

## A Fusion-based Multiclass AdaBoost for Classifying Object Poses Using Visual and IR Images

*Master of Science Thesis in Communication Engineering,*

**MOHAMED HASHIM CHANGRAMPADI**

Department of Signals and Systems, *Signal Processing Group*  
CHALMERS UNIVERSITY OF TECHNOLOGY  
Gothenburg, Sweden, 2011  
Report No. EX083/2011



Master of Science Thesis in Communication Engineering  
REPORT NO. EX083/2011

# A Fusion-based Multiclass AdaBoost for Classifying Object Poses Using Visual and IR Images

Mohamed Hashim Changrampadi

Supervisor and examiner  
Prof. Irene Y.H. Gu



Department of Signals and Systems  
*Division of Signal processing*  
CHALMERS UNIVERSITY OF TECHNOLOGY  
Göteborg, Sweden 2011

A Fusion-based Multiclass AdaBoost for Classifying Object Poses using  
Visual and IR images

© Mohamed Hashim, 2011

Report No. EX083/2011

Department of Signals and Systems  
Chalmers University of Technology  
SE-412 96 Göteborg  
Sweden.  
Telephone: + 46 (0)31-772 1000

Göteborg, Sweden 2011

## **Abstract**

Visual object recognition and classification have wide applications in many areas, e.g., medical, consumer, surveillance. Many investigations have been done in this field, yet there exists great potentials of improvement in order to bridge the gap between computer vision and human recognition system.

In this thesis, human faces are considered as the object of interest. The motive of this thesis is to classify object poses in visual and thermal infrared (IR) images, and then apply a fusion technique to improve the classification rate. A thermal IR camera is used to capture thermal images, creating a thermal IR face pose database. Both IR and visual face images are used for training and testing. A new approach, multi-classifier AdaBoost, is proposed for multi-class classification. It is used to classify object poses in visual and IR images. Though individual classification of visual or thermal IR images achieves a classification rate around 95%, the efficiency is further improved with the fusion of these two types of images (> 98%). The classification method is also tested for another database.

Viewing that a thermal IR camera is an expensive solution for consumer applications, alternative equipment such like Near-IR webcam and Kinect are investigated and tested in this work. Using images captured by a self-made Near-Infrared (NIR) web-camera or Kinect, our preliminary tests indicate the proposed classifier results in approximately similar performance using such images.



## **Acknowledgement**

All Praise and thanks to Almighty Allah who has blessed me with innumerable gifts and for providing me a lust for research that has ended me in Chalmers. I utilize this opportunity to thank Chalmers University of Technology and in specific Department of Signals and Systems for sharing the valuable resources and laying down the foundation for my thesis. I foremost express my deep gratitude to my supervisor and examiner Prof. Irene Gu , who has lead me, guided me, directed me, and pushed me to achieve the goals that I intended. Special thanks to her for being patient and believing in me to deliver her expectations. I also thank Mr. Zulfiqar Hasan for his valuable suggestions.

I would like to thank the Department of Power for lending the thermal camera to build the visual-thermal database. I would like to specially thank Syed Masih Emami and Yixiao Yun for their tiring effort in creating the database. I also acknowledge the contributions of all those who allowed us to capture their face image for the database.

Heartfelt thanks to all my friends for their support and encouragement that led me to complete my thesis. All the useful and the useless discussion with Naga, Ajay, Shanmukh, Ahmer, Umair was pleasant. Finally the overall credit and thanks goes to my family for supporting me with tremendous love and care. Thanks to my nieces and nephews, Fathima, Juveriya, Aiman, Maaz, Owais and Saad, for entertaining and keeping me active all through the thesis work.





