y_i = -1} \overline{w}_{i,t}. \tag{2.9}$$

(c) Calculate the weak hypothesis for each partition S^j_t by

$$f^j_t(x) = [W^j_{t+}(1 - \overline{W}^j_{t+}) - W^j_{t-}(1 - \overline{W}^j_{t-})]. \tag{2.10}$$

For any instance x_i, its weak hypothesis is $f^p_t(x)$, where p is the index of partition which x_i belongs to.
(c) Update the weights for all instances by (2.4) and (2.5).
3. Set $F_T(x_i) = \sum_{t=1}^{T} f_t(x_i)$, and then output the strong classifier $H(x_i) = sign[F_T(x_i)]$.

2.2.4 Other AdaBoost variants

Except Real, Gentle, and Modest AdaBoost, there are many other variants proposed recently. For example, BrownBoost was proposed to reduce the influence of outliers in training [15]. LPBoost was created to optimize the minimal margin of training instances by utilizing linear programming [16]. Unfortunately, a comparison study showed that LPBoost overall performs worse than AdaBoost [17]. MadaBoost and SmoothBoost were proposed to improve the robustness against malicious noise data [18, 19]. Other AdaBoost variants such as AdaC1, AdaC2, AdaC3, AdaCost, CSB0, CSB1, CSB2, and RareBoost assign weights to positive training data and negative training data differently to achieve a better performance on imbalanced data sets [20–23]. There are also some variants which trade off the integrity of the training set for faster training [24–27]. SoftBoost, Soft-LPBoost, ReweightBoost, Interactive Boosting, and RobustBoost were proposed to suppress the generalization ability of AdaBoost [28–33]. SoftBoost optimizes a soft margin instead of the hard margin used in AdaBoost [28, 29]. While Soft-LPBoost is a combination of SoftBoost and LPBoost [30]. ReweightBoost establishes a tree structure by reusing the selected weak classifiers. However, it can not use other structures as its weak classifiers except stump decision trees [31]. Interactive Boosting assigns weights to both features and training data [32].

RobustBoost is an improvement of BrownBoost [33]. The above five AdaBoost variants actually achieve better generalization errors than AdaBoost. Nevertheless, they increase the amount of complicated calculation, which may cause a longer training time. AdaTree is another AdaBoost variant which aims at speeding up the training process. It selects weak classifiers the same as in AdaBoost, but combines them non-linearly [34]. FM-AdaBoost and FloatBoost delete the less effective weak classifiers so that they outperform AdaBoost if they have the same number of weak classifiers as AdaBoost [35, 36]. However, they require more training cycles compared with AdaBoost. On the other hand, Filterboost and Regularized AdaBoost were introduced to solve overfitting problems [37, 38]. Filterboost utilizes a new logistic regression technique whereas Regularized AdaBoost needs validation subsets to identify and regulate the overfitting iteratively. In the last decade, A novel boosting algorithm called SemiBoost has been developed extensively. It integrates supervised learning with semi-supervised learning by utilizing both labelled and unlabelled training data [39]. Its purpose is to improve the robustness when the labelled training data are not sufficient [40]. Besides the above discussed variants, Conservative.2 AdaBoost, AdaBoost.M1, and Aggressive AdaBoost were proposed to handle multiple classification problems [41].

2.2.5 Margins in AdaBoost variants

In AdaBoost and its variants, the classification margin of an instance x_i is regarded as the difference between prediction confidence of weak hypotheses producing correct classification and that of weak hypotheses causing misclassification [42]. It is in the range $[-1, 1]$. The instance x_i is correctly classified if and only if its margin is positive [42]. The margin of an instance x_i in AdaBoost is defined by [42]:

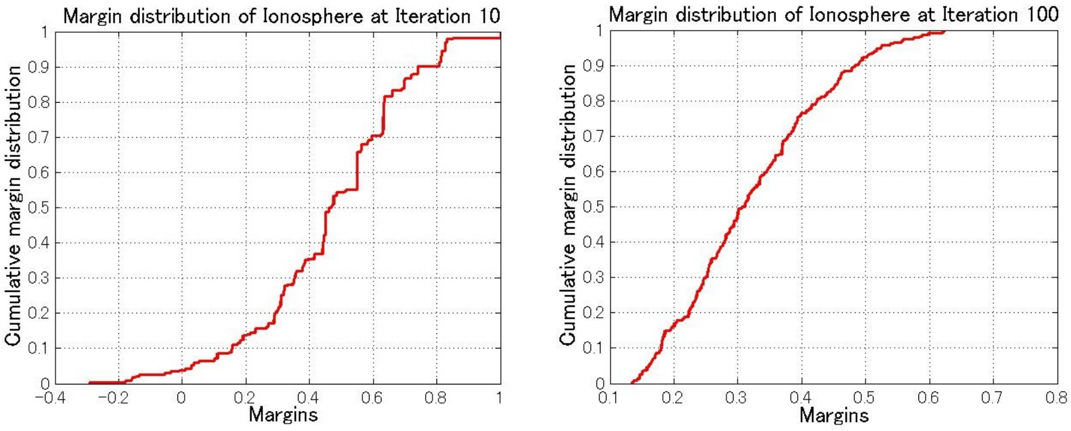
$$\text{Margin}_T(x_i) = \frac{y_i \sum_1^T \alpha_t h_t(x_i)}{\sum_1^T \alpha_t}. \tag{2.11}$$

Here $\sum_1^T \alpha_t h_t(x_i)$ is the final strong classifier, and $h_t(x_i)$ is the weak classifier at t-th iteration.

Differently from AdaBoost, Most variants such as Real, Gentle, and Modest AdaBoost fold the parameter α_t into the weak hypothesis. For example, for an instance x_i, the value of its weak hypothesis at the t-th iteration $f_t(x_i)$ is a real number whose sign represents the class of the weighted majority of training data in the partition x_i belongs to, and the absolute value $|f_t(x_i)|$ shows a prediction confidence. The larger its prediction confidence is, the higher classification ability the weak hypothesis has. Then we can deduce the following equation by utilizing weak hypotheses with prediction confidence [7].

$$\text{Margin}_T(x_i) = \frac{y_i \sum_1^T f_t(x_i)}{\sum_1^T |f_t(x_i)|}, \tag{2.12}$$

where $f_t(x_i)$ denotes the weak hypothesis of instance x_i at t-th iteration. In (2.12), T is the number of iterations, and the weak hypothesis $f_t(x_i)$ is actually a function from instance space X to real numbers. Figure 2.2 shows the cumulative margin distributions of Real AdaBoost in a data set Ionosphere at Iteration 10 and 100. From Fig. 2.2, we can see that the margins of more and more instances become positive as the number of iterations increases. This means more and more instances are correctly classified when the number of weak classifiers is increased.



(a) Margin distribution at Iteration 10

(b) Margin distribution at Iteration 100

Figure 2.2: Cumulative margin distributions of Real AdaBoost

2.3 Face detection in gesture communication

In a gesture communication system, face also carries some particular messages. In most cases, the relative distance between hands and face plays an important role in sign language recognition. Moreover, face expression is also important to some kinds of sign languages like Japanese Sign Language. To obtain the face information correctly, first we need to detect the face position and extract the face patch. Face detection is more advanced than hand detection since faces of different people share a similar pattern as Fig. 2.3 shows. This pattern includes two parts, one is that the eye areas are always darker than nose area. Another is the eye areas are always darker than the area below two eyes. There are many features can be utilized to recognize a face, such as SIFT (Scale invariant feature transform), HOG (Histogram oriented gradients), LBP (Local binary patterns), Haar-like features and so on. On the other hand, a lot of algorithms are also applied to face detection, such as SVM (Support vector machine), PCA(Principal Component Analysis), Random forest, AdaBoost, GMM, and so on. Among these features and algorithms, Viola-Jones method which combines Haar-like features with AdaBoost achieved high accuracy in face detection [1]. This method also computationally outperforms other approaches that are based on SIFT or HOG. In recent years, many researchers improved Viola-Jones face detector by various techniques [43, 44].

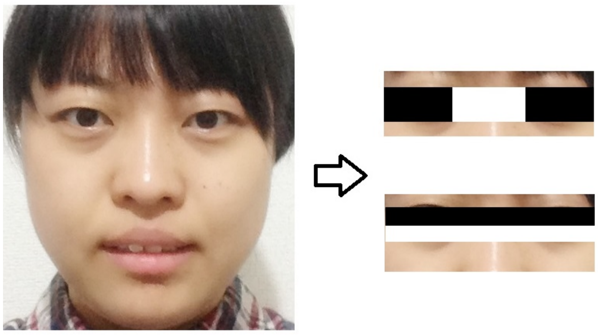


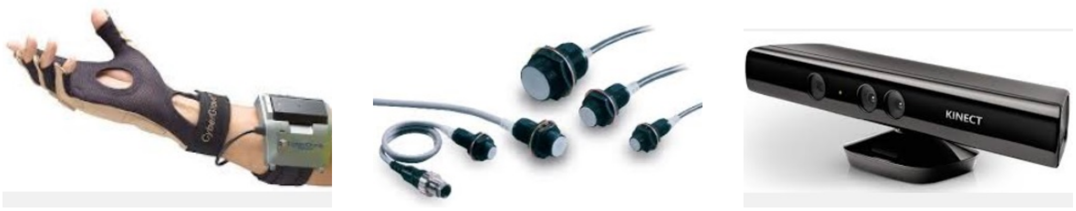
Figure 2.3: Face pattern shared by different faces

2.4 Hand detection in gesture communication

In this section, first we compare device-based hand detection with vision-based hand detection. Then we discuss the features and algorithms currently used in vision-based hand detection.

2.4.1 Device-based vs vision-based hand detection

Hand detection approaches can be divided into two categories according to the way how the hand information is collected. One category is device-based, and another is vision-based. Device-based methods collect the hand information via specific devices like data glove, sensor, and kinect. Data glove shown in Fig. 2.4 (a) can not only locate the positions of hands, but also the positions of fingers. It is easy to get the detail information of hands with data gloves [45]. However, wearing data gloves may cause uncomfortableness or troublesome. Researchers also use sensors as shown in Fig. 2.4 (b) to transmit the information of hands [46]. Unfortunately, sensors suffer from the same problem as data gloves does. Furthermore, selecting useful data from the information transmitted by data gloves or sensors may be also a tough task in sign language recognition. Nowadays, A novel device called Kinect as shown in Fig. 2.4 (c) becomes more and more popular. It builds the 3D human body first, and then precisely locates each human part such as face, hand, arm, and so on [47–49]. However, Kinect has a distance limit and requires a large amount of computation.



(a) Data glove (b) Sensor (c) Kinect

Figure 2.4: Devices for collecting data from hands

Differently from device-based approaches, vision-based methods use 2D images captured by cameras. Compared with multidimensional data captured by specific devices, 2D information is easier to be processed. Furthermore, vision-based methods do not have the problem caused by wearing. Nevertheless, vision-based methods detect the hand positions more difficultly compared with device-based techniques.

2.4.2 Vision-based hand detection approaches

In vision-based hand detection, many features and techniques are applied to recognize hands. Among these features, skin color is widely used to show the potential area of hands. A technique called skin-color segmentation was often conducted to deflate the range for hand detection [50]. In recent years, researchers prefer to combine skin-color segmentation with other algorithms to improve the robustness of hand detection systems. Skin-color segmentation was combined with optical flow [51]. While Wen and Zhan combined skin-color segmentation with Camshift [52, 53]. Camshift is an object tracking algorithm based on histograms of color components. Francke et al. applied GMMs (Guassian Mixture Models) to the skin-color segmentation, and then they utilized Meanshift to track hands [54].

Besides skin color, motion information is also used in vision-based hand detection. Cisneros and Rodriguez devised a detection system which is based on frame difference [55]. While Alsoos and Joukhadar combined skin-color segmentation with Motion History Images (MHI) [56]. A novel hand detection system which integrates skin-color segmentation, frame difference, and gradient tracking was proposed in [57].

Contours and shapes are also discriminant features for vision-based hand detection systems. Ong and Rowden utilized Shape-context to detect hands. Their system achieved a 99.8% detection rate in videos with simple background [58]. Other researchers combined skin-color segmentation with hand shape matching [59, 60]. A new approach which integrates skin-color segmentation, hand shape detection, and context recognition together to detect hands was proposed in [61].

Actually, motion, gradient, skin-color, context, shape, and contour are very important features for detecting hands. However, they are not robust sufficiently against background change, illumination change, and rotation. To devise hand detection systems with high accuracy, many researchers applied more robust features such as SIFT and HOG. For example, Wang and Wang combined SIFT with AdaBoost in [62]; it performed very well in hand detection. HOG-based algorithms were also introduced to hand detection systems [63–65]. A novel system combined HOG with LTP (Local trinary patterns) features whereas another one integrated HOG with LBP features [66, 67]. HOG and SIFT features were proved more robust than others like skin color, gradient, and so on. However, processing these features is computationally expensive.

Statistical models were also applied to detect hands [68, 69]. Nevertheless, they can only deal with limited hand shapes. If we apply this technique to handle many hand shapes, it is difficult to implement it in real time because many hand models should be loaded simultaneously. Some researchers used wrist or finger detection to locate hand positions [70–72]. Unfortunately, they have to combine other techniques to extract the hand patches in proper size. A pixel-level hand detection approach utilized 200 million labelled hand pixels to train the hand detector [73]. This method obtained strong robustness to rotation and illumination change. However, its ability of classifying hands from other skin color parts is unclear because videos contain faces or arms were not tested in the experiments. It is showed that approaches based on robust features and machine learning algorithms are more suitable for detecting multivariate hands [74, 75]. Recently, Viola-Jones method which is based on Haar-like features and AdaBoost was introduced to face detection [1]. As the popularity of Viola-Jones method increased, many improvements were also proposed. For example, Mustafa et al. improved Viola-Jones method by utilizing Gentle AdaBoost instead of AdaBoost [76].

Viola-Jones method was first applied to hand detection by Kolsch and Turk. They trained 8 hand detectors to recognize 8 hand shapes [77]. Kolsch and Turk proved that their approach is robust enough to background and illumination changes, but not robust against rotation [78]. To detect rotated hands, Barczak and Dadgostar utilized rotated hand instances for training the detectors [79]. In addition, a hybrid method which combines Haar-like features with skin-color segmentation was also introduced to detect hands [80]. For reducing the detection area of Viola-Jones hand detector, Canalis et al. utilized face detection to predict the potential positions of hands [81]. On the other hand, Guo et al. utilized pixel-based hierarchical features instead of Haar-like features in hand detection to improve the accuracy [82].

Viola-Jones method is proved suitable for real-time hand detection systems. However, it is confined to detect limited hand shapes since each hand shape needs one trained detector and many instances for training. To solve this problem, Wu and Nagahashi utilized training instances in multiple hand shapes to train one detector [83]. However, it needs a lot of training time, and causes a high false positive rate. To decrease the false positive rate, Wu and Nagahashi proposed a new preprocessing technique that uses small hand parts instead of the whole hand images as training instances [84]. Unfortunately, it requires test images in high resolution and other approaches to bound the hand area in proper size.

In this dissertation, we also utilize training instances in multiple hand shapes to train one hand detector. However, we propose background-masking method to reduce the false positive rate and training time. Furthermore, we devise Penalized AdaBoost to replace AdaBoost for a better classification performance. In next chapter, we will explain Penalized AdaBoost in details.

Chapter 3

Proposed Penalized AdaBoost

3.1 Basic knowledge: Gentle AdaBoost

Since Penalized AdaBoost is based on Gentle AdaBoost, we briefly review the basic idea of Gentle AdaBoost in this section. Gentle AdaBoost is an AdaBoost variant which calculates its weak hypothesis by minimizing the weighted least square error in each iteration. In this dissertation, we use CART as weak classifiers. Nevertheless, the weak hypotheses are not identically CART. Next we will introduce the weak hypotheses in Gentle AdaBoost in details.

3.1.1 Weak hypotheses

The weak hypotheses of Gentle AdaBoost are calculated by (2.6). From (2.6), we can see that the value of the weak hypothesis of each partition shows a trend of the prior probability. For example, we suppose $f_t^1(x)$ is the weak hypothesis of partition S_t^1, then we can see that $W_{t+}^1/(W_{t+}^1 + W_{t-}^1)$ is the weighted prior probability that an instance in partition S_t^1 is positive. On the other hand, $W_{t-}^1/(W_{t+}^1 + W_{t-}^1)$ shows the weighted prior probability that an instance in partition S_t^1 is negative. Gentle AdaBoost utilizes the prior probability to do a prediction. Thus, it is more robust than AdaBoost.

3.1.2 Generalization error and margins

The margins denotes the classification margins of training instances. As the number of weak hypotheses increases, the margins of more and more training instances are increased to be positive. That is why the training error can be decreased gradually. The training error reaches to 0 if and only if the margins of all training instances become positive. Reducing the number of negative margins is crucial to the convergence of training error. However, there is no direct relationship between the margins of training instances and generalization errors which are usually measured by the classification errors on test sets. But still we can find a clue between the margins and generalization errors. Freund and Schapire calculated the upper bound of generalization error as [10]

$$U_t = Pr(Margin_t(x) \leq \theta) + O(\sqrt{\frac{d}{m\theta^2}}). \tag{3.1}$$

Where Pr means the empirical probability on the training instances, m is the size of training set, d denotes the VC-dimension of weak hypotheses space, and θ is a number larger than 0. For a given training set, m is fixed, and d is related to the number of iterations. If we want to optimize this upper bound in each iteration, we need choose a proper θ. Although there is no approach to optimize this upper bound [10], we can suppress $Pr(Margin_t(x) \leq \theta)$ for all values of θ by improving the margin distributions as we demonstrate below: Figure 3.1 shows two different margin distributions. From Fig. 3.1, we can see that $Pr(Margin_t(x) \leq \theta)$ in the blue curve is always smaller than that in the red curve for any $\theta > 0$. This means the method which produces the blue curve is better than that creates the red curve in terms of generalization errors. Fig. 3.1 shows a perfect example, in fact, it is difficult to improve the margin distribution for all values of θ. Nevertheless, we can improve the cumulative distribution for most values of θ. In this case, there is a large chance that the generalization error will be reduced according to the improvement of margin distribution.

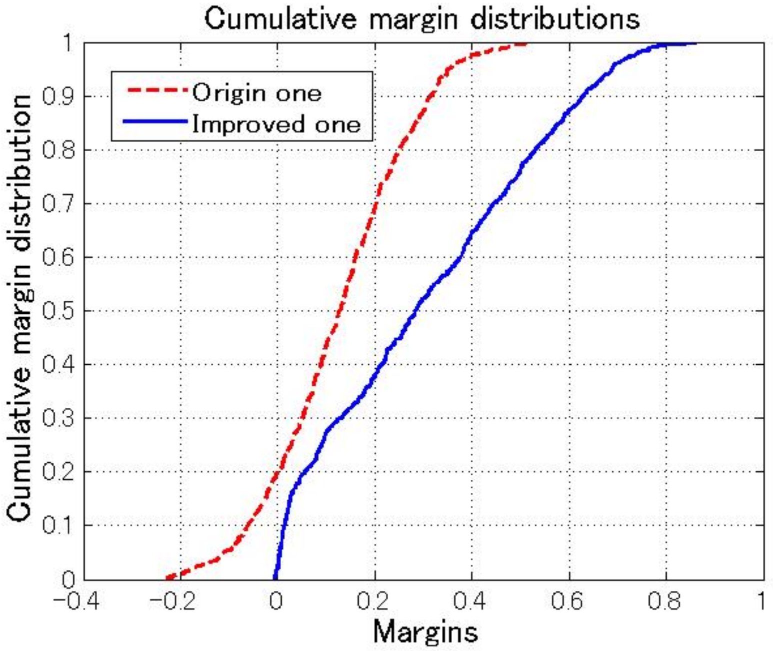


Figure 3.1: Comparison of cumulative margin distributions

3.1.3 Classifier distortion

In this section, we explain the problem of Gentle AdaBoost, and show why it need to be improved. As we know, Gentle AdaBoost increases the weight of instance x_i if it is misclassified by the current weak hypothesis. This kind of weight adjustment encourages Gentle AdaBoost to focus more on small-margin instances and try to correctly classify them in future iterations. Nevertheless, misclassification of small-margin instances could still happen in future loops especially when the data set contains some noise-like instances. If the misclassification occurs, the weights of these noise-like instances will be increased exponentially no matter how large or small their margins are. Accordingly, their margin will be decreased smaller. If these instances are misclassified many times, their weights will become larger and larger. By contrast, the weights of other instances will become smaller and smaller. Finally several large-weight instances will dominate the whole weight distribution to force Gentle AdaBoost to choose weak classifiers that can correctly classify them. Nevertheless, the late selected weak classifiers are more likely to fit these large-weight instances only. Thus the performance of the final strong hypothesis on other instances will be deteriorated. This phenomenon which is known as "classifier distortion", degrades the generalization error of the strong hypothesis because the deviation of the selected weak classifiers is increased in the late training phase as Gentle AdaBoost tries to fit large-weight instances. Therefore, mitigating the increase of large weight is necessary for improving the classification performance of the final strong hypothesis.

3.2 Margin-pruning Boost

In order to solve the classifier distortion discussed above, we proposed a new method which we call Margin-pruning Boost [85]. This algorithm filters instances with large weights, and then resets their weights to be 1. Next we explain Margin-pruning Boost as follows:

Margin-pruning Boost

1. Assign initial weights $w_{i,1} = 1/N$, where $i = 1, 2, ..., N$.
2. Repeat the following tasks for $t = 1, 2, ..., T$:
(a) Train a CART, and then calculate W_{t+}^{j} and W_{t-}^{j} for every partition S_t^j the same as in Step 2(a) of Real AdaBoost.
(b) Calculate the weak hypothesis for each partition S_t^j by (2.6).
(c) Update the weights of all instances as

$$w_{i,t+1} = exp[-y_i \sum_t f_t(x_i)]. \tag{3.2}$$

(d) The threshold R_{t+1} is set as follows:

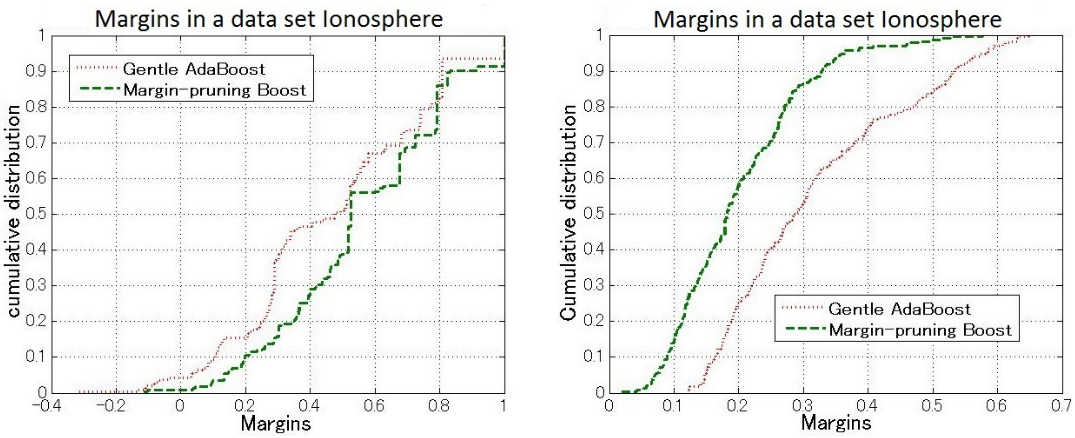
$$R_{t+1} = max_i\{w_{i,t+1}\} - \frac{max_i\{w_{i,t+1}\} - min_i\{w_{i,t+1}\}}{50}. \tag{3.3}$$

For each instance x_i, if $w_{i,t+1} > R_{t+1}$, reset $w_{i,t+1} = 1$ and $\sum_t f_t(x_i) = 0$. Then do normalization for all instances by setting $w_{i,t+1} = w_{i,t+1}/G_t$, where $G_t = \sum_i w_{i,t+1}$.
3. Set $F_T(x_i) = \sum_{t=1}^{T} f_t(x_i)$, and then output the strong classifier $H(x_i) = sign[F_T(x_i)]$.