# Table of Contents

1	Introduction .....	1
1.1	Outline of the work.....	2
2	Background .....	3
2.1	Feature Extraction .....	4
2.1.1	Haar-like features.....	4
2.1.2	Histogram of Oriented Gradients:.....	5
2.2	Weak learners.....	7
2.2.1	Decision tree based weak learner.....	8
2.2.2	Perceptron-based Classifier.....	10
2.3	Boosted Classifiers .....	11
2.3.1	AdaBoost .....	11
2.3.2	Multiclass AdaBoost:.....	17
3	Equipment.....	19
3.1	Thermal Imaging.....	19
3.1.1	Thermal IR Camera.....	19
3.1.2	Creating Chalmers Visual-Thermal Database .....	20
3.2	Self-made NIR Webcam .....	21
3.2.1	Procedure to convert a conventional webcam to NIR webcam .....	22
3.2.2	Demo Experiments.....	23
3.3	KINECT.....	24
3.3.1	Computer Vision Applications Using Kinect .....	24
3.3.2	Interfacing Kinect to PC.....	25
3.3.3	Demo Experiments.....	27
4	Classification of Object Poses from 2D Image .....	28
4.1	Overview .....	28
4.2	Feature Computation.....	28
4.2.1	Computation of Integral Image.....	29
4.3	Multi-Threshold Weak Learners:.....	31
4.4	Multiclass AdaBoost Classifier.....	32

4.4.1	Pseudo code.....	33
4.4.2	Choosing a weak learner .....	34
4.4.3	Computing classifier weights .....	35
4.4.4	Update of sample weights:.....	36
4.4.5	Final Estimation.....	36
4.5	Fusion of Visual and Thermal Classifier.....	37
5	Experiments and Results.....	39
5.1	FERET Database .....	39
5.2	Chalmers Visual-Thermal Database .....	40
5.2.1	Classification Using Visual Images.....	40
5.2.2	Classification using Thermal IR images.....	43
5.2.3	Fusion of Visual and Thermal IR Classification.....	45
5.3	Evaluation.....	46
5.4	Demo Experiment.....	48
6	Conclusion.....	49
6.1	Future Work .....	49
7	Bibliography .....	51

# 1 Introduction

In this ever-changing world of engineering and technology, each new day comes with new innovation in various fields that wants to imitate the human capabilities. One such area is the object recognition and classification where research has led the machine to see and recognize like humans. The ability of a human to recognize the object is mimicked by machine learning process. Machine learning is a way of training using a learning algorithm which makes the machine to respond any query by understanding its relation to trained data. As we are concerned in this thesis with object classification, we discuss some of its methods, challenges and some applications. The common forms of machine learning are supervised learning and unsupervised learning. The supervised learning deals with the training of labeled data and classifies the test data into any of the predetermined class. These training with labeled data will become a tedious job for a large dataset. Then unsupervised learning uses the inputs without labeled classes, and fits a model to the input and classifies with the created model. Supervised learning is considered in this thesis. The basic blocks of a supervised learning consist of a feature extraction and a classifier. Some unique features from the inputs are extracted and the classifier maps these features to the desired output.

Though these evolving techniques work well in many occasions, yet they fail to perform in some conditions. Object classification has to overcome lot of complex challenges. There are several parameters that affect the classification and pose a challenge to it. Some of them are variation in lighting conditions, different size and shape, complex postures, occlusion, background clutter etc. The variations in light condition tend to have the most effect in terms of the facial pose classification. The pose changes randomly and hence it is very difficult to train every single possibility of pose. Investigations on numerous techniques have been on achieving the robustness and accuracy to reduce the complexity. One way is the fusion of thermal and visual band images to overcome the limitations of individual classification [1] [2] [3].

The use of thermal infrared (IR) images has achieved better results where the visual systems fail in uncontrolled lighting conditions. The thermal IR band lies in mid-wave infrared (MWIR, 3-5 $\mu\text{m}$ ) and long-wave infrared (LWIR, 8-14 $\mu\text{m}$ ). A thermal image is the visual representation of the energy that a body emits and transmits, while it is the reflected one in the case of visual imaging. Thermal sensors basically estimate the temperature which is the measure of the energy but due to multiple sources the estimate cannot be very accurate. In a thermal imaging camera these energy levels are processed to produce a visual interpretation which will be considered as the thermal image. This image, as depends upon the thermal sensors, does not represent the actual temperature of the object but gives the approximation compared to the

rest. Hence thermal imaging can be utilized in the abnormal light conditions or even in pitch darkness giving us an alternative to the visual recognition limitations. However thermal imaging is also subject to degradation in face image classification especially when cosmetics are involved like glasses which block the thermal emissions. Hence using only thermal images would not be a good choice for efficient classification.