In Step 2(d), we can see that Margin-pruning Boost resets the summed weak hypotheses and weights for instances whose weights are larger than the threshold at the current loop. This technique is effective at restraining the increase of large weight in the early training phase because most of instances filtered by the thresholding have a weight larger than 1. In this situation, the resetting actually reduces the influence of large-weight instances on the training. After resetting, the weights of these filtered instances become smaller than the original ones. Then in future iterations, we still calculate weak hypotheses for these instances. If their weights exceed the threshold in any future loop, they will be reset again. In Margin-pruning Boost, we do not exclude these large-weight instances completely since they are not necessarily noise data. In order to avoid the classifier distortion, we just keep the weights of these noise-like instances small by dynamically resetting them. Nevertheless, the weights of instances filtered by the thresholding are not larger than 1 necessarily as the number of iterations increases. If the filtered instances have a weight smaller than 1, resetting their weights to be 1 actually increases the influence of these instances.

On the other hand, from (2.12), we can see that the resetting initializes the margins of these filtered instances to 0. From this point of view, Margin-pruning Boost improves the margin distribution in the early training phase since most filtered instances have negative margins. Unfortunately, it does not work well as the number of loops increases because the margins of most filtered instances will become positive in the late training phase.

In general, Margin-pruning Boost succeeds in reducing the generalization error in the early training phase, but fails in the late training phase. We can also see this point from the following margin distributions in Fig. 3.2. To create the margin distributions, we use 2/3 of the data for training. Figs. 3.2 (a) and 3.2 (b) show the margin distributions of a data set Ionosphere at round 10 and 100 separately. From Fig. 3.2, we notice that Margin-pruning Boost performs better than Gentle AdaBoost at round 10. Nevertheless, the performance drops when the number of iterations reaches to 100. To solve the problem of Margin-pruning Boost, we propose another variant called Penalized AdaBoost, and then we will introduce it in the next section.



(a) Margin distribution at Iteration 10

(b) Margin distribution at Iteration 100

Figure 3.2: Cumulative margin distributions of the data set Ionosphere

3.3 Penalized AdaBoost: the algorithm

Penalized AdaBoost is an improvement of Margin-pruning Boost. It not only solves the problem of Margin-pruning Boost we discussed above, but also utilizes a margin feedback factor to penalize the misclassification of small-margin instances.

Penalized AdaBoost

1. Assign the initial weights $w_{i,1} = 1/N$, where $i = 1, 2, ..., N$.

2. Repeat the following tasks for $t = 1, 2, ..., T$:

(a) Train a CART, and then calculate W^j_{t+} and W^j_{t-} for each partition S^j_t the same as in Step 2(a) in Real AdaBoost.

(b) Compute a margin feedback factor $d_{i,t}$ as

$$d_{i,t} = \exp(-\text{Margin}_{t-1}(x_i))/D_t, \tag{3.4}$$

where D_t equals to $\sum_i \exp(-\text{Margin}_{t-1}(x_i))$. Then compute D^j_{t+} and D^j_{t-} of each partition S^j_t by

$$D^j_{t+} = \sum\nolimits_{i:x_i \in S^j_t \wedge y_i = 1} d_{i,t}. \tag{3.5}$$

$$D^j_{t-} = \sum\nolimits_{i:x_i \in S^j_t \wedge y_i = -1} d_{i,t}. \tag{3.6}$$

(c) Calculate the weak hypothesis for every S_t^j by

$$f_t^j(x) = \begin{cases} (W_{t+}^j - W_{t-}^j)(1 - D_{t-}^j), & if\ W_{t+}^j > W_{t-}^j \\ (W_{t+}^j - W_{t-}^j)(1 - D_{t+}^j), & otherwise \end{cases}. \tag{3.7}$$

For any instance x_i, set its weak hypothesis to be $f_t^p(x)$, where p is the index of partition which x_i falls into.
(d) Update the weights for all instances as

$$w_{i,t+1} = exp[-y_i \times \sum\nolimits_t f_t(x_i)]. \tag{3.8}$$

(e) For any instance x_i, if $w_{i,t+1} > R_{t+1}$ and $\text{Margin}_t(x_i) < 0$, reinitialize instance x_i as follows:

$$w_{i,t+1} = 1, \ \ \sum\nolimits_t f_t(x_i) = 0. \tag{3.9}$$

$$R_{t+1} = max_i\{w_{i,t+1}\} - \frac{max_i\{w_{i,t+1}\} - min_i\{w_{i,t+1}\}}{\beta}. \tag{3.10}$$

Then do the normalization by setting $\sum_i w_{i,t+1} = 1$.
3. Set $F_T(x) = \sum_{t=1}^{T} f_t(x)$, and then output the strong classifier $H(x) = sign[F_T(x)]$.

Penalized AdaBoost is introduced by the above algorithm. It computes the weak hypotheses by utilizing a margin feedback factor in Step 2(b) and 2(c). Furthermore, it improves the thresholding technique of Margin-pruning Boost in Step 2(e). Here we tune the parameter β in (22) as follows: first we evaluate the classification performances on 5 data sets with different values of β ($\beta = 10$, 30, 50, 70, 90), and then we choose the value with the best performance for β.

3.4 Weight increase limit in Penalized AdaBoost

In this section, we discuss how Penalized AdaBoost limits the weight increase of noise-like data comparing with Gentle AdaBoost and Margin-pruning Boost.

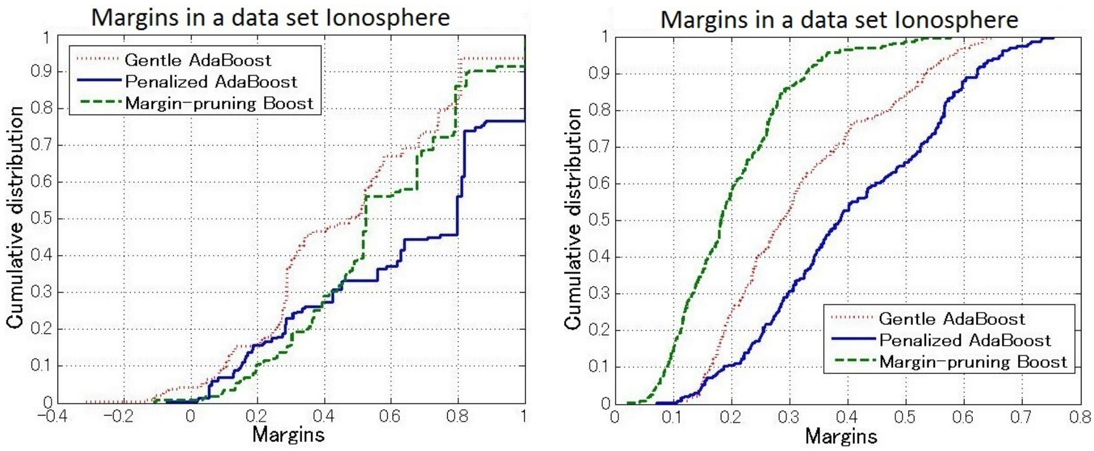
From Step 2(e) in Penalized AdaBoost, we can see that only instances whose margins are negative and whose weights exceed the threshold R_{t+1} are reset. This means that Penalized AdaBoost resets the weights of noise-like instances to be smaller under all circumstances. Compared with Gentle AdaBoost and Margin-pruning Boost, Penalized AdaBoost always keeps the weights of the noise-like instances small during the whole training. Therefore, it can reduce the bad influence from these noise-like data. Furthermore, differently from Margin-pruning Boost, Penalized AdaBoost tunes the parameter β by empirical studies.

With respect to the margins, Penalized AdaBoost only reinitializes the negative margins to 0 during the training. Thus, it can enlarge the margin distribution more than Gentle AdaBoost and Margin-pruning Boost especially when the number of loops increases.

In summary, the weight updating policy in Penalized AdaBoost is more efficient at suppressing the generalization errors. We can also see the same point from the cumulative margin distributions shown in Fig. 3.3. Figs. 3.3 (a) and 3.3 (b) show the margin distributions of the data set Ionosphere at iteration 10 and 100. In Fig. 3.3, the blue curves are results of Penalized AdaBoost. Figure 3.4 shows the generalization errors of the same data set. We notice that Penalized AdaBoost enlarges the margin distributions more than Gentle AdaBoost and Margin-pruning Boost as the number of loops increases. That can explain why it achieves the best generalization error in Fig. 3.4.

3.5 Penal policy in Penalized AdaBoost

In this section, we explain how Penalized AdaBoost restrains the misclassification of small-margin instances. Compared with Gentle AdaBoost and Margin-pruning Boost, it utilizes the margins in the previous loop to restrict the misclassification of small-margin instances in the current loop. In Step 2(c) of Penalized AdaBoost, we notice that D^j_{t+} and D^j_{t-} are calculated from the margin feedback factors of misclassified instances. We conclude that $(1 - D^j_{t+})$ and $(1 - D^j_{t-})$ are proportional to the margins according to (2.12), (3.5), (3.6) and (3.7). That means a misclassification of small-margin instances will result in small $(1 - D^j_{t+})$



(a) Margin distribution at Iteration 10 **(b)** Margin distribution at Iteration 100

Figure 3.3: Comparison of cumulative margin distributions on Ionosphere

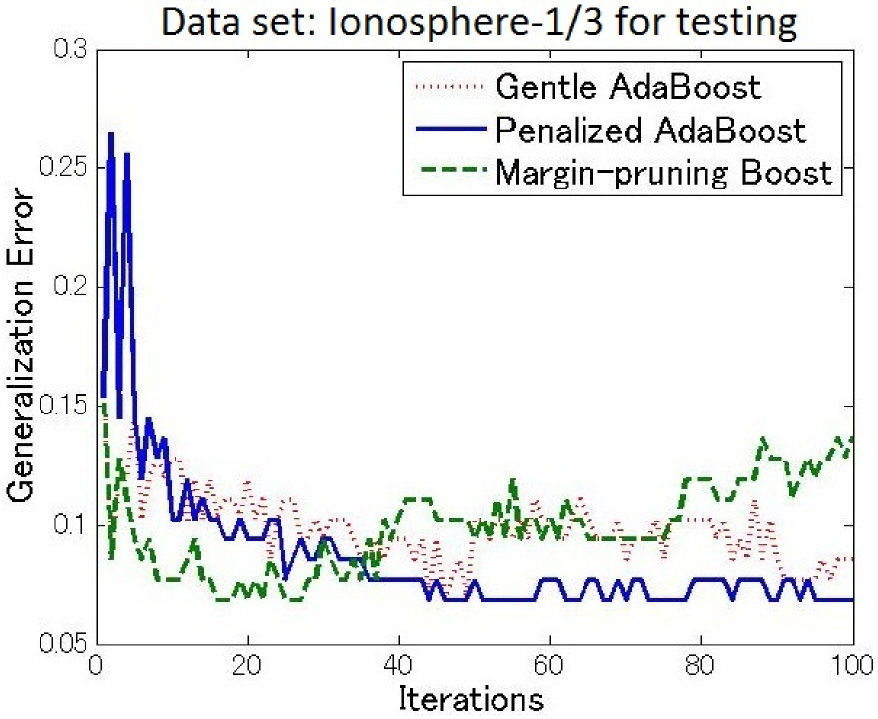


Figure 3.4: Comparison of generalization errors in data set Ionosphere

and $(1 - D^j_{t-})$. Then they will degrade the prediction confidence of the current weak hypothesis $f^j_t(x)$. Therefore, the misclassification of small-margin instances is easier to be corrected in Penalized AdaBoost, compared with the case without

$(1 - D^j_{t+})$ and $(1 - D^j_{t-})$ in Gentle AdaBoost and Margin-pruning Boost.

By contrast, a misclassification of instances with large margins will lead to large $(1-D^j_{t+})$ and $(1-D^j_{t-})$. Then the prediction confidence of the current weak hypothesis will be increased relatively. That means weak hypotheses misclassifying large-margin instances are more competent than those misclassifying small-margin instances. Although the weights of large-margin instances are increased due to the misclassification, they can not contribute to the classifier distortion because they were small in previous loops.

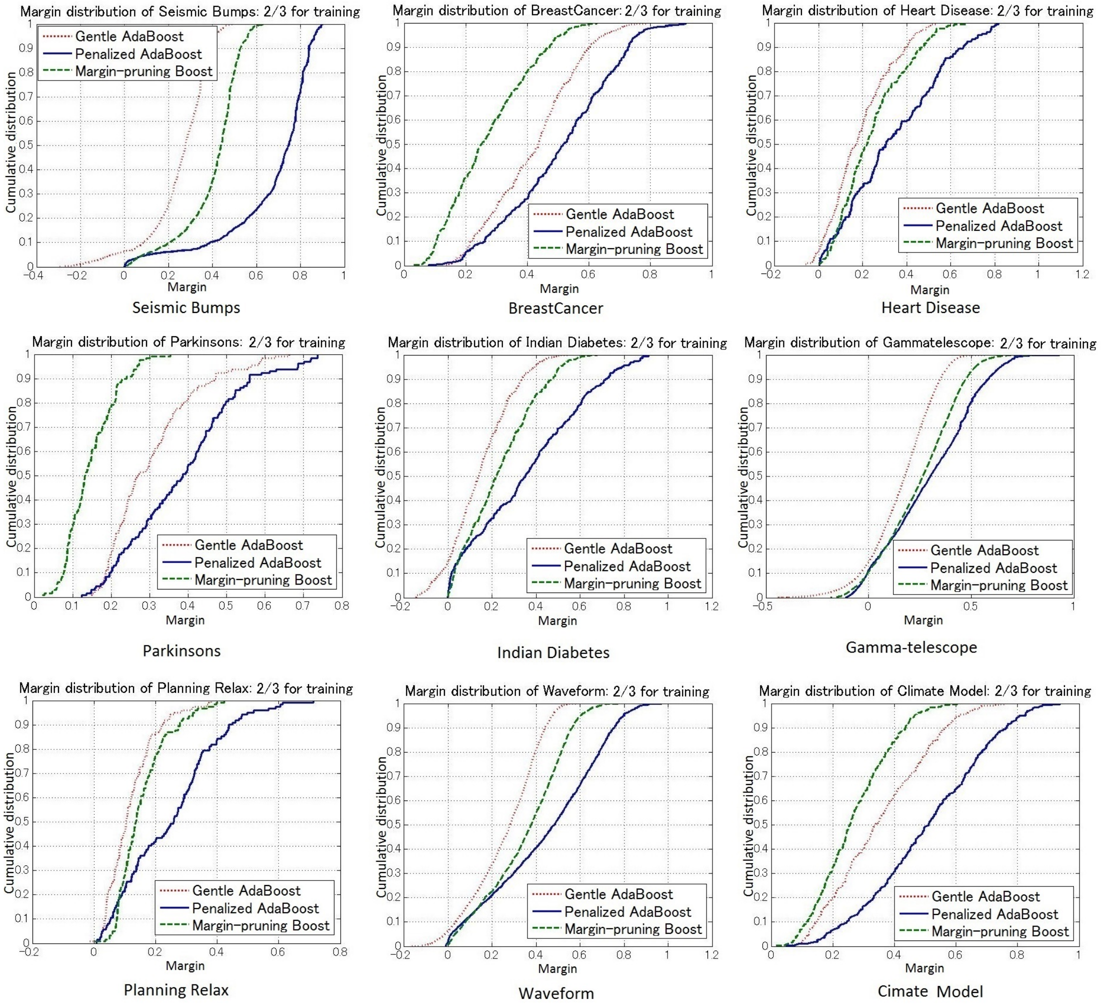
On the other hand, we know that the margin will be decreased if the misclassification occurs. However, in Penalized AdaBoost, the margin reduction of instances with small margins is mitigated by reducing the prediction confidence of the current weak hypothesis. Therefore, Penalized AdaBoost enlarges the margin distribution more than Gentle AdaBoost and Margin-pruning Boost due to the penal policy.

In general, both the weight updating and penal policy of Penalized AdaBoost can contribute to the improvement of margin distributions. Furthermore, they are effective at avoiding the classifier distortion. Therefore, Penalized AdaBoost is more robust than Gentle AdaBoost and Margin-pruning Boost.

3.6 Margins in Penalized AdaBoost

In this section, we compare the margin distributions of Gentle AdaBoost, Margin-pruning Boost, and Penalized AdaBoost in other data sets except Ionosphere. These margin distributions are computed by using 2/3 of the data in each data set as training instances.

Figure 3.5 shows the comparison results. In Fig. 3.5, the red, green, and blue curve show the results of Gentle AdaBoost, Margin-pruning Boost, and Penalized AdaBoost respectively. From Fig. 3.5, we can conclude that Penalized AdaBoost outperforms Gentle AdaBoost and Margin-pruning Boost with respect to the enlargement of margin distributions. Penalized AdaBoost also proves that reducing the importance of difficult-to classify instances properly actually is necessary for improving the generalization errors.



Seismic Bumps

BreastCancer

Heart Disease

Parkinsons

Indian Diabetes

Gamma-telescope

Planning Relax

Waveform

Cimate Model

Figure 3.5: Cumulative margin distributions

Chapter 4

Proposed Face and Hand Detection System

4.1 Viola-Jones Method

Nowadays, Viola-Jones method and its variants are widely utilized in gesture communication systems because of their high performance on both the speed and accuracy [86–89]. Viola-Jones detector is a combination of Haar-like features and AdaBoost classifier. Haar-like features can be computed from a set of rectangle masks as shown in Fig. 4.1. which include line features, edge features, and center-surrounded features. The value of each feature equals to the sum of pixel

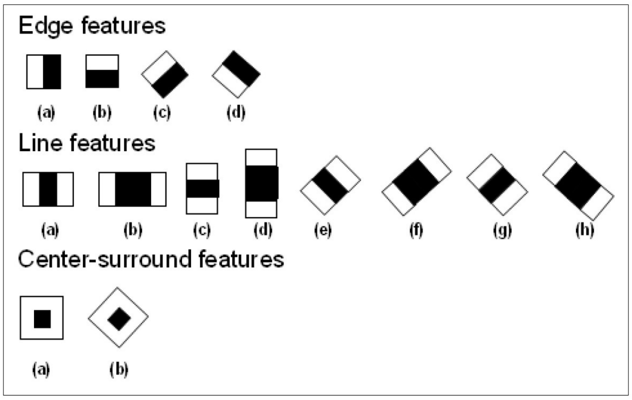


Figure 4.1: Haar-like features

intensities within the white rectangle subtracted by the sum of pixel intensities within the black rectangle.

For an instance x_i which is a 20×20 pixels sub-window of an image, its weak classifier $S_j(x_i)$ is defined by the following equation with parameters f_j, θ_j and p_j. Here f_j is a Haar-like feature, θ_j is a threshold and p_j is a sign which belongs to {+1, -1}.

$$S_j(x_i) = \begin{cases} 1, & if P_j f_j(x_i) \leq P_j \theta_j \\ 0, & otherwise \end{cases}. \tag{4.1}$$

In Viola-Jones method, each Haar-like feature can create one weak classifier. Then these weak classifiers will be integrated into one strong classifier by AdaBoost or its variants. In addition, Viola-Jones method improves the precision and detection speed by a cascading scheme. The framework of Viola-Jones method is shown in Fig. 4.2. From Fig. 4.2, we can see that only the instances

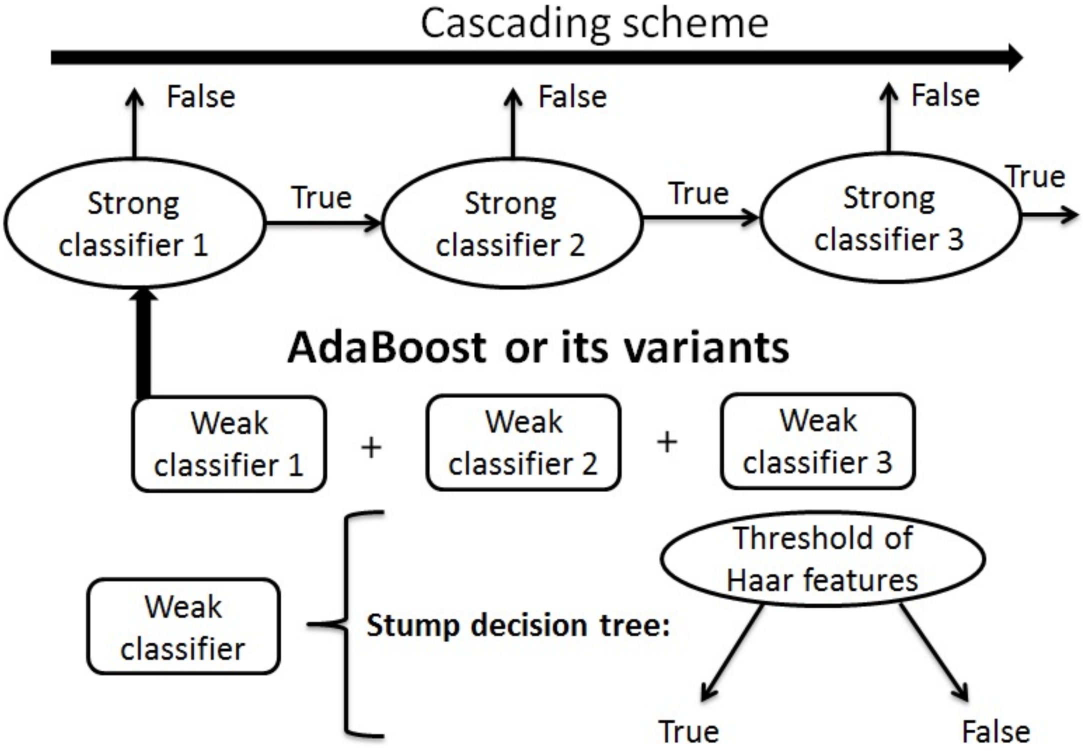


Figure 4.2: Framework of Viola-Jones method

which pass the previous classifiers can be processed by the next classifier. This kind of cascading structure reduces the false positive rate of the final detector effectively.

4.2 Face detection based on Penalized AdaBoost

Although Viola-Jones method has already achieved considerable success in face detection, we can still improves its performance by utilizing our new proposed Penalized AdaBoost.

4.2.1 Training data collection

With respect to the training data in face detection, we uses face images from MIT data set as positive instances, and images without any face as negative instances. Figures 4.3 and 4.4 show the positive examples and negative examples respectively.

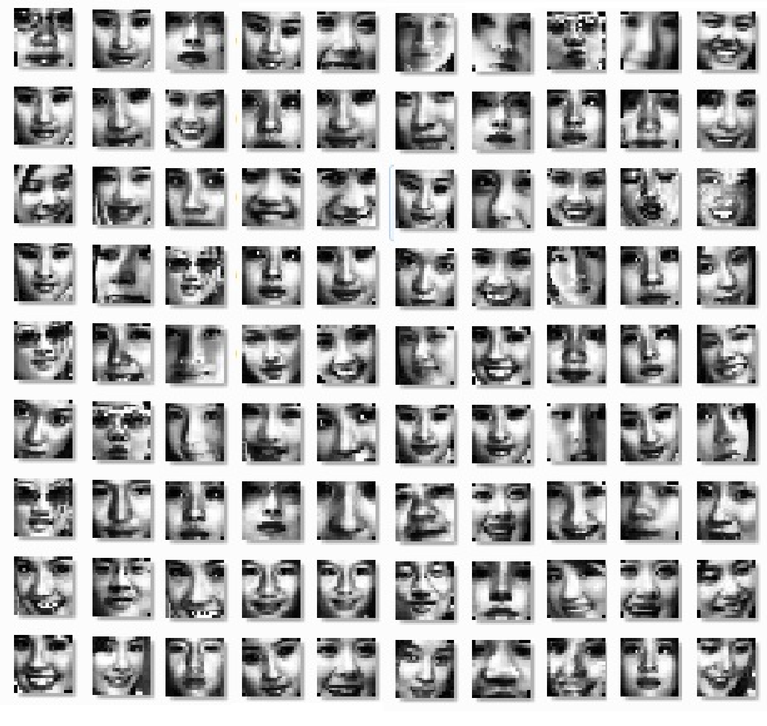


Figure 4.3: Positive training instances in face detection

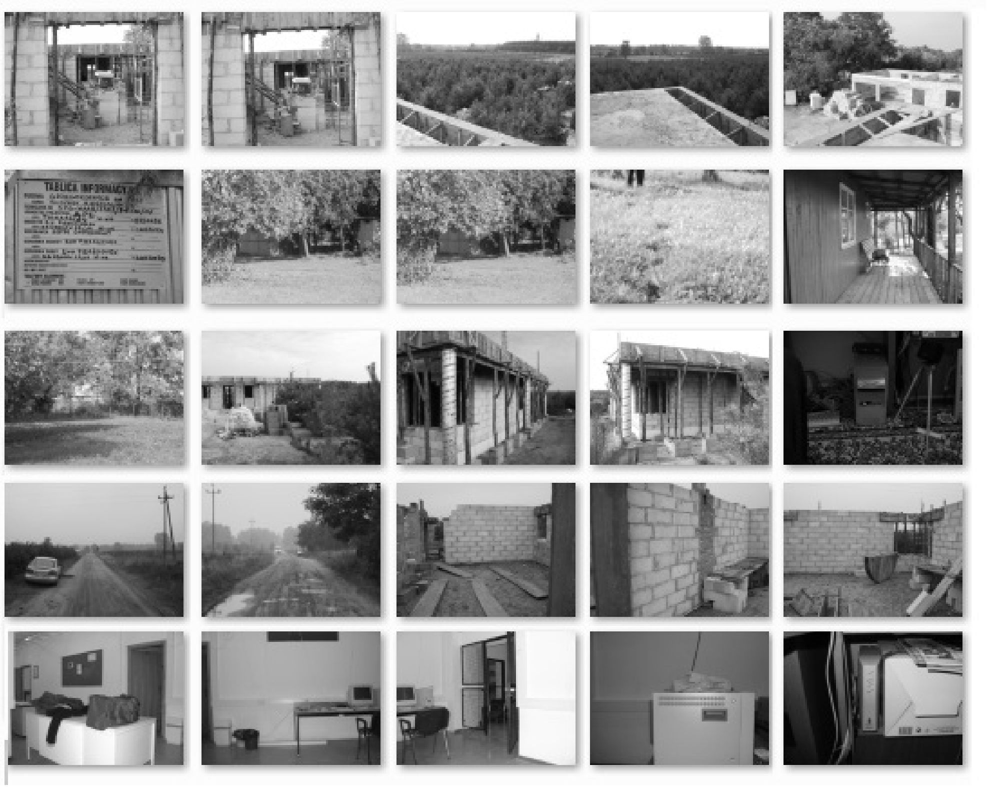


Figure 4.4: Negative training instances in face detection

The positive instances are 20×20 pixels images whereas the negative instances are images with 640×480 pixels.

4.2.2 Face detection system

Our face detector is trained by the proposed Penalized AdaBoost (explained in Chapter 3) instead of Discrete AdaBoost which was used in Viola-Jones method. The framework of our face training mechanism is shown in Fig. 4.5.

In Fig. 4.5, the Haar-like features are computed from the positive and negative images. Thus, the trained face detector can distinguish faces from non-faces. Furthermore, our trained face detector is more robust than Viola-Jones face detector because Penalized AdaBoost used in our system is proved better than other AdaBoost variants with respect to generalization errors.

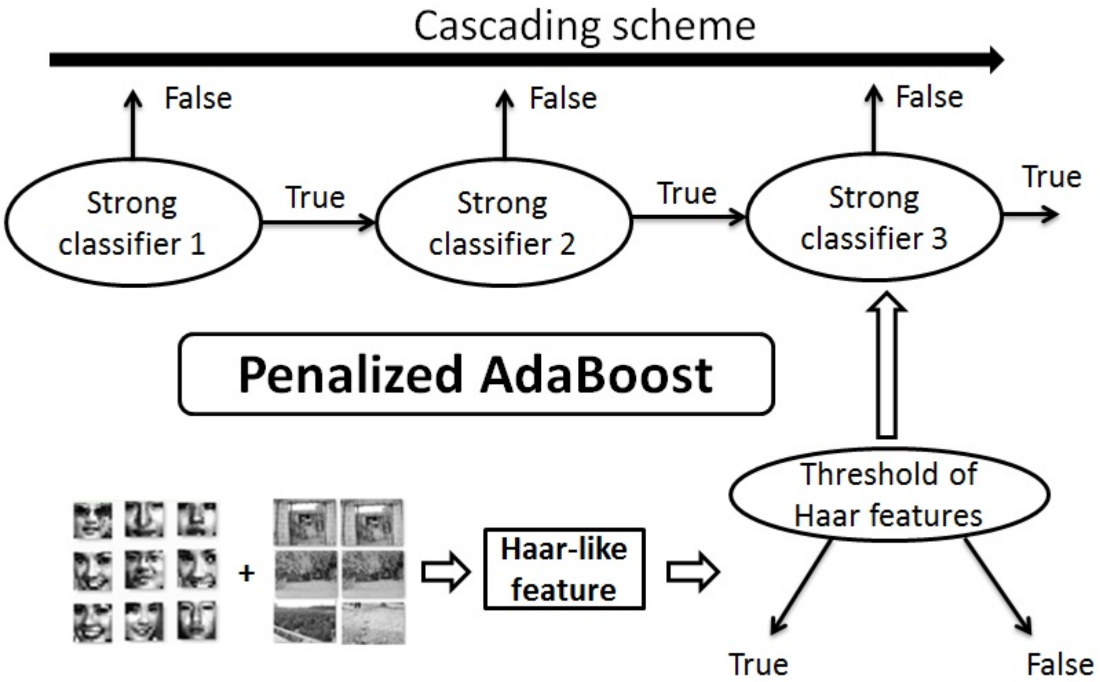


Figure 4.5: Training system using Penalized AdaBoost

4.3 Hand detection based on Penalized AdaBoost

In this section, first we describe how we collect training data for training a hand detector. Then we explain the problem of Viola-Jones hand detector. Finally we show how we improve the Viola-Jones hand detector by combining the proposed background-masking, Motion filtering, and Penalized AdaBoost.

4.3.1 Training data collection

Our hand detector totally handles 27 kinds of hand shapes as shown in Fig. 4.6. Thus, we collect the positive instances based on the 27 hand shapes.

In our hand detection system, each hand shape shown in Fig. 4.6 is captured from three different viewpoints by parallel moving the camera in X axis as shown in Fig. 4.7, in five directions $(-45^o, 0^o, 45^o, 90^o, 135^o)$ by counter-clockwise rotating our camera in X-Z plane as shown in Fig. 4.8, and in both left and right hands [6]. Therefore, we totally collect 30 $(3 \times 5 \times 2)$ training instances for each hand shape [6]. Figure 4.9 shows an example of the collection for one hand shape.

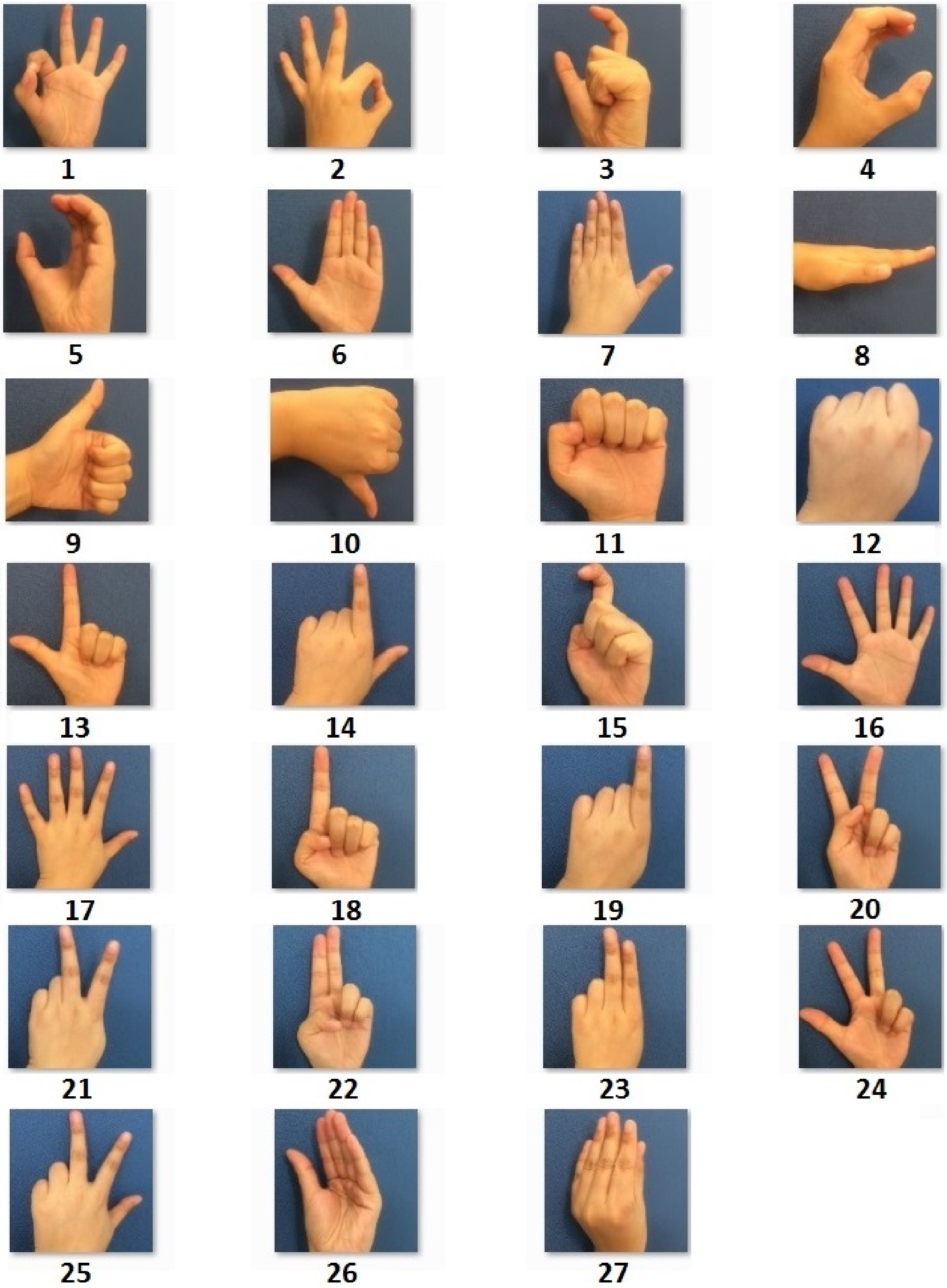


Figure 4.6: Hand shapes

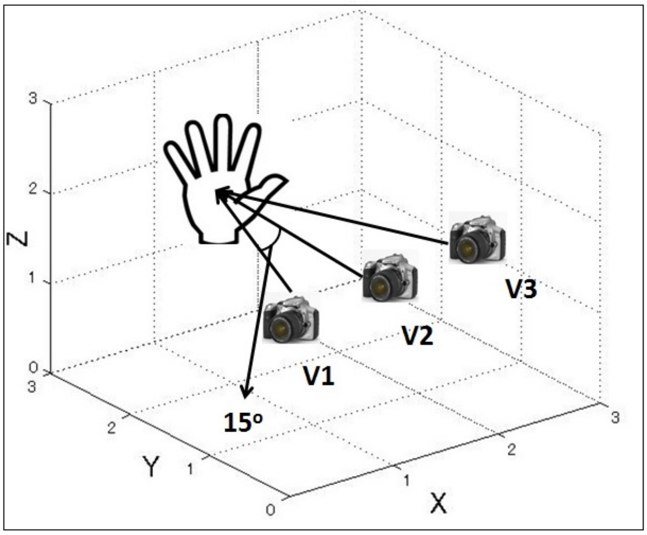


Figure 4.7: Three different viewpoints

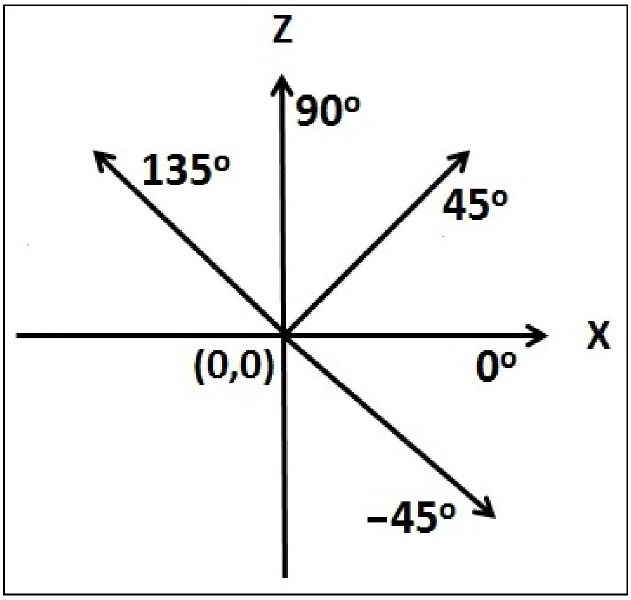


Figure 4.8: Five different directions

In Fig. 4.8, we notice that the five directions almost contain the moving range of a human hand.

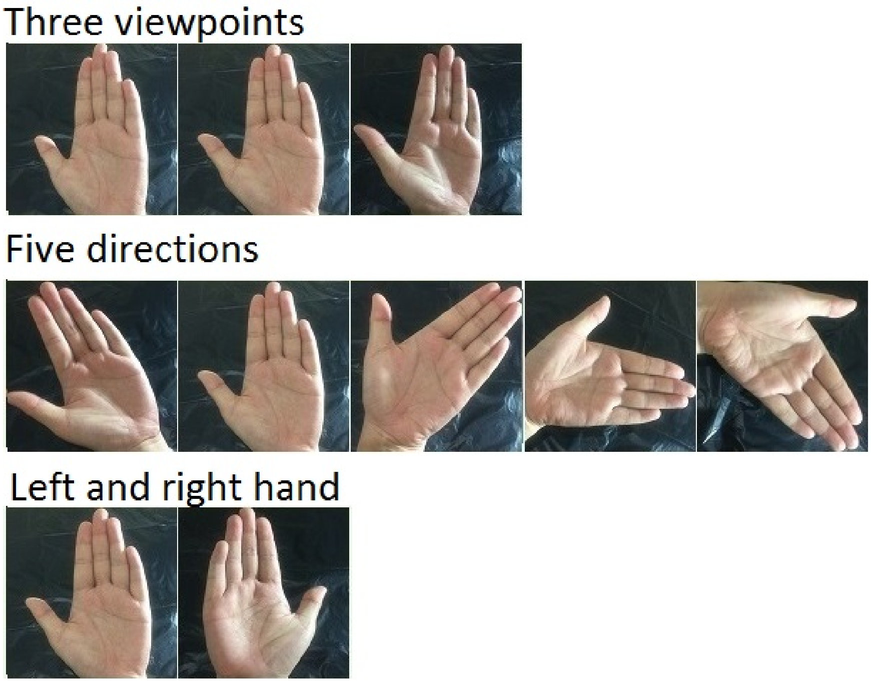


Figure 4.9: Training instances for one hand shape

4.3.2 Analysis of Viola-Jones hand detector

If we use Viola-Jones method to train a hand detector which can deal with different backgrounds, we need positive training instances in diverse backgrounds. Here we suppose that the Viola-Jones hand detector is designed to handle images in the three different backgrounds as shown in Fig. 4.10. Then for each hand shape in Fig. 4.6, we need collect $30 \times 3 = 90$ training instances. Obviously it will result in laborious labelling work and a long training time. Furthermore,

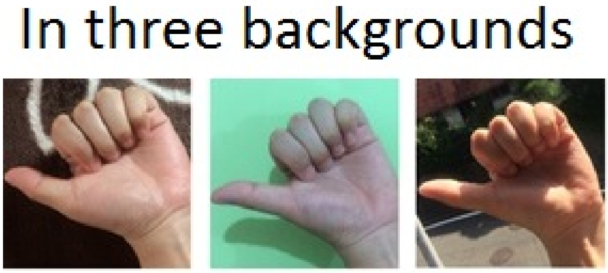


Figure 4.10: Training instances with three different backgrounds

when we used the positive instances in different backgrounds to train the hand detector, we found that the false positive rate is almost 100% [83]. Next we will analyze the reason for high false positive rate. In the training of the hand detector, Haar-like features of instances in different backgrounds should be calculated. This process may lead to the high false positive rate because of the background noise. For example, as we calculate the value of one Haar-like feature from the rectangle area in different backgrounds as shown in Fig. 4.11, we have to include non-hand pixels since there is a background part in every training instance. Therefore, according to different backgrounds, the value of the same Haar-like feature from the same hand shape varies. Then Viola-Jones method has to set loose thresholds to tolerate diverse values of the same Haar-like feature due to the various backgrounds. These loose thresholds will cause a high false positive rate of the hand detector.

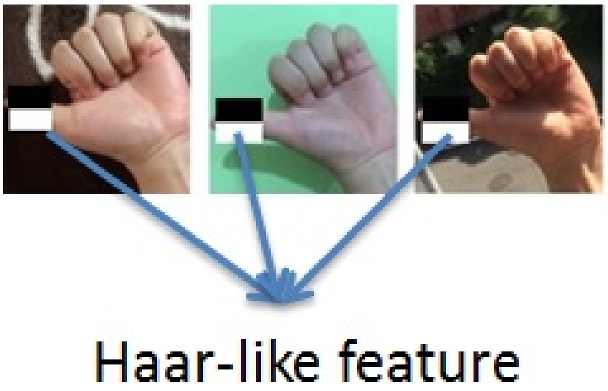


Figure 4.11: Haar-like feature computed from instances in three backgrounds

4.3.3 Background-masking

In order to reduce the training time and false positive rate, we propose a new technique which we call background-masking. It utilizes training instances in only one background (Here we use the black background to stand out hand pixels). This method has two parts. One part is for training instances, and another is for test images. For training instances, we combine HSV and YCrCb color spaces to do a skin color segmentation. Figure 4.12 shows the processing of background-masking for training instances. In Fig. 4.12, we set the pixel intensities of non-hand area to be 0 to remove the background noise.

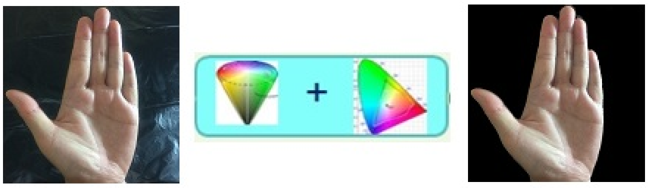


Figure 4.12: The process of background-masking for training instances

The second part of background-masking is for test images, the process is introduced as follows: we first detect the face area because face detection is far advanced than hand detection. Then we utilize a 3 × 3 window to extract skin color pixels in the cheek areas as shown in Fig. 4.13, and then we can get the thresholds of Y, Cr, Cb, H, S, and V using the extracted skin color pixels. Finally we implement the skin color segmentation for test images by these thresholds [6]. Figure 4.14 shows the result of background-masking for a test image.

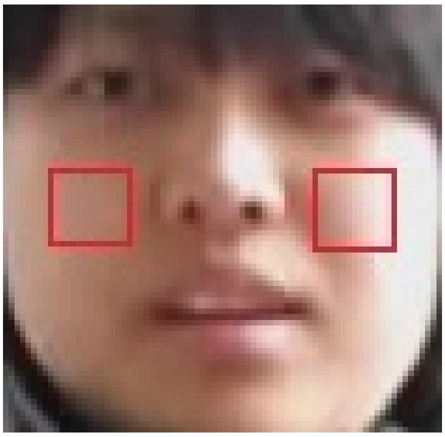


Figure 4.13: Skin color pixels in two checks

We apply background-masking to both the training instances and test images. Thus, the trained hand detector can recognize hands in background-masked test images. Since background-masking utilizes positive training instances with only one background, it requires far less training time when compared with Viola-Jones method which use instances in diverse backgrounds. Moreover, background-masking can reduce the false positive rate because it removes background noise effectively. Nevertheless, the hand detector is also required to distinguish hands from the face in Fig. 4.14. Furthermore, background-masking fails when the test images include some skin-color-like backgrounds. In section 4.3.5, we will explain how we solve the problem of background-masking.

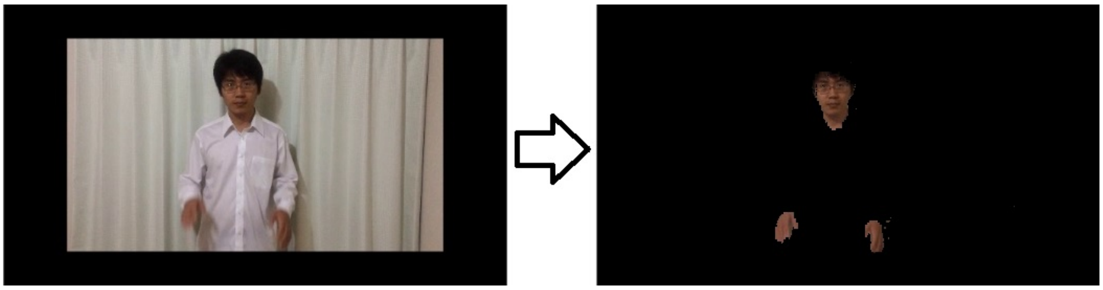


Figure 4.14: The result of background-masking for a test image

4.3.4 Hand training system based on Penalized AdaBoost

Since Penalized AdaBoost is proved more robust than other AdaBoost variants in Chapter 3, we use it to train the hand detector. The training mechanism for hand detector is shown in Fig. 4.15. In Fig. 4.15, we utilize background-masked hand patches as positive training instances, and non-hand images as negative training instances. The weak classifiers are computed based on Haar-like features of both the positive and negative training instances. In the training system, Penalized AdaBoost is applied to boost several weak classifiers into one strong classifier. Similarly to Viola-Jones method, strong classifiers in our system are also organized by a cascading scheme. On one hand, background-masking reduces both the training instances and false positive rate. On the other hand, Penalized AdaBoost improves the classification performance of the trained hand detector.

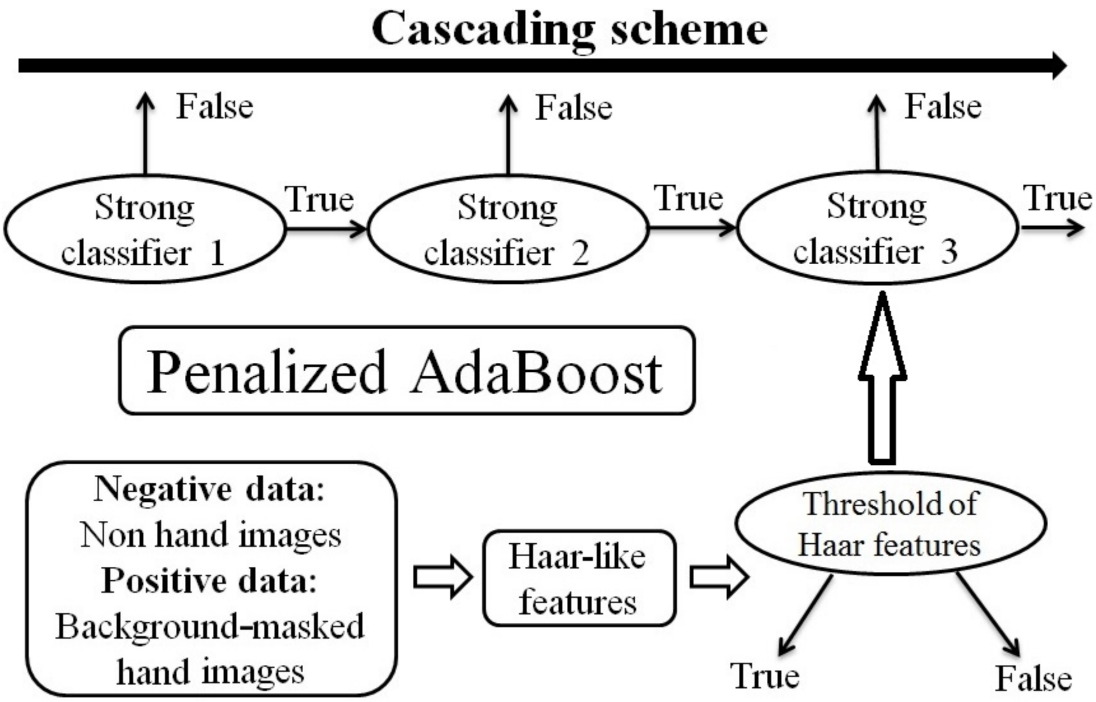


Figure 4.15: Training mechanism for training the hand detector

4.3.5 Hand detection system

In this section, we describe the proposed hand detection system which is based on the trained detector shown in Fig. 4.15. In the hand detection system, first we utilize background-masking to process the test frame, and then we adopt a motion filter which is based on frame difference to reduce the detection range. Figure 4.16 shows the structure of our hand detection system. From Fig. 4.16, we can see that the motion filter solves the problems of background-masking discussed in Section 4.3.3.

The motion filter can exclude non-moving parts effectively. Nevertheless, the hand areas may be also removed if the motion is not large sufficiently. To solve the problem, we check whether a hand is detected in each iteration [6]. If a hand is detected, we move to process the next frame; otherwise, the frame difference between f and $f+2$ frames will be used to enlarge the moving part in the frame-differenced image [6]. Figure 4.17 shows one example of hand detection by our proposed system [6]. In Fig. 4.17 (b), we notice that background-masking fails

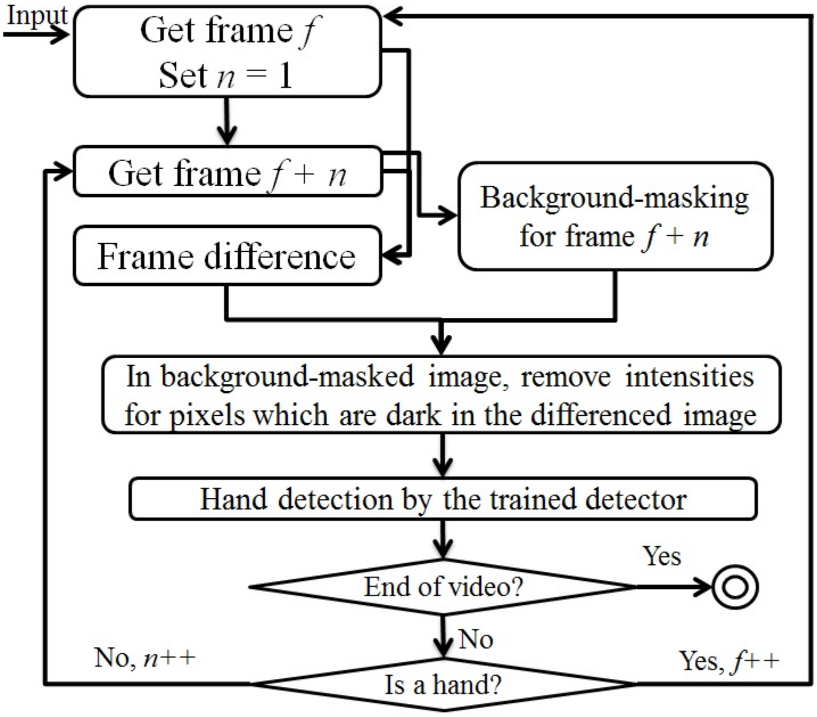
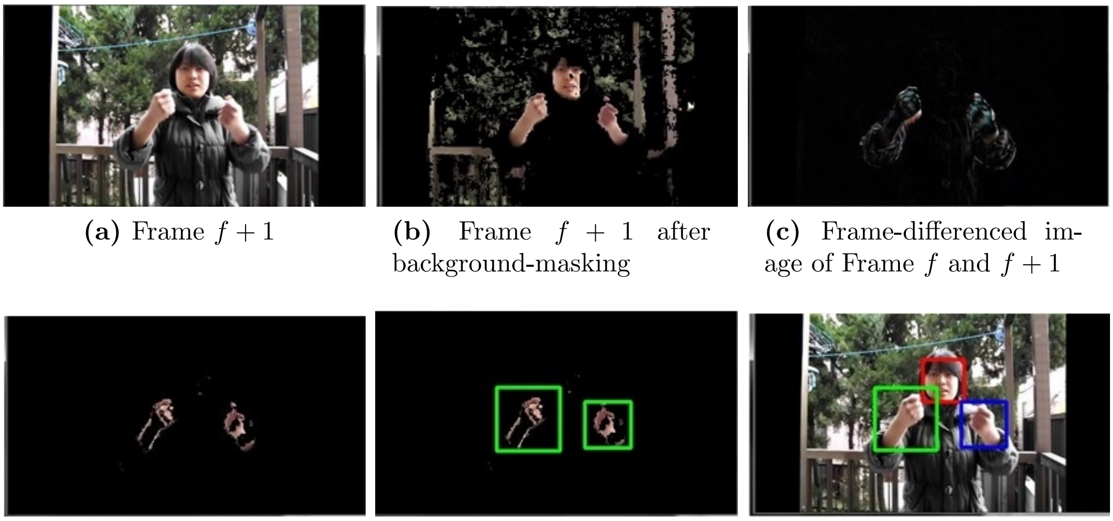

Input
Get frame f
Set $n = 1$
Get frame $f + n$
Background-masking for frame $f + n$
Frame difference
In background-masked image, remove intensities for pixels which are dark in the differenced image
Hand detection by the trained detector
End of video?
Yes
No
Is a hand?
No, n++
Yes, f++

Figure 4.16: The structure of the proposed hand detection system



(a) Frame $f + 1$

(b) Frame $f + 1$ after background-masking

(c) Frame-differenced image of Frame f and $f + 1$

(d) In (b), removes intensities of pixels dark in (c)

(e) The result after doing hand detection in (d)

(f) The result of face and hand detection

Figure 4.17: An example of hand detecting

because of the illumination change and complicated background in the test frame. However, the motion filter efficiently excludes non-hand parts in the background-masked image as shown in Fig. 4.17 (d). Our hand detection system performs well in test images in diverse backgrounds, even in skin-color-like backgrounds as long as these backgrounds are stationary.

Our hand detection system is different from Viola-Jones method in the following points.

- Differently from Viola-Jones method which is based on Discrete AdaBoost, our system utilizes the new proposed Penalized AdaBoost to train the face and hand detectors. Penalized AdaBoost can improve the classification performance of the trained detectors.

- Compared with Viola-Jones method, our system adopts background-masking for both training instances and test images. The proposed background-masking not only saves a lot training time, but also reduces the false positive rate significantly.

- Our detection system applies a motion filter to distinguish hands from other skin color parts. However, Viola-Jones method does not use any motion information.

Chapter 5

Proposed Parameterized AdaBoost

5.1 Relationship between margins and weight updating in Real AdaBoost

Parameterized AdaBoost was proposed to speed up the training process of Real AdaBoost [8]. In this section, we will briefly introduce some basic ideas of Real AdaBoost. Real AdaBoost can enlarge the margins of training data iteratively because there is a relationship between the weight updating policy and the margins of training instances. According to (2.4) and (2.5), the updated weight of an instance x_i can be computed by

$$w_{i,t+1} = \frac{w_{i,t} e^{-y_i f_t(x_i)}}{\sum_i w_{i,t} e^{-y_i f_t(x_i)}}. \tag{5.1}$$

Similarly, we can calculate $w_{i,t}$ as

$$w_{i,t} = \frac{w_{i,t-1} e^{-y_i f_{t-1}(x_i)}}{\sum_i w_{i,t-1} e^{-y_i f_{t-1}(x_i)}}. \tag{5.2}$$

Then substitute (5.2) for $w_{i,t}$ in (5.1), we get

$$w_{i,t+1} = \frac{w_{i,t-1} exp[-y_i \sum_{t-1}^{t} f_t(x_i)]}{\sum_i w_{i,t-1} exp[-y_i \sum_{t-1}^{t} f_t(x_i)]}. \tag{5.3}$$

In the same way, the updated weight can be deduced by

$$w_{i,t+1} = \frac{w_{i,1} \times exp[-y_i F_t(x_i)]}{\sum_i w_{i,1} \times exp[-y_i F_t(x_i)]} = \frac{exp[-y_i F_t(x_i)]}{\sum_i exp[-y_i F_t(x_i)]}. \tag{5.4}$$

Where $F_t(x_i) = \sum_{t=1}^{t} f_t(x_i)$ and $\sum_i exp[-y_i F_t(x_i)]$ is a normalizer. From (2.12) and (5.4), we find that $w_{i,t+1}$ is inversely exponentially proportional to the margins. This means that Real AdaBoost assigns larger weights to small-margin instances. Focusing on increasing small margins is effective at reducing the training error. Nevertheless, it may lead to the misclassification of other instances whose margins are positive, but near to 0. The margins of these instances are probably changed from positive into negative due to the misclassification. If we can curb the misclassification of instances whose margins are slightly larger than 0 on the premise that large weights are assigned to instances with small margins, we can speed up the decreasing of training error.

5.2 Parameterized AdaBoost

In this section, we describe Parameterized AdaBoost which introduces a parameter to penalize the misclassification of already correctly classified instances. Since Parameterized AdaBoost prevents positive margins that are slightly larger than 0 from being turned into negative. It can obtain a faster convergence of training error when compared with Real AdaBoost.

5.2.1 Basic algorithm

Parameterized AdaBoost is introduced as follows:

Parameterized AdaBoost

1. Set initial conditions: $w_{i,1} = 1/N$ and $F_0(x_i) = 0$, where $i = 1, 2, ..., N$.
2. Repeat the following steps for $t = 1, 2, ..., T$:
(a) Train a CART, and then compute W^j_{t+} and W^j_{t-} for every partition S^j_t the same as in Step 2(a) of Real AdaBoost.
(b) Set the weak hypotheses by (2.3).
(c) Set $F_t(x_i) = F_{t-1}(x_i) + f_t(x_i)$ and $\alpha \in (0, 1)$. Then update the weights of all instances as

$$w_{i,t+1} = exp[-y_i F_t(x_i) - \alpha|y_i F_t(x_i)|]/G_t. \tag{5.5}$$

$$G_t = \sum_i exp[-y_i F_t(x_i) - \alpha|y_i F_t(x_i)|]. \tag{5.6}$$

3. Finally output the strong classifier $H(x_i) = sign[F_T(x_i)]$.

From the above algorithm, we can see that the difference between Real and Parameterized AdaBoost is the weight updating policy. In Parameterized AdaBoost, we add a parameter α and an absolute item $|y_i F_t(x_i)|$ to curb the misclassification of instances whose margins are slightly larger than 0.

With respect to the tuning of parameter α, we measure the training and test errors on a data set Gamma Telescope which includes nearly 20000 instances by using different values of α ($\alpha = 0.1, 0.3, 0.5, 0.7, 0.9$). Finally we find that $\alpha = 0.5$ has the best overall performance. The values smaller than 0.5 can not obtain a sufficient improvement of training errors whereas the values larger than 0.5 is prone to overfiting [8].

5.2.2 Comparison of Real and Parameterized AdaBoost

In this section, first we demonstrate that Parameterized AdaBoost is as effective as Real AdaBoost because it also assigns larger weights to small-margin instances. We notice that the weight updating equation (5.5) in Parameterized AdaBoost can be replaced with

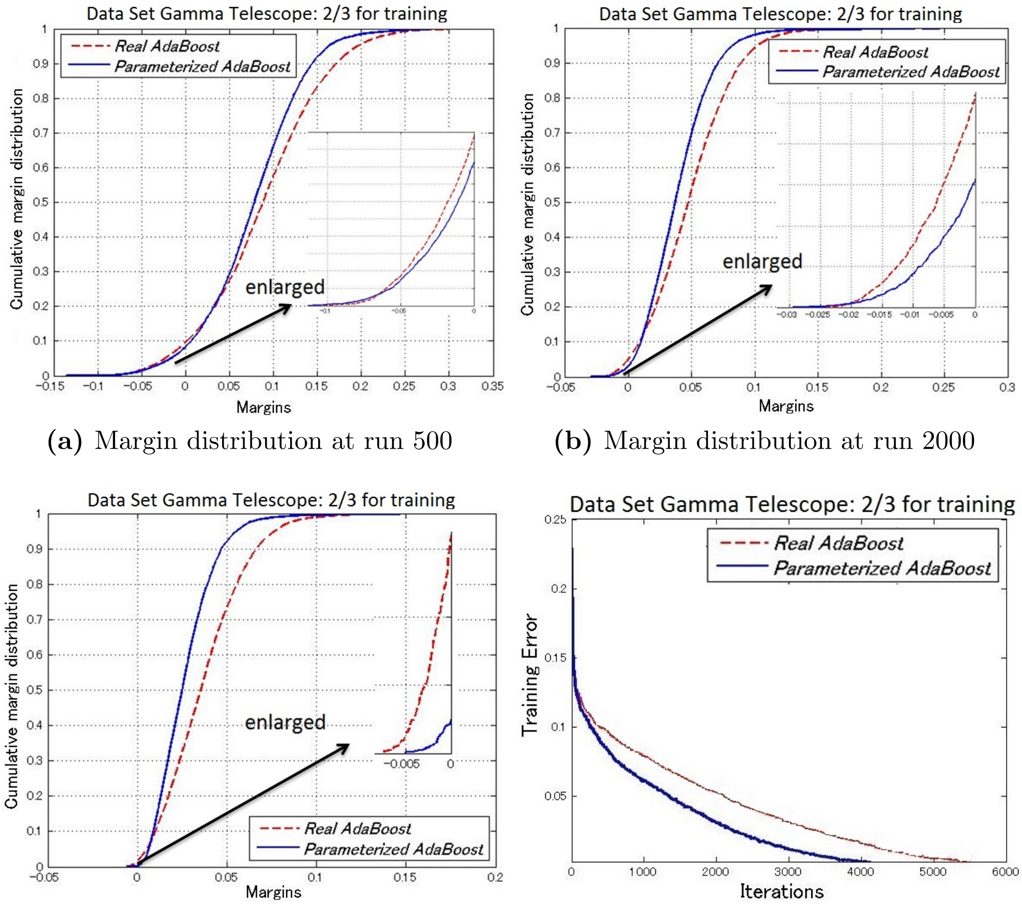
$$w_{i,t+1} = \begin{cases} exp[-(1+\alpha)y_i F_t(x_i)]/G_t, if & y_i F_t(x_i) > 0 \\ exp[-(1-\alpha)y_i F_t(x_i)]/G_t, & otherwise \end{cases} \tag{5.7}$$

From (5.7), we can conclude that the weight of an instance x_i reduces as its margin increases as long as $0 < \alpha < 1$. Therefore, small-margin instances in Parameterized AdaBoost can always get larger weights the same as in Real AdaBoost. We notice that this condition can be guaranteed if and only if $\alpha \in (-1, +1)$. Nevertheless, we need $\alpha > 0$ to curb the misclassification of instances whose margins are near 0. In (5.5), $\alpha|y_i F_t(x_i)|$ is positive if and only if $\alpha > 0$. Moreover, the value of $\alpha|y_i F_t(x_i)|$ is large when the margin of x_i is far from 0. On the contrary, this value is small when the margin of x_i is near 0. This means instances whose margins are near 0 can obtain more weight than those whose margins are far from 0. Because Parameterized AdaBoost focuses more on instances whose margins are near 0, it can curb the misclassification of these instances. By assigning more weight to these instances, Parameterized AdaBoost can prevent the margins which are slightly larger than 0 from being turned into negative, and encourage the margins which are slightly smaller than 0 to become positive. Therefore, it can reduce the training error more than Real AdaBoost.

5.2.3 Margins in Parameterized AdaBoost

From the above analysis, we can conclude that Parameterized AdaBoost increases the number of positive margins more than Real AdaBoost in each iteration. We compare the cumulative margin distributions in Parameterized AdaBoost with those in Real AdaBoost. Figures. 5.1 (a), 5.1 (b), and 5.1 (c) show the results on a data set Gamma Telescope after 500, 2000, and 4000 iterations respectively [8]. While Fig. 5.1 (d) shows the convergence of training error in Parameterized AdaBoost compared with that in Real AdaBoost [8]. In Fig. 5.1, the dashed curves represent Real AdaBoost whereas the solid curves represent Parameterized AdaBoost. To produce these margin distributions, we use CART-2 as the weak classifiers since it can show the difference of Parameterized and Real AdaBoost clearly (Boosting with CART-1 converges too slow and boosting with CART-3 converges too fast).

From Fig. 5.1, we can see that Parameterized AdaBoost has less training instances with negative margins than Real AdaBoost at the same round. Thus it can obtain a faster convergence of training error than Real AdaBoost [8].



(a) Margin distribution at run 500

(b) Margin distribution at run 2000

(c) Margin distribution at run 4000

(d) Convergence of training error

Figure 5.1: (a)–(c) Cumulative distributions of margins, and (d) Training errors

5.3 Discussion

Real AdaBoost reduces the training errors faster than other AdaBoost variants because it directly optimizes the upper bound of training errors [90]. To speed up the training process of Real AdaBoost, we propose Parameterized AdaBoost which adopts a new weight adjustment policy. Parameterized AdaBoost is as effective as Real AdaBoost since it gives more weight to small-margin instances. However, differently from Real AdaBoost, Parameterized AdaBoost focuses more on instances whose margins are near 0 and tries to keep their margins positive in future loops. Thus, it increases the number of positive margins more than Real

AdaBoost in each loop. This kind of weight updating is not only suitable for Real AdaBoost, but also easy to be applied to other AdaBoost variants such as Gentle AdaBoost, Modest AdaBoost, Float AdaBoost, and so on [8].

Chapter 6

Experiments

6.1 For Penalized AdaBoost

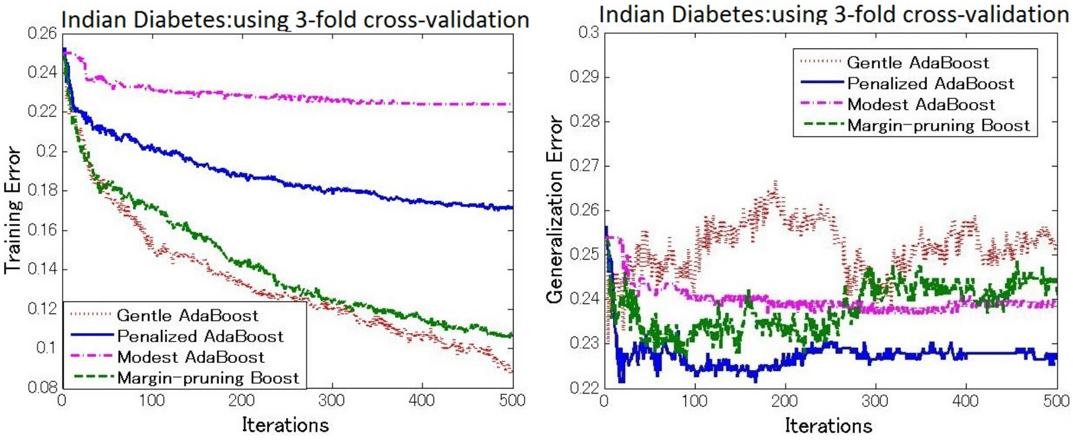
This section evaluates the performance of our proposed Penalized AdaBoost. First we explain the instructions of our experiments, and then we show the experimental results, finally we discuss the generalization ability of Penalized AdaBoost by analyzing these experimental results.

6.1.1 Experiments setting

In this experiment, we compare Penalized AdaBoost with Gentle AdaBoost, Modest AdaBoost, and Margin-pruning Boost by using 26 binary classification data sets from UCI repository [91]. In this dissertation, we use Matlab AdaBoost Toolbox provided in [92] to run AdaBoost variants. To evaluate the test errors, we use 3-fold cross-validation. Next we will explain what a cross-validation is. A k-fold cross-validation means the training data is randomly divided into k equal size sub sets. Then one of the k sub sets is retained for testing, and the other $k-1$ sub sets are used for training. The cross-validation process will be repeated k times by utilizing each of the k sub sets as test data. Finally we can get k test errors, and then the final generalization error is the average of the k test errors.

6.1.2 Results evaluation

This section shows the experimental results. Figures. 6.1 (a) and 6.1 (b) show the comparison results of training and generalization errors on a data set Indian Diabetes [7]. We also evaluate the generalization errors in other data sets after 200 iterations. Table. 6.1 summarizes the results [7]. In addition, we measure the generalization errors on the total 26 data sets after 500 iterations. Table. 6.2 shows the comparison results [7]. In Tabs. 6.1 and 6.2, No.S and No.F denote the number of instances and number of features, GAB, MAB, MPB, and PAB represent Gentle AdaBoost, Modest AdaBoost, Margin-pruning Boost and Penalized AdaBoost separately; and the underlined numbers shows the best performance.



(a) Training errors

(b) Generalization errors

Figure 6.1: Comparison of training and generalization errors

6.1.3 Analysis and discussion

From Fig. 6.1, we can see that the proposed Penalized AdaBoost outperforms other three AdaBoost variants in terms of the generalization errors which are measured by the classification error on the test data. On the other hand, we can conclude that Modest AdaBoost can hardly reduce the training error from Fig. 6.1 (a). Nevertheless, the training error in Penalized AdaBoost is decreased gradually. We also notice that Penalized AdaBoost decreases the training error slower than Gentle AdaBoost and Margin-pruning Boost.

Table 6.1: Comparison results of generalization errors after 200 iterations

Data sets:26	No.S	No.F	GAB	MAB	MPB	PAB
Banana	5000	2	0.2713	0.2804	0.2526	0.2809
Bankruptcy	175	6	0.0400	0.0344	0.0400	0.0286
Australian	690	14	0.1435	0.1435	0.1391	0.1304
Banknote	1372	4	0.0022	0.0372	0.0058	0.0080
Breast Cancer	569	4	0.0246	0.0422	0.0669	0.0281
Blood Transfusion	748	4	0.2312	0.2352	0.2218	0.2298
Climate model	540	20	0.0593	0.0963	0.0704	0.0630
Glass	214	10	0.0560	0.0701	0.0886	0.0653
Gamma Telescope	19020	10	0.1546	0.2297	0.1454	0.1548
German	1000	24	0.2580	0.2950	0.2470	0.2450
Heart Disease	270	13	0.2333	0.1704	0.2000	0.1630
Hepatitis	155	19	0.2130	0.2515	0.2129	0.1869
Indian Diabetes	768	8	0.2578	0.2383	0.2318	0.2253
Indian Liver	579	9	0.3005	0.3230	0.2850	0.3126
Ionosphere	351	34	0.0969	0.0684	0.0997	0.0826
Parkinsons	195	22	0.0872	0.1692	0.0974	0.0923
Planning Relax	182	12	0.4012	0.3519	0.4175	0.3186
Ringnorm	7400	20	0.0268	0.0470	0.0277	0.0509
Spambase	4601	57	0.0546	0.0895	0.0528	0.0617
SPECTFHeart	267	44	0.2285	0.2472	0.2584	0.1985
Splice	2991	60	0.0675	0.0919	0.0612	0.0639
Steel Plates	1941	27	0.2324	0.2751	0.2318	0.2421
Twonorm	7400	20	0.0305	0.0315	0.0292	0.0304
Waveform	3304	21	0.0975	0.0962	0.0917	0.0853
Wine Quality	6497	11	0.0054	0.0240	0.0052	0.0054
WPBC	198	34	0.2727	0.3030	0.3081	0.2323
Sum			3.8465	4.2421	3.8880	3.5857
Comparison to GAB Comparison to GAB			0.0000 improved	0.3956 degraded	0.0415 degraded	0.2608 improved

From Tabs 6.1 and 6.2, we can see that Penalized AdaBoost achieves better generalization errors than the other three variants. Furthermore, It obtains the best performance in 10 data sets in Tab. 6.1, and in 16 data sets in Tab. 6.2. Comparing the results in Tab. 6.1 with that in Tab. 6.2, we also conclude that

Table 6.2: Comparison results of generalization errors after 500 iterations

Data sets:26	GAB	MAB	MPB	PAB
Banana	0.2725	0.2804	0.2547	0.2738
Bankruptcy	0.0400	0.0344	0.0400	0.0458
Australian	0.1638	0.1435	0.1377	0.1333
Banknote	0.0036	0.0372	0.0058	0.0058
Breast Cancer	0.0246	0.0369	0.0756	0.0246
Blood Transfusion	0.2540	0.2352	0.2272	0.2258
Climate model	0.0648	0.0963	0.0722	0.0593
Glass	0.0607	0.0748	0.1025	0.0606
Gamma Telescope	0.1519	0.2292	0.1424	0.1473
German	0.2730	0.2950	0.2490	0.2430
Heart Disease	0.2481	0.1704	0.2000	0.1852
Hepatitis	0.2130	0.2515	0.2129	0.2128
Indian Diabetes	0.2526	0.2383	0.2409	0.2266
Indian Liver	0.3074	0.3558	0.2953	0.3057
Ionosphere	0.0940	0.0627	0.0997	0.0912
Parkinsons	0.0974	0.1641	0.0974	0.0872
Planning Relax	0.4397	0.3519	0.4175	0.3462
Ringnorm	0.0249	0.0309	0.0251	0.0318
Spambase	0.0574	0.0895	0.0517	0.0572
SPECTFHeart	0.2285	0.2584	0.2772	0.2022
Splice	0.0746	0.0919	0.0622	0.0605
Steel Plates	0.2293	0.2777	0.2267	0.2334
Twonorm	0.0296	0.0286	0.0296	0.0276
Waveform	0.0984	0.0962	0.095	0.0881
Wine Quality	0.0054	0.0231	0.0060	0.0051
WPBC	0.2828	0.3030	0.3081	0.2374
Sum	3.9920	4.2569	3.9524	3.6175
Comparison to GAB	0.0000	0.2649	0.0396	0.3745
Comparison to GAB	improved	degraded	improved	improved
Parallel comparison to Tab.1	0.1455	0.0148	0.0644	0.0318
Parallel comparison to Tab.1	degraded	degraded	degraded	degraded

Penalized and Modest AdaBoost are more robust against overfitting than Gentle AdaBoost and Margin-pruning Boost [7]. In general, Penalized AdaBoost is more robust and stable than other compared variants. Moreover, its performance improves as the number of iterations increases.

6.2 For face and hand detection system

In this section, we evaluate the performance of our face-hand detection system. Here we test the face detector and hand detector respectively.

6.2.1 Experiments setting

First we evaluate our face detector (trained by our proposed Penalized AdaBoost) by comparing it with other detectors trained by Gentle and Modest AdaBoost. Here we use 4 kinds of data sets to test the three different face detectors. The 4 kinds of data sets are listed as follows:

- Dataset 1: one-face images in Bao database [93].
- Dataset 2: multiple-face images in Bao database [93].
- Dataset 3: CMU-MIT database [94].
- Dataset 4: sign language video signed by different signers.

Secondly, we evaluate the proposed hand detection system which is a combination of Haar-like features, background-masking, motion filtering, and Penalized AdaBoost. Here we test the following hand detectors under five different circumstances [6]:

- Detector 1: hand detector trained by Gentle AdaBoost; using positive instances that are collected in three different backgrounds shown in Fig. 4.10 (Totally 27 $\textit{hand types} \times 5 \textit{ directions} \times 3 \textit{ viewpoints} \times 2 \textit{ both hands} \times 3 \textit{ backgrounds} = 2430$ instances), and 4860 negative instances.
- Detector 2: hand detector trained by Gentle AdaBoost; using background-masked hand images (Totally 27 $\textit{hand types} \times 5 \textit{ directions} \times 3 \textit{ viewpoints} \times 2 \textit{ both hands} \times 1 \textit{ backgrounds} = 810$ instances), and 1620 negative instances.
- Detector 3: hand detector trained by Gentle AdaBoost; using positive and negative training data the same as Detector 2, and using the proposed hand detection system (shown in Fig. 4.16).

- Detector 4: hand detector trained by Modest AdaBoost; using positive and negative training data the same as Detector 2, and using the proposed hand detection system (shown in Fig. 4.16).
- Detector 5: hand detector trained by the proposed training system as shown in Fig. 4.15 (based on Penalized AdaBoost); and using the proposed hand detection system (shown in Fig. 4.16).

For each hand detector, we train 16 cascaded strong classifiers. The size of each training instance is 20 × 20 pixels. To compare the five detectors, we use 4 kinds of videos, and they are listed as follows:

- Video 1: video in a simple background; signed by two signers; including both long and short sleeves scenes; and totally 381 frames.
- Video 2: video in a skin-color-like background; signed by two signers; including both long and short sleeves scenes; and totally 421 frames.
- Video 3: video in a complicated background; signed by one signer; including long sleeves scene; and totally 880 frames.
- Video 4: video in a complicated background with illumination change; signed by one signer; including long sleeves scene, and totally 1081 frames.

The frame size of all the four videos is 480 × 270 pixels.

6.2.2 Results evaluation in face detection

We compare the face detector train by our Penalized AdaBoost with those trained by Gentle and Modest AdaBoost. Tables 6.3, 6.4, 6.5, and 6.6 shows the comparison results in four face data sets respectively.

Table 6.3: Comparison results on face Dataset 1

Dataset 1: 150 faces	Recall (%)	Precision (%)	F-measure (%)
Detector-GAB	*47.33*	98.61	*63.96*
Detector-MAB	**54.67**	*94.25*	69.20
Detector-PAB	53.33	**98.77**	**69.26**

Table 6.4: Comparison results on face Dataset 2

Dataset 2: 1258 faces	Recall (%)	Precision (%)	F-measure (%)
Detector-GAB	*55.25*	91.69	*68.95*
Detector-MAB	**65.50**	*84.86*	73.93
Detector-PAB	65.10	**91.82**	**76.19**

Table 6.5: Comparison results on face Dataset 3

Dataset 3: 182 faces	Recall (%)	Precision (%)	F-measure (%)
Detector-GAB	*54.40*	97.06	*69.72*
Detector-MAB	57.69	*92.11*	70.95
Detector-PAB	**59.34**	**97.30**	**73.72**

Table 6.6: Comparison results on face Dataset 4

Dataset 4: 1483 faces	Recall (%)	Precision (%)	F-measure (%)
Detector-GAB	*69.05*	97.80	*80.95*
Detector-MAB	81.86	*97.50*	90.00
Detector-PAB	**88.20**	**99.47**	**93.50**

In tabs. 6.3, 6.4, 6.5, and 6.6, Detector-GAB, Detector-MAB, and Detector-PAB denote the hand detectors trained by Gentle, Modest, and Penalized AdaBoost respectively. The bold numbers show the best results whereas the tilted numbers show the worst results.

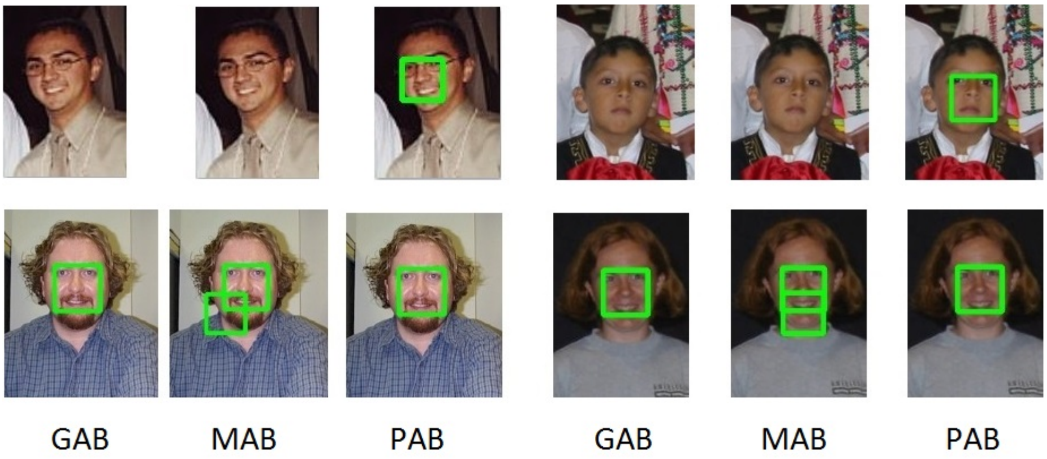


Figure 6.2: Examples of test results on Dataset 1

Figures. 6.2, 6.3, 6.4, and 6.5 shows some examples of test results on Datasets 1, 2, 3, and 4 separately. Here GAB, MAB, and PAB means Gentle, Modest, and Penalized AdaBoost respectively.

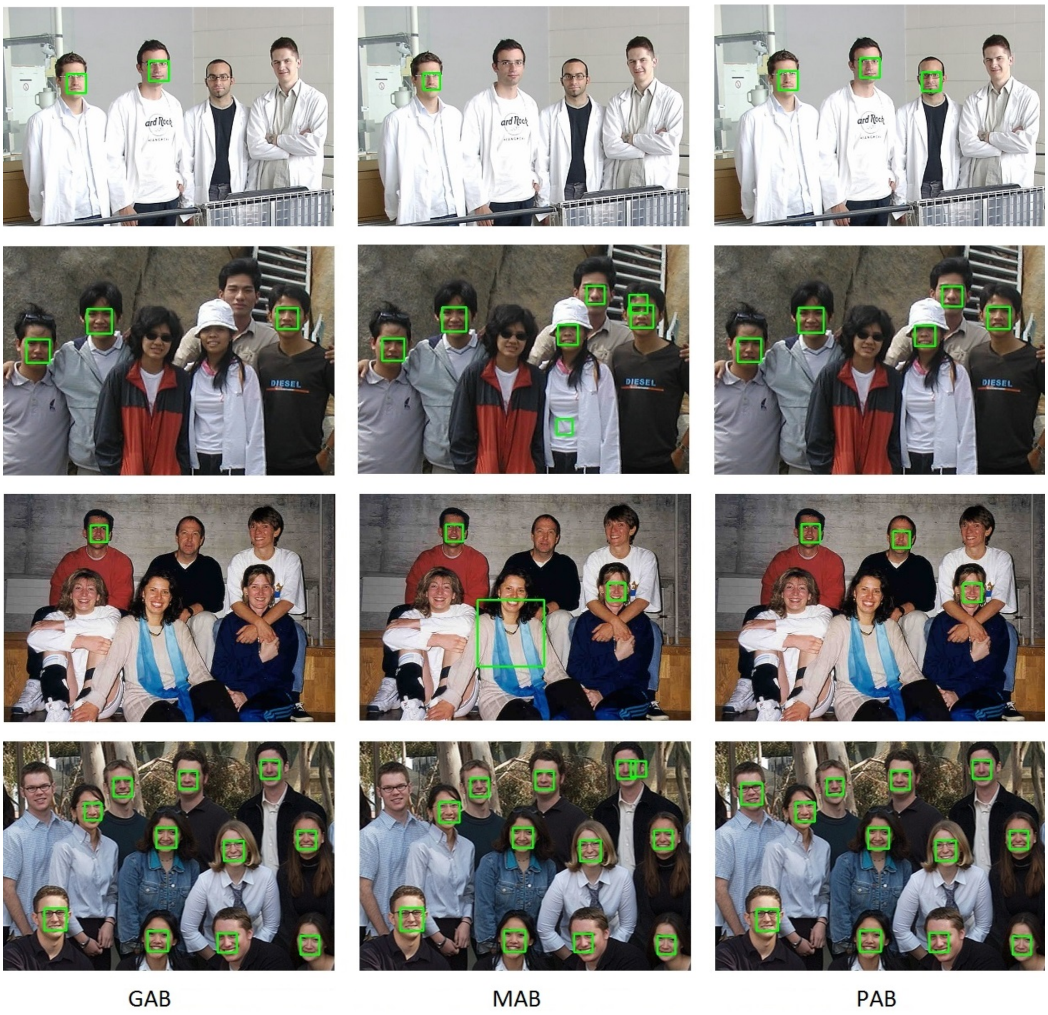


Figure 6.3: Examples of test results on Dataset 2

6.2.3 Results evaluation in hand detection

This section evaluates the performance of the five detectors mentioned in Section 6.2.1 in both the training and test phases. Figure 6.6 shows the ROC curves of the five detectors in the training [6]. While Tab. 6.7 shows the required training time of the five hand detectors [6].

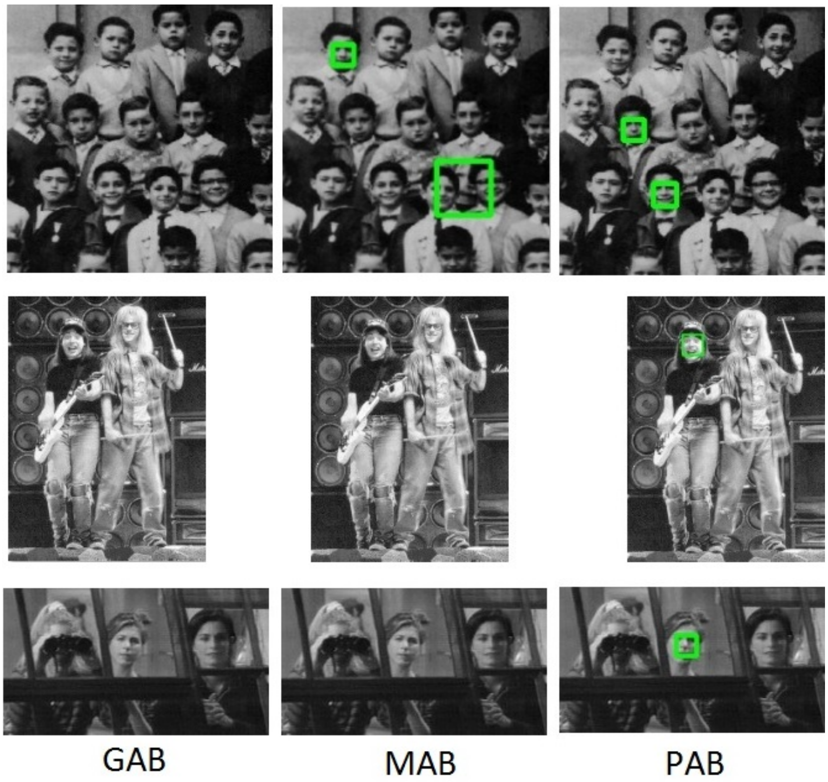


Figure 6.4: Examples of test results on Dataset 3

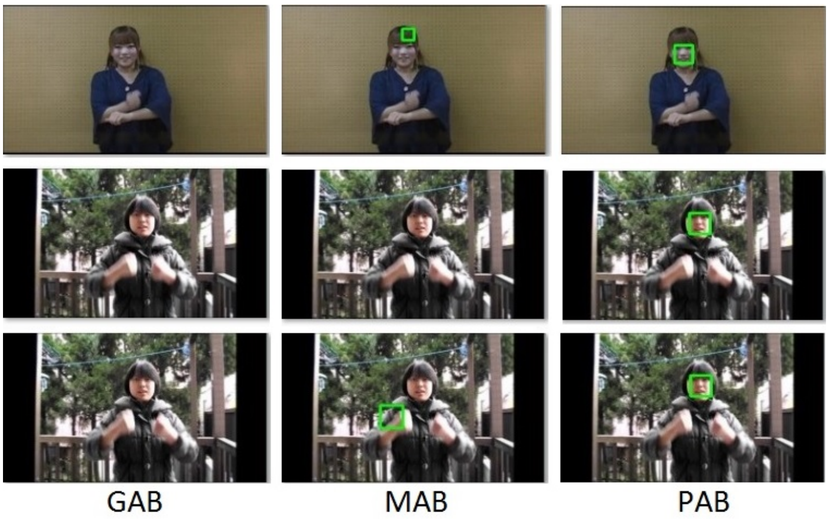


Figure 6.5: Examples of test results on Dataset 4

In this dissertation, we test the five hand detectors by four videos. Tables.

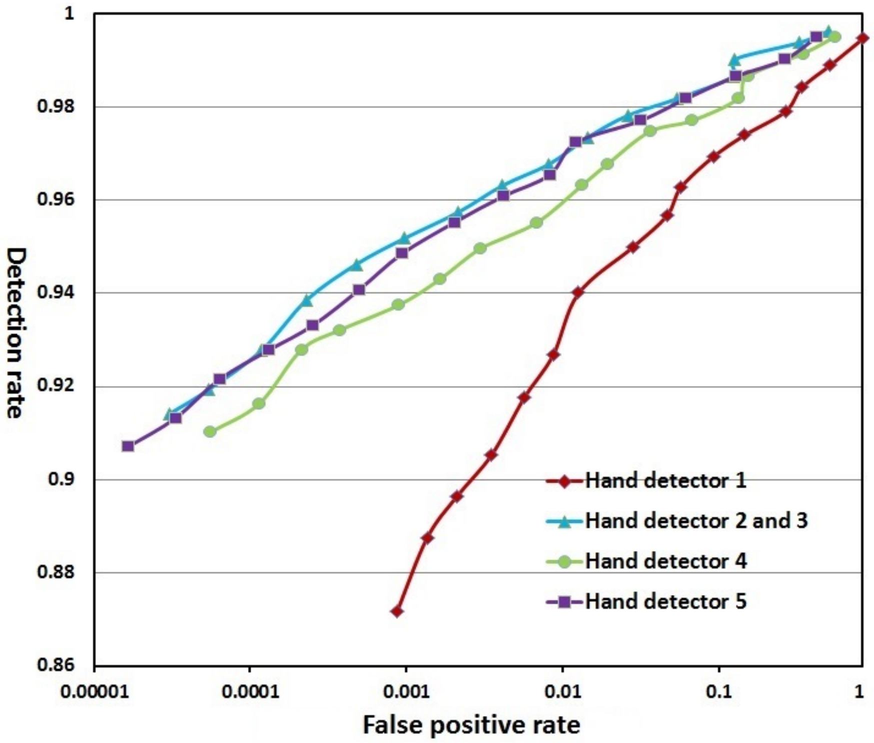


Figure 6.6: ROC curves of the five detectors

Table 6.7: Required training time of the five detectors

Detector-1	Detector-2	Detector-3	Detector-4	Detector-5
19 hours and 22 minutes	27 minutes	27 minutes	43 minutes	41 minutes

6.8, 6.9, 6.10, and 6.11 show the comparison results in Videos. 1, 2, 3, and 4 respectively [6]. Here the bold values represent the best results whereas the tilted values stand for the worst results. Figure 6.7 shows some examples of the test results in hand detection.

Table 6.8: Comparison results of the five detectors on Video 1

Video 1: 697 hands	Recall(%)	Precision(%)	F-measure(%)
Detector-1	*78.48*	*45.36*	*57.49*
Detector-2	80.92	92.46	86.31
Detector-3	83.79	89.02	86.33
Detector-4	**86.94**	95.28	90.92
Detector-5	86.51	**95.87**	**90.95**

Table 6.9: Comparison results of the five detectors on Video 2

Video 2: 657 hands	Recall(%)	Precision(%)	F-measure(%)
Detector-1	*55.71*	*37.42*	*44.77*
Detector-2	78.23	78.71	78.47
Detector-3	78.54	**87.46**	82.76
Detector-4	78.39	80.47	79.42
Detector-5	**82.04**	86.10	**84.02**

Table 6.10: Comparison results of the five detectors on Video 3

Video 3: 1232 hands	Recall(%)	Precision(%)	F-measure(%)
Detector-1	*55.93*	*29.22*	*38.39*
Detector-2	81.33	84.27	82.77
Detector-3	78.65	84.19	81.33
Detector-4	79.14	**90.11**	84.27
Detector-5	**81.66**	89.50	**85.40**

Table 6.11: Comparison results of the five detectors on Video 4

Video 4: 1512 hands	Recall(%)	Precision(%)	F-measure(%)
Detector-1	*41.73*	*32.98*	*36.84*
Detector-2	73.48	79.02	76.15
Detector-3	64.29	88.69	74.54
Detector-4	77.45	91.06	83.71
Detector-5	**78.57**	**94.96**	**86.00**

In Fig. 6.7, the red rectangle represents the face, and the green and blue rectangles represent two different hands respectively.

6.2.4 Analysis and discussion

First we analyze the performance of face detectors. From Tabs. 6.3 and 6.4, we can see that the face detector trained by Modest AdaBoost obtains the best recall. While the face detector trained by our Penalized AdaBoost achieves the best precision and F-measure. From Tabs. 6.5 and 6.6, we notice that the face detector trained by our Penalized AdaBoost overall outperform those trained by Gentle and Modest AdaBoost. Furthermore, the precision of face detector trained by Modest AdaBoost is worst in all four test data sets. From Figs.

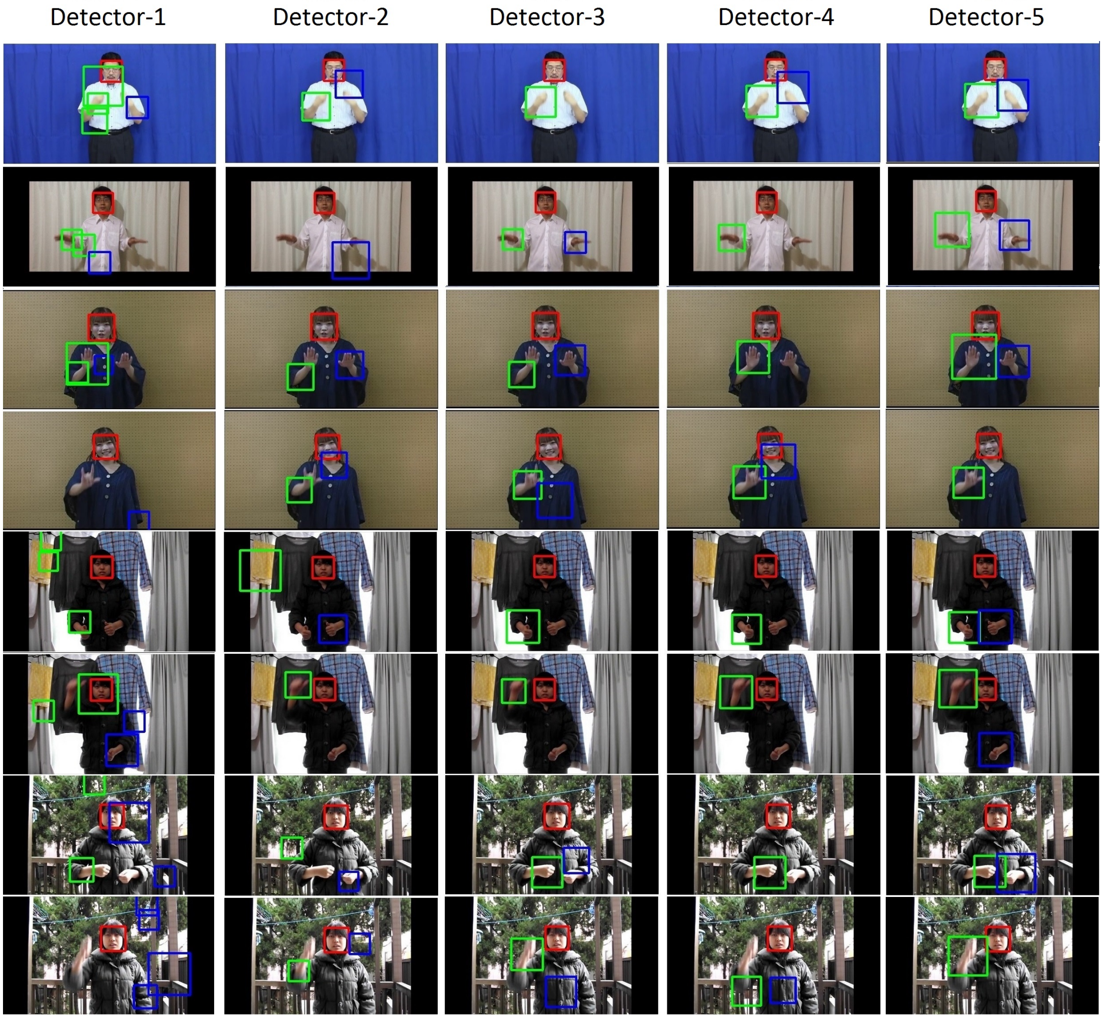


Figure 6.7: Examples of test results in hand detection

6.2, 6.3, 6.4, and 6.5, we can conclude that the face detector trained by our Penalized AdaBoost not only improves the detection rate, but also reduces the false positive rate significantly when compared with those trained by Gentle and Modest AdaBoost.

Secondly, we discuss the performance of hand detectors. From Fig. 6.6 and Tab. 6.7, we notice that hand detectors based on the proposed background-masking (Detectors 2, 3, 4, and 5) not only reduce a lot training time, but also suppress the false positive rate considerably [6]. Furthermore, the hand detector trained by Penalized AdaBoost achieves a lowest false positive rate [6].

From Tabs. 6.8, 6.9, 6.10, and 6.11, we find that Detector 1 which is based on Haar-like features and Gentle AdaBoost has the worst performance. When comparing Detectors 1 and 2, we notice that our proposed background-masking improves the recall and precision significantly. Nevertheless, from Tabs. 6.9 and 6.11, we find that background-masking performs not so well because of the skin-color-like background and illumination change [6]. Comparing Detectors 2 and 3, we can see that the motion filter improves the precision effectively even in videos have skin-color-like backgrounds [6]. We also find that the hand detector trained by Penalized AdaBoost overall outperforms those trained by Gentle and Modest AdaBoost by comparing Detectors 3, 4, and 5. From Fig. 6.7, we conclude that our proposed hand detection system improves the recall and precision considerably. Furthermore, it can extract the hand patches in proper size [6].

In general, our proposed AdaBoost variant Penalized AdaBoost improves the classification performance significantly in both face and hand detection. Moreover, our proposed hand detection system which integrates Haar-like features, background-masking, motion filtering, and Penalized AdaBoost achieves the best performance [6].

6.3 For Parameterized AdaBoost

This section evaluates the performance of our proposed Parameterized AdaBoost by comparing it with Real AdaBoost.

6.3.1 Experiments setting

We compare Parameterized and Real AdaBoost on totally 18 binary classification data sets from UCI repository [91]. Here we use CART-2 as weak classifiers for both Real and Parameterized AdaBoost. For each data set, we train the classifiers using 2/3 of its data, and test the trained classifier using 1/3 of its data. The training of classifiers terminates when the training error reaches to 0. In this experiments, we also use Matlab AdaBoost Toolbox [92]. To evaluate the training errors and test errors more fairly, we use 3-fold cross-validation.

6.3.2 Results evaluation

Table 6.12 summarizes the comparison results of Real and Parameterized AdaBoost on 18 data sets [8]. In Tab. 6.12, Convergence-Cost denote the numbers of iterations until the training errors converge to 0 or a constant value, Test-Error denotes the classification errors of the converged strong classifiers in test subsets. In this table, RAB and PAB denote Real AdaBoost and Parameterized AdaBoost respectively, and the bold numbers represent the best performance.

Table 6.12: Comparison results on 18 data sets

	Convergence-Cost		Test-Error	
Data sets:18	RAB	PAB	RAB	PAB
Bankruptcy	5	5	0.0343	0.0343
Banknote	11	**9**	0.01093	**0.0066**
BreastCancer(WDBC)	9	**7**	0.04223	**0.0422**
Fertility	8	**7**	0.21953	**0.21003**
Glass	5	**4**	0.05163	**0.03757**
German Numeric	410	**291**	**0.27603**	0.27803
Haberman's Survival	230	**173**	0.29737	**0.29083**
Heart Disease	29	**24**	**0.23703**	0.24447
Ionosphere	**9**	10	**0.1111**	0.11396
Indian Liver	106	**95**	**0.28843**	0.2936
Diabetes	155	**131**	0.29817	**0.27993**
Planning Relax	24	**22**	**0.35727**	0.36283
Ringnorm	116	**98**	**0.02973**	0.032
Spambase	520	**376**	0.0563	**0.05346**
Sonar	**8**	10	0.21663	**0.21166**
Twonorm	121	**108**	**0.03147**	0.03553
Wine Quality	40	**38**	0.00523	**0.00507**
Wisconsin Prognostic	12	12	0.2677	**0.26766**
Sum	1818	**1420**	2.80135	**2.76773**
Comparison	21.9% improved		1.2% improved	

6.3.3 Analysis and discussion

From Table 6.12, we find that our proposed Parameterized AdaBoost improves the training speed significantly when compared with Real AdaBoost. Moreover, the test errors of Parameterized AdaBoost are better than, or similar to those of Real AdaBoost in almost data sets [8].

Chapter 7

Comparison of AdaBoost Variants

In this chapter, we compare our proposed AdaBoost variants Parameterized AdaBoost, Margin-pruning Boost, and Penalized AdaBoost with three traditional ones (Real, Gentle, and Modest AdaBoost). Differently from other comparison studies, we compare these variants by analyzing the classification margins mathematically.

7.1 Statistical analysis

7.1.1 Real AdaBoost vs. Gentle AdaBoost

In Real AdaBoost, the upper bound of training error is optimized directly in each iteration. In contrast, Gentle AdaBoost optimizes the weighted least square error iteratively. Since Real AdaBoost decreases the training error exponentially, its training error converges faster than that of Gentle AdaBoost in most cases. On the other hand, Gentle AdaBoost are trying to reduce the variance of its weak hypotheses by minimizing the weighted least square error at each round. Therefore, it is more stable and robust than Real AdaBoost in terms of the generalization errors [9].

7.1.2 Real AdaBoost vs. Parameterized AdaBoost

Parameterized AdaBoost is proposed to speed up the training process of Real AdaBoost. It calculates the weak hypotheses the same as Real AdaBoost, but updates the weights of instances differently. In the boosting process, there are some instances whose margins are near 0. These instances could easily increase the training error if they are misclassified in the next iteration. on the contrary, they could decrease the training error significantly if they are correctly classified at the next round. In order to correctly classify instances whose margins are near 0 in future loops, Parameterized AdaBoost focuses more on these instances by introducing a parameter α. From (5.5), we notice that $\alpha \times |y_i F_t(x_i)|$ decreases more weight for instances whose margins are far from 0 and less weight for instances whose margins are near 0. Thereby, instances whose margins are near 0 in Parameterized AdaBoost can obtain more weight than those in Real AdaBoost. Therefore, Parameterized AdaBoost can converge its training error faster than Real AdaBoost. However, Parameterized AdaBoost is more prone to overfitting than Real AdaBoost because it focuses more on instances whose margins are near 0 at the cost of focusing less on instances with minimal margins. Nevertheless, experimental results in Chapter 6 shows that Parameterized AdaBoost can perform very similarly to, or slightly better than Real AdaBoost with respect to the generalization errors if it uses stump decision trees as weak classifiers because stump decision trees are more resistant to overfitting than CART with a lot of inner nodes [9].

7.1.3 Gentle AdaBoost vs. Modest AdaBoost

In order to reduce the generalization error, Modest AdaBoost adopts an inverted weight distribution to stand out weak hypotheses which can correctly classify instances with small margins. However, we find that the performance of Modest AdaBoost is not stable because its accuracy drops sharply in some data sets. Next we will give an example to illustrate the reason. When $W^j_{t+} > W^j_{t-}$, if $W^j_{t+}(1 - \overline{W}^j_{t+}) > W^j_{t-}(1 - \overline{W}^j_{t-})$ also holds, the factor $(1 - \overline{W}^j_{t+})$ gives higher prediction confidence to the weak hypotheses which can correctly classify small-margin instances. Similarly, the factor $(1 - \overline{W}^j_{t-})$ decreases the prediction confidence of weak hypotheses that misclassify small-margin instances. Under this circumstance, Modest AdaBoost can outperform Gentle AdaBoost with respect to the generalization errors. Nevertheless, if $W^j_{t+}(1 - \overline{W}^j_{t+}) < W^j_{t-}(1 - \overline{W}^j_{t-})$ holds in the case $W^j_{t+} > W^j_{t-}$, the sign of the weak hypothesis will go to opposite. Thus, the factor $(1 - \overline{W}^j_{t-})$ will decrease the prediction confidence of weak hypotheses which correctly classify small-margin instances. At the same time, the factor $(1 - \overline{W}^j_{t+})$ will increase the prediction confidence of weak hypotheses that misclassify small-margin instances. If this happens, Modest AdaBoost will lead to far worse generalization errors when compared with Gentle AdaBoost [9].

7.1.4 Gentle AdaBoost vs. Margin-pruning Boost

Margin-pruning Boost is proposed to solve the overfitting problem of Gentle AdaBoost. In Gentle AdaBoost, the weights of misclassified instances are increased and the weights of correctly classified instances are decreased iteratively. This type of weight updating encourages Gentle AdaBoost to focus more on difficult-to-classify instances. However, it could easily lead to the classifier distortion because the performance of the final strong classifier is degraded by these difficult-to-classify instances especially when these instances are noise data. To solve this problem, Margin-pruning Boost introduces a threshold to filter instances whose weights are too large, and resets their weights to be 1 [85].

Margin-pruning Boost is effective at reducing the influence from noise-like instances in the early training phase. Nevertheless, its performance drops as the number of iteration increases because the weights of instances filtered by the thresholding become smaller and smaller. Especially in the late training phase, the weights of the filtered instances are probably reduced smaller than 1. Thereby, resetting their weights to be 1 will increase the influence of these noise-like instances [9].

7.1.5 Margin-pruning Boost vs. Penalized AdaBoost

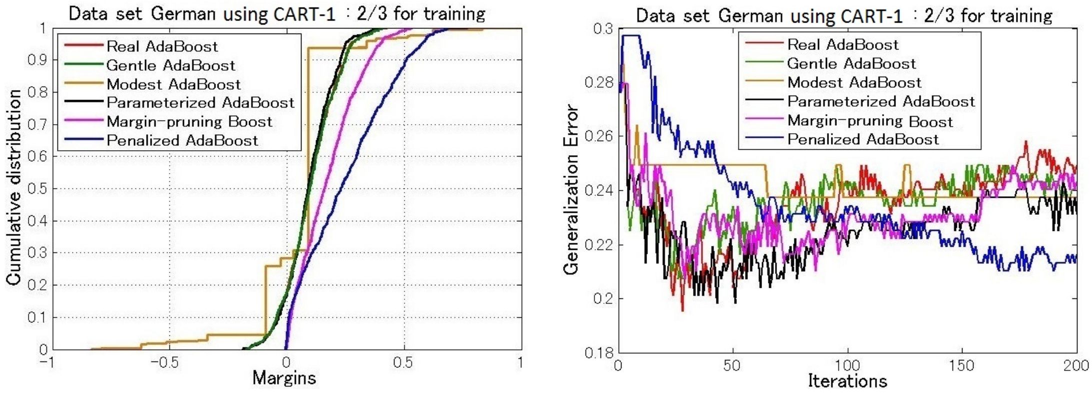
Penalized AdaBoost is different from Margin-pruning Boost in the following two respects.

- **(1)** Penalized AdaBoost utilizes a margin feedback factor to give higher prediction confidence to weak hypotheses which can correctly classify small-margin instances. From (3.4), (3.5), (3.6) and (3.7), we notice that the factors $(1 - M_{t-}^{j})$ and $(1 - M_{t+}^{j})$ are proportional to the margins of training data. They are computed from the sum of margin feedback factors of misclassified instances in each iteration. Therefore, misclassifying small-margin instances results in small $(1 - M_{t-}^{j})$ and $(1 - M_{t+}^{j})$. In this case, the prediction confidence of weak hypotheses misclassifying small-margin instances will be decreased. Since Penalized AdaBoost highlights more competent weak hypotheses, it is more robust than Margin-pruning Boost. Here we notice that Modest AdaBoost is successful in standing out more competent weak hypotheses in some cases, but it fails in other cases. By contrast, Penalized AdaBoost can always emphasize more competent weak hypotheses. So it is more stable than Modest AdaBoost [9].

- **(2)** Penalized AdaBoost solves the problem of Margin-pruning Boost by adopting a more adaptive thresholding technique. In Penalized AdaBoost, only instances whose weights exceed the threshold and whose margins are negative will be reset. This process makes sure that the reset weights are smaller than the original ones under all circumstances. For noise-like instances, Penalized AdaBoost does not exclude them completely since they are not noise necessarily. However, it keeps the weights of these noise-like instances small to reduce their influence on the final strong classifier. Thus, Penalized AdaBoost can achieve better generalization errors than Margin-pruning Boost [9].

7.2 Margin distributions

Differently from other comparison work, here we compare Real AdaBoost, Gentle AdaBoost, Modest AdaBoost, Parameterized AdaBoost, Margin-pruning Boost, and Penalized AdaBoost by analyzing their cumulative margin distributions. In this comparison, we utilize three types of CART as weak classifiers to evaluate the six variants. The three types of CART include CART-1 (CART with one inner node), CART-2 (CART with two inner nodes), and CART-3 (CART with three inner nodes). For each data set, we retain 2/3 of its data for training. Figure 7.1 (a) shows the margin distributions of the six variants using CART-1 in a data set German at iteration 200. Figure 7.1 (b) shows the generalization errors of the six variants in the same data set. In Fig. 7.1 (a), we find that Penalized AdaBoost enlarges the whole margin distribution more than other variants. Thus, it can obtain the best generalization error in Fig. 7.1 (b). Here we also notice that the margin curve of Modest AdaBoost is not as smooth as those of other variants. That may explain why its generalization error in Fig. 7.1 (b) is not decreased gradually [9].

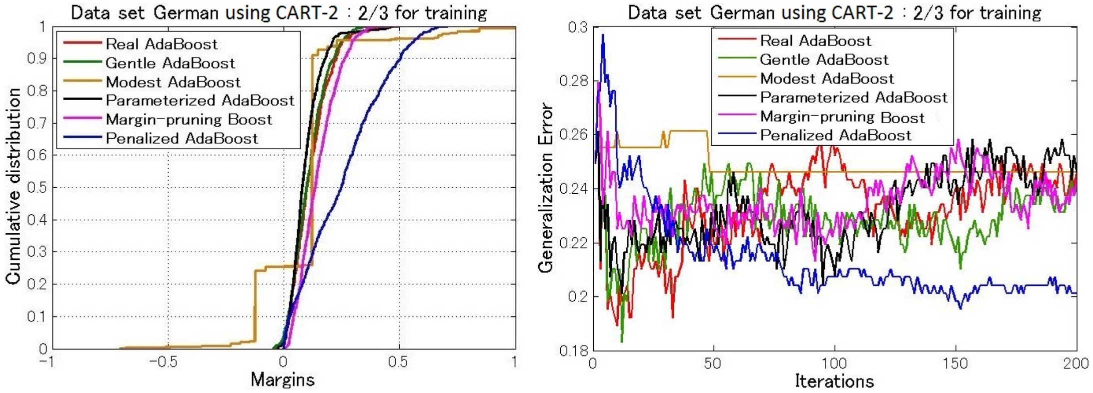


(a) Margin distributions using CART-1 **(b)** Generalization errors using CART-1

Figure 7.1: Margin distributions and generalization errors based on CART-1

We also compare margin distributions of the six variants based on CART-2 in the same data set. Figure 7.2 (a) shows the result at iteration 200. While Fig. 7.2 (b) shows the generalization errors of the six variants using CART-2. From Fig. 7.2, we can see that the generalization abilities of the six variants are consistent with their performance on the margins [9].

Then we evaluate margin distributions of the six variants based on CART-3 in the same data set. Figure 7.3 (a), (b), and (c) show the margin curves at iteration 10, 100, and 1000 respectively. From the three figures, we notice that the margins of training data are enlarged gradually when the number of iterations increases. Moreover, we can see that Margin-pruning Boost outperforms Gentle AdaBoost in Figs. 7.3 (a) and 7.3 (b). Nevertheless, it performs worse than Gentle AdaBoost

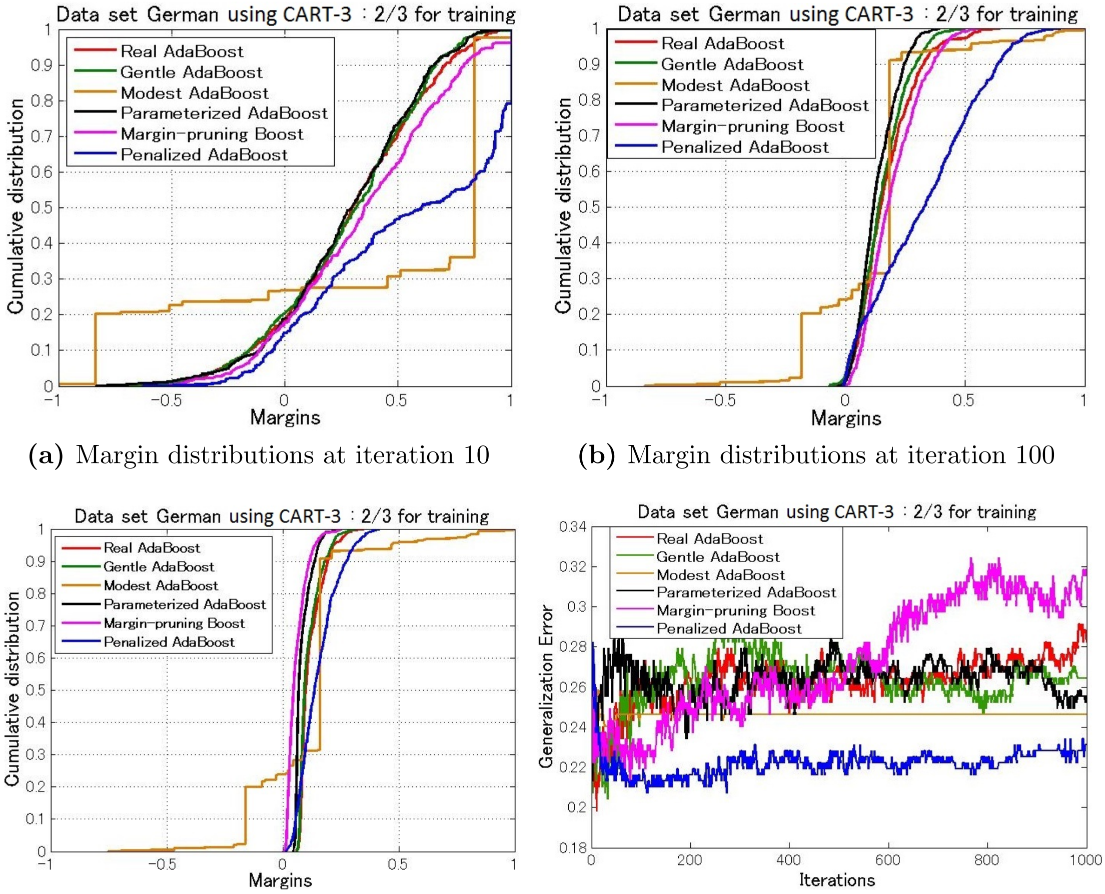


(a) Margin distributions using CART-2 **(b)** Generalization errors using CART-2

Figure 7.2: Margin distributions and generalization errors based on CART-2

in Fig. 7.3 (c). This phenomenon shows that the generalization ability of Margin-pruning Boost drops as the number of iteration increases. By contrast, Penalized AdaBoost outperforms other variants in most cases. Therefore, it is more robust and stable than others. Figure 7.3 (d) shows the generalization errors of the six variants based on CART-3. In Fig. 7.3 (d), Margin-pruning Boost obtains better generalization errors than Gentle AdaBoost before iteration 500. Unfortunately, it results in severe overfitting problem after iteration 500 [9].

Finally, we compare magrin distributions of the six variants in other data sets. Figures 7.4 and 7.5 show the comparison results. From the two figures, we find that Penalized AdaBoost overall outperforms other five variants on enlarging the margins of training data. The performance of Real and Gentle AdaBoost is very similar, and Parameterized AdaBoost performs slightly worse than Real AdaBoost when it utilizes CART-2 and CART-3 as its weak classifiers. Margin-pruning Boost performs worse than Gentle AdaBoost as the number of iterations increases. Comparing with other variants, the margin curves of Modest AdaBoost are not smooth, which may result in an unstable performance on its generalization errors [9].



(a) Margin distributions at iteration 10

(b) Margin distributions at iteration 100

(c) Margin distributions at iteration 1000

(d) Generalization errors based on CART-3

Figure 7.3: Margin distributions and generalization errors based on CART-3

7.3 Generalization abilities

In this section, we compare the generalization abilities of the six AdaBoost variants by experiments. We totally use 25 binary classification data sets from UCI [91]. In each data set, we utilize Matlab AdaBoost Toolbox [92] and 3-fold cross-validation to measure the generalization errors of the six variants. First we evaluate the generalization abilities of the six variants based on CART-1. Table 7.1 shows the comparison results at iteration 200 [9]. Tables 7.2 and 7.3 show the generalization errors of the six variants based on CART-1 at iteration 500 and 800 separately [9]. We also compare the six variants based on CART-2 and CART-

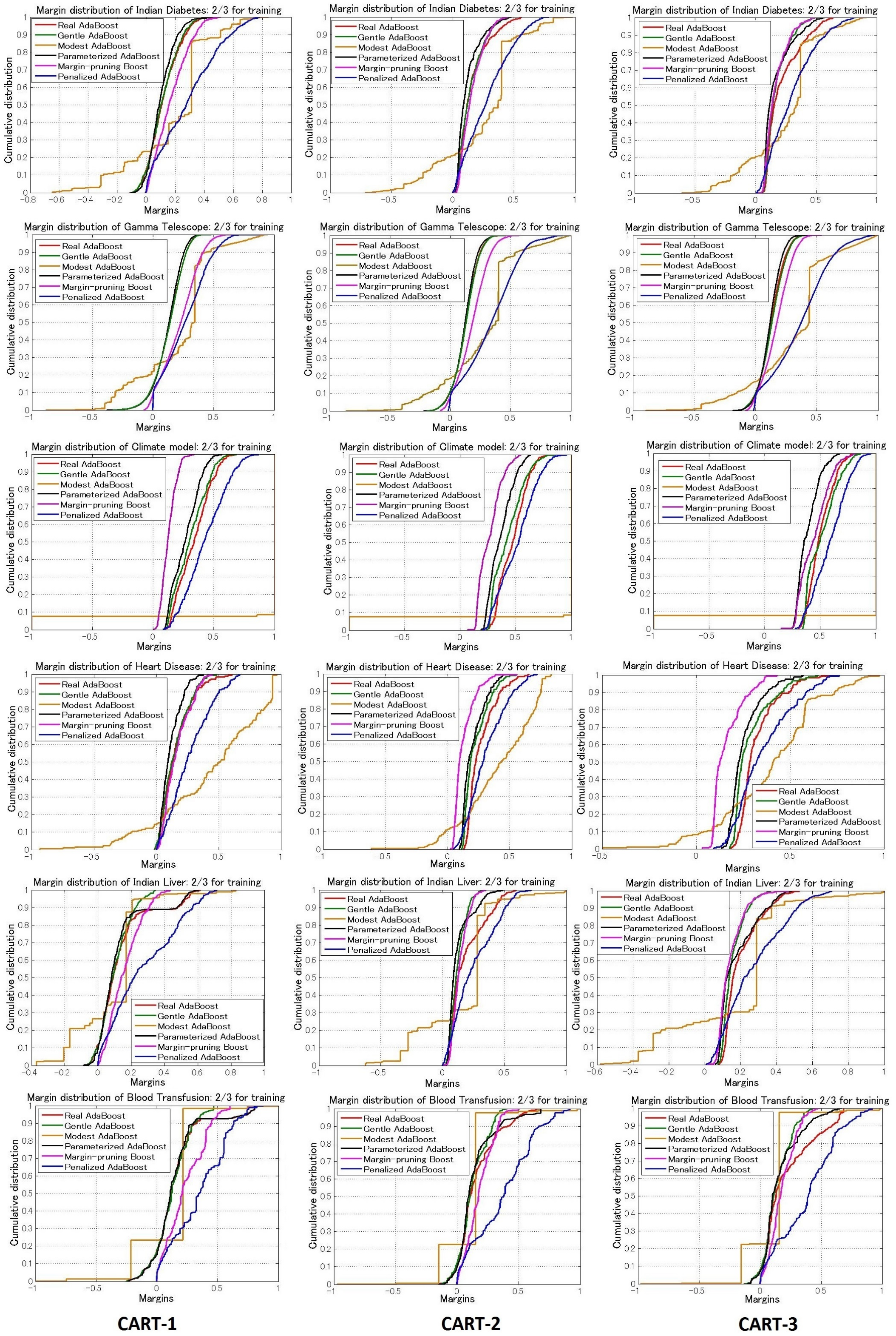


Figure 7.4: Margin distributions at iteration 200 - 1

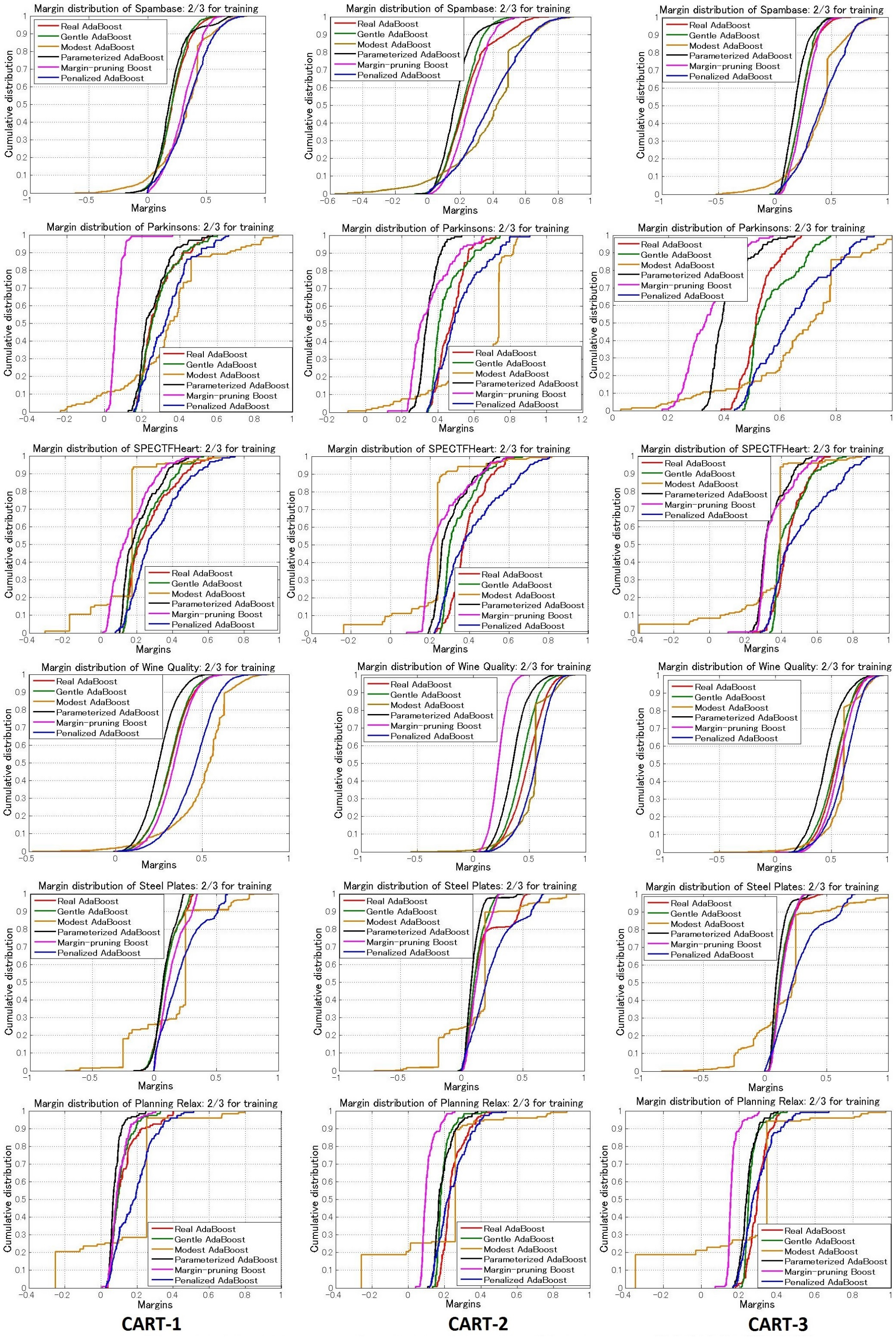


Figure 7.5: Margin distributions at iteration 200 - 2

3 with respect to the generalization errors. The comparison results based on CART-2 are shown in Tab. 7.4 [9], and the comparison results based on CART-3 are shown in Tab. 7.5 [9].

Table 7.1: Comparison results at iteration 200 based on CART-1

Data sets:25	RAB	GAB	MAB	PAAB	MPB	PAB
Australian	0.1536	0.1435	0.1435	0.1449	0.1391	**0.1304**
Blood Transfusion	0.2392	0.2312	0.2352	0.2392	**0.2218**	0.2298
Banknote	0.0029	0.0022	0.0372	**0.0015**	0.0058	0.0080
Breast Cancer	0.0281	**0.0246**	0.0422	0.0299	0.0669	0.0281
Banana	0.2732	0.2713	0.2804	0.2672	**0.2526**	0.2809
Climate model	0.0704	**0.0593**	0.0963	0.0759	0.0704	0.0630
Gamma Telescope	0.1525	0.1546	0.2297	0.1489	**0.1454**	0.1548
German	0.2560	0.2580	0.2950	0.2490	0.2470	**0.2450**
HeartDisease	0.2259	0.2333	0.1704	0.2185	0.2000	**0.1630**
Hepatitis	0.2200	0.2130	0.2515	0.2065	0.2129	**0.1869**
ImageSegment	0.0061	**0.0045**	0.0076	0.0061	0.0106	0.0061
IndianDiabetes	0.2448	0.2578	0.2383	0.2539	0.2318	**0.2253**
IndianLiver	0.3074	0.3005	0.3230	0.3092	**0.2850**	0.3126
Parkinsons	**0.0872**	**0.0872**	0.1692	0.0923	0.0974	0.0923
PlanningRelax	0.4286	0.4012	0.3519	0.4448	0.4175	**0.3186**
Ringnorm	0.0269	**0.0268**	0.0470	0.0293	0.0277	0.0509
Sonar	**0.1445**	0.1589	0.1637	0.1540	0.2117	0.1734
Spambase	0.0580	0.0546	0.0895	0.0554	**0.0528**	0.0617
SPECTFHeart	0.2097	0.2285	0.2472	0.2172	0.2584	**0.1985**
Splice	0.0675	0.0675	0.0919	0.0685	**0.0612**	0.0639
SteelPlates	**0.2277**	0.2324	0.2751	0.2344	0.2318	0.2421
Twonorm	0.0303	0.0305	0.0315	0.0324	**0.0292**	0.0304
Waveform	0.0993	0.0975	0.0962	0.0996	0.0917	**0.0853**
WBPC	0.2475	0.2727	0.3030	0.2879	0.3081	**0.2323**
WineQuality	0.0055	0.0054	0.0240	0.0062	**0.0052**	0.0054
Sum	3.8128	3.8170	4.2405	3.8727	3.8820	**3.5887**
VS.GAB	-0.0042	0.0000	0.4235	0.0557	0.0650	**-0.2283**
No.Best	3	5	0	1	8	**9**
No.To.GAB	11	–	5	9	**15**	13

In Tabs. 7.1, 7.2, 7.3, 7.4, and 7.5, RAB, GAB, MAB, PAAB, MPB, PAB

Table 7.2: Comparison results at iteration 500 based on CART-1

Data sets:25	RAB	GAB	MAB	PAAB	MPB	PAB
Australian	0.1681	0.1638	0.1435	0.1638	0.1377	**0.1333**
Blood Transfusion	0.2446	0.2540	0.2352	0.2500	0.2272	**0.2258**
Banknote	0.0029	0.0036	0.0372	**0.0022**	0.0058	0.0058
BreastCancer	0.0281	**0.0246**	0.0369	0.0264	0.0756	**0.0246**
Banana	0.2734	0.2725	0.2804	0.2662	**0.2547**	0.2738
Climatemodel	0.0704	0.0648	0.0963	0.0741	0.0722	**0.0593**
Gamma Telescope	0.1522	0.1519	0.2292	0.1479	**0.1424**	0.1473
German	0.2700	0.2730	0.2950	0.2710	0.2490	**0.2430**
HeartDisease	0.2481	0.2481	**0.1704**	0.2333	0.2000	0.1852
Hepatitis	0.2262	0.2130	0.2515	0.2132	0.2129	**0.2128**
ImageSegment	0.0045	0.0045	0.0045	**0.0030**	0.0061	**0.0030**
IndianDiabetes	0.2487	0.2526	0.2383	0.2565	0.2409	**0.2266**
IndianLiver	0.3005	0.3074	0.3558	0.2971	**0.2953**	0.3057
Parkinsons	**0.0821**	0.0974	0.1641	0.0923	0.0974	0.0872
PlanningRelax	0.4342	0.4397	0.3519	0.4064	0.4175	**0.3462**
Ringnorm	0.0264	**0.0249**	0.0309	0.0280	0.0251	0.0318
Sonar	**0.1446**	0.1638	0.1589	0.1684	0.2548	0.1734
Spambase	0.0593	0.0574	0.0895	0.0578	**0.0517**	0.0572
SPECTFHeart	0.2097	0.2285	0.2584	0.2060	0.2772	**0.2022**
Splice	0.0726	0.0746	0.0919	0.0752	0.0622	**0.0605**
SteelPlates	0.2318	0.2293	0.2777	0.2339	**0.2267**	0.2334
Twonorm	0.0291	0.0296	0.0286	0.0326	0.0296	**0.0276**
Waveform	0.0990	0.0984	0.0962	0.1038	0.0950	**0.0881**
WBPC	0.2475	0.2828	0.3030	0.2424	0.3081	**0.2374**
WineQuality	0.0060	0.0054	0.0231	0.0054	0.0060	**0.0051**
Sum	3.8800	3.9656	4.2484	3.8569	3.9711	**3.5963**
VS.GAB	-0.0856	0.0000	0.2828	-0.1087	0.0055	**-0.3693**
No.Best	2	2	1	2	5	**15**
No.To.GAB	14	–	9	14	16	**20**

Table 7.3: Comparison results at iteration 800 based on CART-1

Data sets:25	RAB	GAB	MAB	PAAB	MPB	PAB
Australian	0.1754	0.1696	0.1435	0.1899	0.1362	**0.1348**
Blood Transfusion	0.2487	0.2473	0.2379	0.2527	0.2272	**0.2218**
Banknote	0.0036	0.0044	0.0372	**0.0022**	0.0058	0.0036
BreastCancer	**0.0246**	0.0281	0.0369	**0.0246**	0.0774	**0.0246**
Banana	0.2740	0.2753	0.2804	0.2687	**0.2530**	0.2709
Climatemodel	0.0759	0.0685	0.0963	0.0722	0.0722	**0.0611**
Gamma Telescope	0.1530	0.1531	0.2290	0.1485	**0.1425**	0.1448
German	0.2800	0.2770	0.2950	0.2780	0.2550	**0.2430**
HeartDisease	0.2519	0.2333	**0.1704**	0.2519	0.2000	0.2111
Hepatitis	0.2197	0.2195	0.2646	0.2068	0.2129	**0.2000**
ImageSegment	0.0045	0.0045	0.0045	**0.0030**	0.0061	**0.0030**
IndianDiabetes	0.2513	0.2513	0.2396	0.2591	0.2396	**0.2305**
IndianLiver	0.2971	0.3005	0.3592	0.2988	0.3022	**0.2936**
Parkinsons	0.0872	0.1026	0.1641	**0.0821**	0.0974	0.0923
PlanningRelax	0.4669	0.4287	**0.3519**	0.4230	0.4175	0.3737
Ringnorm	0.0258	**0.0235**	0.0273	0.0288	0.0262	0.0288
Sonar	**0.1493**	0.1542	0.1638	0.1732	0.2500	0.1685
Spambase	0.0593	0.0563	0.0880	0.0572	**0.0517**	0.0561
SPECTFHeart	0.2172	0.2247	0.2697	**0.2060**	0.2734	0.2135
Splice	0.0766	0.0772	0.0919	0.0772	0.0642	**0.0598**
SteelPlates	0.2303	0.2303	0.2736	0.2349	**0.2262**	0.2318
Twonorm	0.0301	0.0300	0.0286	0.0315	0.0309	**0.0269**
Waveform	0.0987	0.0981	0.0962	0.1065	0.0956	**0.0902**
WBPC	0.2424	0.2828	0.2980	0.2475	0.3081	**0.2273**
WineQuality	0.0057	0.0058	0.0235	0.0055	0.0072	**0.0046**
Sum	3.9492	3.9466	4.2711	3.9298	3.9785	**3.6163**
VS.GAB	0.0026	0.0000	0.3245	-0.0168	0.0319	**-0.3303**
No.Best	2	1	2	5	4	**14**
No.To.GAB	14	–	8	13	14	**22**

Table 7.4: Comparison results at iteration 200 based on CART-2

Data sets:25	RAB	GAB	MAB	PAAB	MPB	PAB
Australian	0.1449	0.1478	0.1420	0.1580	0.1420	**0.1246**
Blood Transfusion	0.2581	0.2540	0.2366	0.2608	0.2419	**0.2204**
Banknote	0.0051	**0.0022**	0.0058	**0.0022**	0.0044	0.0029
BreastCancer	0.0264	**0.0229**	0.0387	0.0299	0.0316	0.0246
Banana	0.1021	**0.1002**	0.2430	0.1006	0.1045	0.1479
Climatemodel	0.0574	0.0574	0.0944	0.0593	0.1019	**0.0519**
Gamma Telescope	0.1301	0.1317	0.2006	**0.1295**	0.1320	0.1426
German	0.2710	0.2690	0.2890	0.2850	0.2580	**0.2380**
HeartDisease	0.2667	**0.2000**	0.2074	0.2556	0.2889	0.2037
Hepatitis	0.1934	0.2001	0.2320	**0.1675**	0.2709	0.1934
ImageSegment	0.0091	0.0061	**0.0045**	**0.0045**	0.0091	**0.0045**
IndianDiabetes	0.2591	0.2682	**0.2357**	0.2839	0.2500	0.2370
IndianLiver	0.2953	0.2815	0.3592	0.3005	**0.2694**	0.3057
Parkinsons	0.0667	**0.0564**	0.1385	0.0667	0.0974	0.0769
PlanningRelax	0.4117	0.3791	**0.3352**	0.3731	0.4392	0.3573
Ringnorm	0.0288	0.0284	0.0312	0.0311	**0.0276**	0.0343
Sonar	0.1396	0.1397	0.1493	0.1443	**0.1395**	0.1491
Spambase	0.0511	0.0506	0.0798	0.0550	**0.0476**	0.0546
SPECTFHeart	**0.2060**	0.2135	0.2472	**0.2060**	0.2434	0.2135
Splice	0.0431	0.0408	**0.0368**	0.0481	0.0408	0.0381
SteelPlates	0.2102	0.2020	0.2694	0.2184	**0.1999**	0.2123
Twonorm	0.0300	0.0303	0.0297	0.0342	0.0288	**0.0270**
Waveform	0.0908	0.1008	0.0920	0.1032	0.0935	**0.0860**
WBPC	0.2424	0.2424	0.3030	0.2374	0.2525	**0.2222**
WineQuality	0.0051	**0.0038**	0.0149	0.0042	0.0060	0.0045
Sum	3.5442	3.4289	4.0159	3.5590	3.7208	**3.3730**
VS.GAB	0.1153	0.0000	0.5870	0.1301	0.2919	**-0.0559**
No.Best	1	6	4	5	5	**8**
No.To.GAB	10	–	8	7	12	**13**

Table 7.5: Comparison results at iteration 200 based on CART-3

Data sets:25	RAB	GAB	MAB	PAAB	MPB	PAB
Australian	0.1478	0.1536	0.1333	0.1594	0.1522	**0.1203**
Blood Transfusion	0.2863	0.2715	0.2392	0.2984	0.2527	**0.2245**
Banknote	0.0036	0.0022	0.0029	**0.0015**	0.0051	0.0022
BreastCancer	0.0334	0.0281	**0.0246**	0.0281	0.0281	0.0281
Banana	0.1040	0.1028	0.2142	0.1047	**0.1002**	0.1132
Climatemodel	0.0611	0.0519	0.0944	0.0593	0.0722	**0.0500**
Gamma Telescope	0.1300	0.1305	0.1770	0.1316	**0.1278**	0.1360
German	0.2870	0.2660	0.2780	0.2920	0.2750	**0.2440**
HeartDisease	0.2074	0.2185	0.2074	0.2074	0.2593	**0.2000**
Hepatitis	**0.1932**	0.1933	0.2252	0.1998	0.1934	0.2127
ImageSegment	**0.0015**	**0.0015**	0.0030	0.0030	0.0061	0.0030
IndianDiabetes	0.2513	0.2552	0.2357	0.2695	0.2643	**0.2344**
IndianLiver	**0.2781**	**0.2781**	0.3437	0.3092	0.2867	0.2971
Parkinsons	0.0718	**0.0615**	0.1026	0.0821	0.0769	0.0769
PlanningRelax	0.3514	0.3844	0.3518	**0.3459**	0.3899	0.3682
Ringnorm	0.0277	**0.0266**	0.0296	0.0295	0.0296	0.0309
Sonar	**0.1202**	0.1395	0.1637	0.1636	0.1347	0.1349
Spambase	0.0528	**0.0454**	0.0728	0.0519	0.0476	0.0500
SPECTFHeart	0.2210	0.2022	0.2472	**0.1948**	0.2285	**0.1948**
Splice	0.0321	0.0361	**0.0294**	0.0428	0.0428	0.0308
SteelPlates	0.2030	0.2020	0.2653	0.2138	0.2035	**0.1999**
Twonorm	0.0319	0.0301	0.0288	0.0299	0.0304	**0.0265**
Waveform	0.1029	0.0972	0.0890	0.1053	0.1038	**0.0872**
WBPC	0.2323	**0.2172**	0.3030	0.2525	0.2273	**0.2172**
WineQuality	0.0051	**0.0038**	0.0125	0.0048	0.0042	**0.0038**
Sum	3.4369	3.3992	3.8743	3.5808	3.5423	**3.2866**
VS.GAB	0.0377	0.0000	0.4751	0.1816	0.1431	**-0.1126**
No.Best	4	7	2	3	2	**12**
No.To.GAB	10	–	9	6	6	**17**

represent Real AdaBoost, Gentle AdaBoost, Modest AdaBoost, Parameterized AdaBoost, Margin-pruning Boost, and Penalized AdaBoost respectively. While VS.GAB denotes the residues that the sum of generalization errors of other variants subtracts that of Gentle AdaBoost [9]. In Tabs. 7.1, 7.2, 7.3, 7.4, and 7.5, the bold values show the best generalization error and No.Best denotes the number of best generalization errors. No.To.GAB means the number of data sets on which Gentle AdaBoost are outperformed by other variants [9]. From Tabs. 7.1, 7.2, and 7.3, we notice that Real, Gentle, and Parameterized AdaBoost perform similarly when they utilize CART-1 as their weak classifiers. By contrast, Modest AdaBoost mostly performs worse than other variants. Furthermore, its generalization errors are hardly reduced even as the number of loops increases. When we compare the No.Best of Margin-pruning Boost in Tabs. 7.1, 7.2, and 7.3, we can see that its performance drops sharply when the number of iterations increases. We also notice that Penalized AdaBoost completely outperforms other five variants on VS.GAB, No.Best, and No.To.GAB. From Tabs. 7.1, 7.4, and 7.5, we can see that the number of inner nodes of CART is important to the generalization errors. We also notice that Gentle AdaBoost is slightly better than Real AdaBoost especially using CART-2 and CART-3. On the other hand, the performance of Parameterized AdaBoost and Margin-pruning Boost drop sharply when using CART-2 and CART-3. This suggests that the two variants are more suitable for CART-1. We also see that Modest AdaBoost based on CART-2 or CART-3 performs better than that based on CART-1. This may suggest that Modest AdaBoost is more suitable for CART with many inner nodes. From all the five tables, we can conclude that the performance of Gentle and Penalized AdaBoost is not influenced by neither the number of inner nodes of CART or the number of iterations. However, Penalized AdaBoost is significantly better than Gentle AdaBoost under most circumstances [9].

Chapter 8

Conclusion

8.1 Observations

Hand and face detection is an essential prerequisite of gesture communication systems. Among current object detection methods, vision-based approaches are preferable because they are most natural and comfortable for people who are gesturing. As a vision-based object detection technique, Viola-Jones method which combines Haar-like features with AdaBoost has achieved substantial success in face detection. Nevertheless, it is difficult to detect hands precisely by Viola-Jones method because hands are highly deformable compared with faces.

Our research aims at improving the accuracy of Viola-Jones method. In our work, we first detect the face part because face detection is far more advanced than hand detection, and then use the face information to improve the false positive rate of the hand detector. In addition, we devise a new AdaBoost variant which we call Penalized AdaBoost to improve the classification performance in both face and hand detection. This dissertation describes our proposed system in details. For example, Chapter 3 explains our proposed AdaBoost variant Penalized AdaBoost. It aims at solving classifier-distortion of Gentle AdaBoost. Compared with Gentle AdaBoost, Penalized AdaBoost selects more competent weak learners in each iterations. At the same time, it restrains the weight increase of noise-like instances. Penalized AdaBoost enlarges the whole margin distribution more than other AdaBoost variants such as Real AdaBoost, Modest AdaBoost, and so on. Therefore, it is most robust among these compared variants.

Chapter 4 describes our proposed face and hand detection systems. The training processes of our face and hand detectors are based on Penalized AdaBoost. Moreover, we devise a new skin-color segmentation technique which we call background-masking to remove the background part of both training and test data. This technique not only reduces the number of training instances significantly, but also decreases the false positive rate of our trained hand detector. On the other hand, the face and hand detectors trained by Penalized AdaBoost are proved more robust and stable than those trained by other variants like Real, Gentle, and Modest AdaBoost.

Chapter 5 explains our other contribution Parameterized AdaBoost. It is an improvement of Real AdaBoost. Compared with Real AdaBoost, Parameterized AdaBoost focuses more on instances whose margins are near 0 and tries to correctly classify them in future loops. Parameterized AdaBoost can achieve a faster convergence of training error than Real AdaBoost. Furthermore, it proves that decreasing the sum of instance weights explicitly can speed up the training process. Because Parameterized AdaBoost improves the training error rather than the generalization error, we did not apply this method to our detection system.

Chapter 7 analyzes the generalization abilities of several different AdaBoost variants by comparing their margin distributions. This chapter provides a clue between the margin distributions of training data and generalization errors. Moreover, it shows a direction for improving generalization errors. On the other hand, Chapter 8 describes the process of feature extraction for gesture communication systems after face and hand detection. This chapter shows that gesture communication systems based on our proposed detection approach can achieve high accuracy using only hand images instead of sign language words as training data.

8.2 Limitation

In our research, background-masking is used for both training instances and test images. It suggests a skin-color segmentation based on face information. This process may lead to two limitations as we listed below:

- Background-masking fails when the faces in test images can not be detected. This could happen in the case that test images do not contain face parts, or test images have severe bad illumination in face area.
- Background masking fails when the illumination of face area is severely different from that of hand area because skin-color segmentation based on face information may totally exclude the hand area.

8.3 Future works

In future, we intend to apply our proposed detection approach to a sign language recognition system. Our future work can be described as follows:

- Collect sign language videos by different signers in different backgrounds. For example, collect 100 sign language words by Signer A in Background A, and then collect the same 100 sign language words by Signer B in Background B, and so on, finally use one scenario of collection as the standard data.
- Extract features for every frame of the standard data. These features include hand positions, face position, hand shape, the distance between face and hands.
- Train a temporal sign language recognition model based on the extracted features in the above step.
- Use other scenarios of collection except the standard data as test data, and then compare our recognition model with the current sign language recognition approaches such as HMM (Hidden Markov Model), ANNs (Artificial Neural Networks), and ART (Adaptive Resonance Theory).

Appendix A

Other features in hand detection

In this part, we explain other features used in hand detection such as LBP, SIFT, and HOG.

A.1 LBP

Local binary patterns (LBP) is a type of features which can clearly describe the local texture of images. It has been widely used in image classification and object detection [95]. A 8-neighbours LBP vector can be created by the following steps:

- First divide the test image into several cells, e.g., 32×32 pixels in each cell.

- For each cell, process all its pixels as follows: pick any pixel, compare its intensity with those of its 8 neighbours (left-top, left-middle, left-bottom, top, right-top, right-middle, right-bottom, bottom). If its intensity is greater than the neighbour's intensity, use 1 to replace the intensity of its neighbour; otherwise, use 0. Finally an 8-digit binary number shows the pixel intensities of its 8 neighbours. Then change the binary digit number into a decimal digit number.

- Compute the histogram of the decimal digit numbers obtained in above step for each cell. After we calculate the histograms for all cells, we can get the 8-neighbours LBP vector of the image.

A 16-neighbours LBP vector can be created similarly to the above steps. The calculation of LBP features is slightly shower than that of Haar-like features which we used in our work. Similarly to Haar-like features, LBP features are also not robust to background noise and hand shape change.

A.2 SIFT

Scale-invariant feature transform (SIFT) describes the local features of images by key points and the corresponding eigenvalues [96].

First SIFT creates a set of key points from the images of an object in different scales. Then for each key point, an eigenvalue which describes the neighbourhood information around the key point is computed. Finally, these key points and corresponding eigenvalues are used to identify the object in a test image. Since the extracted key points are extreme points, they can robustly describe the high-contrast regions of an image. SIFT is proved invariant to scale change, rotation, illumination change, affine transformation, and background noise [96].

In hand detection systems, SIFT features are more robust than LBP and Haar-like features. Nevertheless, the computation of SIFT features is rather time-consuming. This suggests that SIFT features may be not suitable for real time applications especially when the size of test images are large. On the other hand, it is difficult to distinguish hands from arms by SIFT features because the extracted key points from hands and arms are highly similar.

A.3 HOG

Histogram of Oriented Gradients (HOG) is a kind of feature descriptors which calculates occurrences of gradient orientation in localized portions of an image [97]. HOG has achieved considerable success in object detection especially in human detection [97].

The calculation of HOG features can be explained briefly as follows:

- Divide an image into several blocks.

- Divide an block into several small regions which are called cells.

- For each cell, compute gradient directions for all pixels within the cell, and then get the local histogram of these computed gradient directions.
- For each block, do normalization for all local histograms within the block. Finally the combination of all normalized local histograms is the HOG descriptor of the block.
- The HOG feature of an image consists of HOG descriptors of all blocks.

Because HOG features operate on localized small cells, they are invariant to geometric transformations such as affine transformation and orientation change. Moreover, the normalization for cells in one block makes HOG features invariant to illumination changes. HOG feature is suitable for detecting fixed shapes like pedestrians since it highlights edges and corners in an test image. Many researchers also utilized HOG features in hand detection systems. Nevertheless, the high computational cost of HOG features is still a problem because most hand detection systems are required to run in real-time.

Bibliography

[1] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 511–518, 2001. 1, 3, 14, 17

[2] M. Jebali, P. Dalle, and M. Jemni. Hmm-based method to overcome spatiotemporal sign language recognition issues. *International Conference on Electrical Engineering and Software Applications*, pages 1–6, 2013. 2

[3] S.H. Yu, C.L. Huang, S.C. Hsu, H.W. Lin, and H.W. Wang. Vision-based continuous sign language recognition using product hmm. *Asian Conference on Pattern Recognition*, pages 510–514, 2011. 2

[4] M. Maebatake, I. Suzuki, M. Nishida, Y. Horiuchi, and S. Kuroiwa. Sign language recognition based on position and movement using multi-stream hmm. *Second International Symposium on Universal Communication*, pages 478–481, 2008. 2

[5] S. Wu and H. Nagahashi. Real-time 2d hands detection and tracking for sign language recognition. *the 8th International Conference on System of Systems Engineering*, pages 40–45, 2013. 3, 4

[6] S. Wu and H. Nagahashi. Robust 2d hand detection for sign language recognition based on a novel boosting method. *IEICE Transactions on Information and Systems*, submitted, 2014. 3, 5, 35, 40, 42, 55, 58, 60, 62, 63

[7] S. Wu and H. Nagahashi. Penalized adaboost: Improving the generalization error of gentle adaboost through a margin distribution. *IEICE Transactions on Information and Systems*, 2014. 3, 5, 13, 52, 54

[8] S. Wu and H. Nagahashi. Parameterized adaboost: Introducing a parameter to speed up the training of real adaboost. *IEEE Signal Processing Letters*, 21(6):687–691, 2014. 5, 45, 47, 48, 50, 64, 65

[9] S. Wu and H. Nagahashi. Analysis of generalization ability for different adaboost variants based on classification and regression trees. *Journal of Electrical and Computer Science*, submitted, 2014. 5, 67, 68, 69, 70, 71, 72, 73, 76, 81

[10] Y. Freund and R.E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999. 7, 20

[11] R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999. 8

[12] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–374, 2000. 9

[13] A. Vezhnevets and V. Vezhnevets. Modest adaboost: Teaching adaboost to generalize better. *Graphicon*, 12(5):987–997, 2005. 10

[14] Y.Z.J. Thongkam, G. Xu, and F. Huang. Breast cancer survivability via adaboost algorithms. *Australian Workshop on Health Data and Knowledge Management*, pages 55–64, 2008. 10

[15] Y. Freund. An adaptive version of the boost by majority algorithm. *the 12th Annual Conference on Computational Learning Theory*, 37(3):102–113, 2000. 11

[16] R.A. Demiriz, K.P. Bennett, and J.S. Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(3):225–254, 2002. 11

[17] H. Li and C. Shen. Boosting the minimum margin: Lpboost vs. adaboost. *Digital Image Computing: Techniques and Applications (DICTA)*, pages 533–539, 2008. 11

[18] C. Domingo and O. Watanabe. Madaboost: A modification of adaboost. *the 13th Annual Conference on Computational Learning Theory*, pages 180–189, 2000. 11

[19] R. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4:633–648, 2003. 11

[20] W. Fan, S.J. Stolfo, J. Zhang, and P.K. Chan. Adacost: Misclassification cost-sensitive boosting. *the 16th International Conference on Machine Learning*, pages 97–105, 1999. 11

[21] M.V. Joshi, V. Kumar, and R.C. Agarwal. Evaluating boosting algorithms to classify rare classes: Comparison and improvements. *IEEE International Conference on Data Mining*, pages 257–264, 2001.

[22] Y. Sun, M.S. Kamel, A.K.C. Wong, and Y. Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[23] K.M. Ting. A comparative study of cost-sensitive boosting algorithms. *the 17th International Conference on Machine Learning*, 37(3):983–990, 2000. 11

[24] M. Seyedhosseini, A.R.C. Paiva, and T. Tasdizen. Fast adaboost training using weighted novelty selection. *International Conference on Neural Networks*, pages 1245–1250, 2011. 11

[25] C. Sun, J. Hu, and K. Lam. Feature subset selection for efficient adaboost training. *IEEE International Conference on Multimedia and Exop*, pages 1–6, 2011.

[26] B. Pau, G. Athithan, and M.N. Murty. Speeding up adaboost classifier with random projection. *the 7th International Conference on Advances in Pattern Recognition*, pages 251–254, 2009.

[27] D. Young and J. Ferryman. Faster learning via optimised adaboost. *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 400–405, 2005. 11

[28] R. Gunnar, O. Takashi, and R.M. Klaus. Soft margins for adaboost. *Machine Learning*, 42(3):287–320, 2001. 11

[29] M. Warmuth, K. Glocer, and G. Ratsch. Boosting algorithms for maximizing the soft margin. *In Advances in Neural Information Processing Systems NIPS*, pages 1–8, 2007. 11

[30] M. Warmuth, K. Glocer, and S. Vishwanathan. Entropy regularized lpboost. *the 19th International Conference on Algorithmic Learning Theory*, pages 256–271, 2008. 11

[31] J. Rodriguez and J. Maudes. Boosting recombined weak classifiers. *Pattern Recognition Letters*, 29(8):1049–1059, 2007. 11

[32] Y. Lu, Q. Tian, and T. Huang. Interactive boosting for image classification. *the 7th International Conference on Multiple Classifier Systems*, pages 180–189, 2007. 11

[33] Y. Freund. A more robust boosting algorithm. *preprint ar-Xiv: 0905.2138 available at http://arxiv.org/abs/0905.2138*, 2009. 11, 12

[34] E. Grossmann. Adatree: Boosting a weak classifier into a decision tree. *Workshop on CVPR*, pages 105–112, 2004. 12

[35] Y. Zhang and P. He. A revised adaboost algorithm: Fm-adaboost. *International Conference on Computer Application and System Modelling*, pages 277–281, 2010. 12

[36] S.Z. Li and Z. Zhang. Floatboost learning and statistical face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9): 1112–1123, 2004. 12

[37] J.K. Bradley and R.E. Schapire. Filterboost: Regression and classification on large datasets. *Neural Information and Processing Systems (NIPS)*, pages 185–192, 2007. 12

[38] Y. Sun, J. Li, and W. Hager. Two new regularized adaboost algorithms. *International Conference on Machine Learning and Applications*, pages 41–48, 2004. 12

[39] P. Mallapragada, R. Jin, A. Jain, and Y. Liu. Semiboost: Boosting for semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):2000–2014, 2009. 12

[40] K. Chen and S. Wang. Semi-supervised learning via regularized boosting working on multiple semi-supervised assumptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):129–143, 2011. 12

[41] L.I. Kuncheva. Error bounds for aggressive and conservative adaboost. *the 4th International Workshop on Multiple Classifier Systems*, pages 25–34, 2003. 12

[42] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5):1651–1686, 1998. 12

[43] Y.Q. Wang. An analysis of the viola-jones face detection algorithm. *Image Processing On Line*, 4(2014):128–148, 2014. 14

[44] J.F. Ren, N. Kehtarnavaz, and L. Estevez. Real-time optimization of viola -jones face detection for mobile platforms. *IEEE Circuits and Systems Workshop: System-on-Chip-Design, Applications, Integration, and Software*, pages 1–4, 2008. 14

[45] P. Kumar, J. Verma, and S. Prasad. Data glove: A wearable real-time device for human computer interaction. *International Journal of Advanced Science and Technology*, 43:15–26, 2012. 15

[46] H. Brashear, T. Starner, P. Lukowicz, and H. Junker. Using multiple sensors for mobile sign language recognition. *the 7th IEEE International Symposium on Wearable Computers*, pages 45–52, 2003. 15

[47] M. Strbac, M. Markovic, and D.B. Popovic. Kinect in neurorehabilitation: Computer vision system for real time hand and object detection and distance estimation. *the 11th Symposium on Neural Network Applications in Electrical Engineering*, pages 127–132, 2012. 15

[48] V. Konovalov, A. Clapes, and S. Escalera. Automatic hand detection in rgb-depth data sequences. *the 16th International Conference of the Catalan Association for Artificial Intelligence*, 256:91–100, 2013.

[49] L. Dong, H. Wang, Z. Hao, and J. Liu. Robust hand posture recognition based on rgbd images. *26th Chinese Control and Decision Conference*, pages 2735–2740, 2014. 15

[50] K.P. Ng, G.Y. Tan, and Y.P. Wong. Vision-based hand detection for registration of virtual objects in augmented reality. *International Journal of Future Computer and Communication*, 2(5):423–427, 2013. 16

[51] P. Morerio, L. Marcenaro, and C.S. Regazzoni. Hand detection in first person vision. *the 16th International Conference on Information Fusion*, pages 1502–1507, 2013. 16

[52] J. Wen and Y. Zhan. Vision-based two hand detection and tracking. *the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, pages 1253–1258, 2009. 16

[53] A.S. Ghotkar and G.K. Kharate. Hand segmentation techniques to hand gesture recognition for natural human computer interaction. *International Journal of Human Computer Interaction*, 3(1):15–25, 2012. 16

[54] H. Francke, J. Solar, and R. Verschae. Real-time hand gesture detection and recognition using boosted classifiers and active learning. *Chapter Lect. Notes in Computer Science*, pages 533–547, 2007. 16

[55] P. Cisneros and P. Rodriguez. Practical hand tracking solution by alternating the use of a priori information. *IEEE 5th Latin American Symposium on Circuits and Systems*, pages 1–4, 2014. 16

[56] M. Alsoos and A. Joukhadar. Posture independent model for hand detection and tracking. *IEEE 6th International Conference on Human System Interaction*, pages 175–179, 2013. 16

[57] Z. Zhang, R. Alonzo, and V. Athitsos. Experiments with computer vision methods for hand detection. *International Conference on Pervasive Technologies Related to Assistive Environments*, pages 1–6, 2011. 16

[58] E.J. Ong and R. Bowden. A boosted classifier tree for hand shape detection. *IEEE Conference on Automatic Face and Gesture Recognition*, pages 889–894, 2004. 16

[59] F. Dadgostar, A.L.C. Barczak, and A. Sarrafzadeh. A color hand gesture database for evaluating and improving algorithms on hand gesture and posture recognition. *In Research Letters in the Information and Mathematical Sciences*, 7:127–134, 2005. 16

[60] B. Toni and J. Darko. A robust hand detection and tracking algorithm with application to natural user interface. *the 35th International Convention (MIPRO)*, pages 1768–1774, 2012. 16

[61] A. Mittal, A. Zisserman, and P. Torr. Hand detection using multiple proposals. *British Machine Vision Conference*, pages 1–11, 2011. 16

[62] C. Wang and K. Wang. Hand posture recognition using adaboost with sift for human robot interaction. *Springer Berlin*, 370:1–6, 2008. 17

[63] A. Misra, A. Takashi, T. Okatani, and K. Deguchi. Hand gesture recognition using histogram of oriented gradients and partial least squares regression. *Conference on Machine Vision Applications*, pages 479–482, 2011. 17

[64] J. Kim, J. Baek, and E. Kim. A part-based rotational invariant hand detection. *2013 International Conference on Fuzzy Theory and Its Application*, pages 127–129, 2013.

[65] J. Guo, J. Cheng, J. Pang, and Y. Guo. Real-time hand detection based on multi-stage hog-svm classifier. *IEEE International Conference on Image Processing*, pages 4108–4111, 2013. 17

[66] M.A.A. Aziz, J. Niu, X. Zhao, J. Li, and K. Wang. Using novel shape, color and texture descriptors for human hand detection. *the 11th International Bhurban Conference on Applied Sciences and Technology*, pages 150–157, 2014. 17

[67] J. Niu, X. Zhao, M.A.A. Aziz, J. Li, K. Wang, and A. Hao. Human hand detection using robust local descriptors. *IEEE International Conference on Multimedia and Expo Workshops*, pages 1–5, 2013. 17

[68] I. Fratric and S. Ribaric. Real-time model based hand localization for unsupervised palmar image acquisition. *International Conference on Biometrics*, pages 1280–1289, 2009. 17

[69] B. Stenger, A. Thayananthan, P.H.S. Torr, and R. Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1372–1384, 2006. 17

[70] M.M. Hasan and P.K. Mishra. Novel algorithm for multi hand detection and geometric features extraction and recognition. *International Journal of Scientific and Engineering Research*, 3(5):1–12, 2012. 17

[71] J.L. Raheja, K. Das, and A. Chaudhary. Fingertip detection: A fast method with natural hand. *International Journal of Embedded Systems and Computer Engineering*, 3(2):85–88, 2011.

[72] E. Yoruk, E. Konukoglu, B. Sankur, and J. Darbon. Shape-based hand recognition. *IEEE Transactions on Image Processing*, 15(7):1803–1815, 2006. 17

[73] C. Li and K.M. Kitani. Pixel-level hand detection in ego-centric videos. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3570–3577, 2013. 17

[74] M.M. Hasan and P.K. Mishra. Hand gesture modeling and recognition using geometric features: A review. *Canadian Journal on Image Processing and Computer Vision*, 3(1):12–26, 2012. 17

[75] G.R.S. Murthy and R.S. Jadon. A review of vision based hand gestures recognition. *International Journal of Information Technology and Knowledge Management*, 2(2):405–410, 2009. 17

[76] R. Mustafa, Y. Min, and D. Zhu. Obscenity detection using haar-like features and gentle adaboost classifier. *The Scientific World Journal, Article ID 753860, 6 pages, doi:10.1155/2014/753860*, 2014, 2014. 17

[77] M. Kolsch and M. Turk. Robust hand detection. *the 6th IEEE International Conference on Automatic Face and Gesture Recognition*, pages 614–619, 2004. 17

[78] M. Kolsch and M. Turk. Analysis of rotational robustness of hand detection with a viola-jones detector. *the 17th International Conference on Pattern Recognition*, pages 107–110, 2004. 17

[79] A.L.C. Barczak and F. Dadgostar. Real-time hand tracking using a set of cooperative classifiers based on haar-like features. *Research Letters in the Information and Mathematical Sciences*, 7:29–42, 2005. 17

[80] S. Bilal, R. Akmeliawati, M.J.E. Salami, A.A. Shafie, and E.M. Bouhabba. A hybrid method using haar-like and skin-color algorithm for hand posture detection, recognition and tracking. *International Conference on Mechatronics and Automation*, pages 934–939, 2010. 17

[81] L. Anton-Canalis, E. Sanchez-Nielsen, and M. Castrillon-Santana. Hand pose detection for vision-based gesture interfaces. *Conference on Machine Vision Applications*, pages 506–509, 2005. 18

[82] J.M. Guo, Y.F. Liu, C.H. Chang, and H.S. Nguyen. Improved hand tracking system. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(5):693–701, 2012. 18

[83] S. Wu and H. Nagahashi. Reducing false positive rate for viola-jones method in hand detection. *Technical Report of Technical Committee on Pattern Recognition and Media Understanding*, 113(196):149–153, 2013. 18, 39

[84] S. Wu and H. Nagahashi. A novel preprocessing technique for training viola-jones hand detector. *IEICE Technical Report*, 114(90):37–41, 2014. 18

[85] S. Wu and H. Nagahashi. A new method for solving overfitting problem of gentle adaboost. *SPIE 9069, Fifth International Conference on Graphic and Image Processing*, pages 1–6, 2014. 22, 69

[86] Q. Chen, N.D. Georganas, and E.M. Petriu. Real-time vision-based hand gesture recognition using haar-like features. *IEEE Conference on Instrumentation and Measurement Technology*, pages 1–6, 2007. 31

[87] Q. Chen, N.D. Georganas, and E.M. Petriu. Hand gesture recognition using haar-like features and a stochastic context-free grammar. *IEEE Transactions on Instrumentation and Measurement*, 57(8):1562–1571, 2008.

[88] A.S. Ghotkar and G.K. Kharate. Study of vision based hand gesture recognition using indian sign language. *International Journal on Smart Sensing and Intelligent Systems*, 7(1):96–115, 2014.

[89] Y. Liu and P. Zhang. An automatic hand gesture recognition system based on viola-jones method and svms. *International Workshop on Computer Science and Engineering*, pages 72–76, 2009. 31

[90] Z.L. Fu. Effectiveness analysis of adaboost. *Journal of Computer Research and Development*, 45(10):1747–1755, 2008. 49

[91] K. Bache and M. Lichman. Uci machine learning repository. *http://archive.ics.uci.edu/ml/*, pages Irvine, CA: University of California, School of Information and Computer Science, 2013. 51, 63, 73

[92] A. Vezhnevets and V. Vezhnevets. Gml adaboost matlab toolbox manual. *http://graphics.cs.msu.ru/ru/science/research/machinelearning/adaboosttoolbox*, pages 1–12. 51, 63, 73

[93] Bao face database. *http://www.facedetection.com/facedetection/datasets.htm*. 55

[94] Cmu-mit face database. *http://vasc.ri.cmu.edu/idb/html/face/frontal_images/*. 55

[95] D.C. He and L. Wang. Texture unit, texture spectrum, and texture analysis. *IEEE Transations on Geoscience and Remote Sensing*, 28:509–512, 1990. 87

[96] D.G. Lowe. Object recognition from local scale-invariant features. *the Seventh IEEE International Conference on Computer Vision*, 2:1150–1157, 1999. 88

[97] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2:886–893, 2005. 88

Publication List

Journal Articles

1. **S. Wu** and H. Nagahashi, Robust 2D Hand Detection for Sign Language Recognition based on a Novel Boosting Method, in revision, 2015.

2. **S. Wu** and H. Nagahashi, Penalized AdaBoost: Improving the Generalization Error of Gentle AdaBoost through a Margin Distribution, *IEICE Transaction on Information Processing*, submitted, 2015.

3. **S. Wu** and H. Nagahashi, Analysis of Generalization Ability for Different AdaBoost Variants based on Classification and Regression Trees, *Journal of Electrical and Computer Science*, vol.2015, pp.1-17, 2015.

4. **S. Wu** and H. Nagahashi, Parameterized AdaBoost: Introducing a Parameter to Speed up the Training of Real AdaBoost, *IEEE Signal Processing Letters*, vol.21, no.6, pp.687-691, 2014.

International Conferences

1. **S. Wu** and H. Nagahashi, A New Method for Solving Overfitting Problem of Gentle AdaBoost, *SPIE 9069, The 5th International Conference on Graphic and Image Processing*, pp.1-6, 2014.

2. **S. Wu** and H. Nagahashi, Real-time 2D Hands Detection and Tracking for Sign Language Recognition, *The 8th IEEE International Conference on System of Systems Engineering*, pp.40-45, 2013.

Domestic Conferences

1. **S. Wu** and H Nagahashi, A Novel Preprocessing Technique for Training Viola-Jones Hand Detector, *IEICE Technical Report*, vol.114, no.90, pp.37-41, 2014.

2. **S. Wu** and H Nagahashi, Reducing False Positive Rate for Viola-Jones Method in Hand Detection, *Technical Report of Technical Committee on Pattern Recognition and Media Understanding*, vol.113, no.196, pp.149-153, 2013.