The fusion of these images becomes the ultimate choice to improve the classification rate and reduce the complexity. The thermal image is most often captured by a specialized thermal camera, which is neither affordable nor easy to obtain for consumer applications. To overcome this issue, near infrared images (NIR) are used, which has the same advantage as the thermal image. The use of NIR has also resulted in good performance [4].

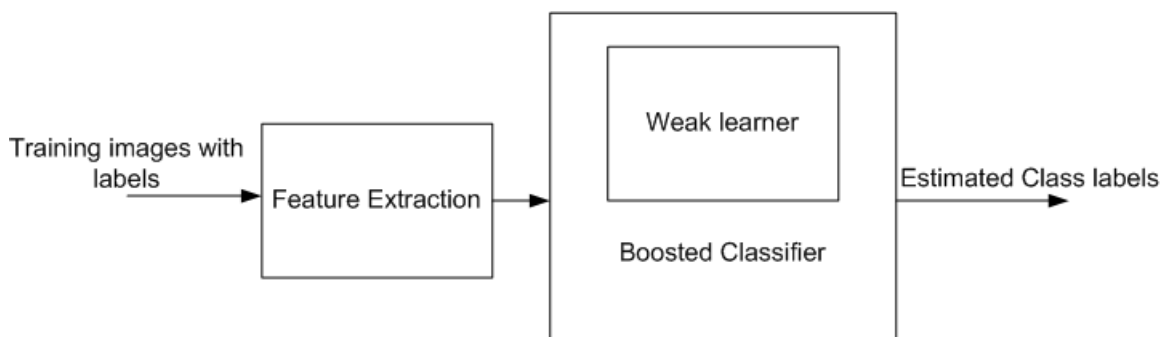
## 1.1 Outline of the work

The current chapter 1 has given an introduction to object classification, its challenges and methods to overcome it. The outline of the rest of the document is listed below.

- **Chapter 2** gives an overview of methods that are investigated for classification of poses. Three major elements of the classification block are discussed. They are feature extraction, weak learner and the boosted classifier. Methods for fusion of visual and thermal infrared images are also stated in this chapter.
- **Chapter 3** introduces the equipments which are used for the experiment. This chapter will enable the reader to know the different equipment that can be considered for image fusion. The use of thermal imaging camera and the creation of Chalmers Visual-thermal database is explained. Procedures for modifying a webcam into near infrared webcam are listed. The use of Kinect for visual and infrared image capture is discussed. Sample images captured with these equipments are shown.
- **Chapter 4** states the proposed technique for classification of object poses. The detailed description of haar-like feature as the feature extraction method is done. The proposed Multiclass-AdaBoost algorithm is explained. The method for fusion of the visual and thermal classifier is also explained.
- **Chapter 5** shows the results obtained by applying the proposed classifier. The performance of the training and the test data is analyzed. The classification rate of using visual or thermal, visual and thermal classifiers are compared. The overall improvement due to fusion is tabulated.
- **Chapter 6** concludes the thesis work with discussions on the improvement by fusion. The extension of the thesis work is included in the future work.

## 2 Background

In this section, we will give an overview of methods that can be used for object classification. The object classification can be handled in many ways. We concentrate on supervised learning using boosted classifiers. Based on this, a brief block diagram of a classification system is shown in Figure 2.1



**Figure 2.1: Basic block for classification of objects**

The training database consists of huge number of images with labels that belongs to their corresponding class. These training images have redundant information if used without any processing. Hence these training images are characterized by their features by some feature extraction method. The features give unique property of the image that stands alone from other images of other class. These unique features are extracted for all images and used by the classifier to classify or group the features of specific class. The classifier can be a simple classification algorithm that recognizes the pattern of particular class and classifies accordingly. However this becomes a difficult job for a single classifier as there may be many features that can be extracted and using only one feature to classify is not sufficient. Hence an ensemble of classifiers is used, that combines many simple classifiers and make a final strong classifier for the final estimation. The simple procedure of a boosted classifier or ensemble classifier is shown Figure 2.1.

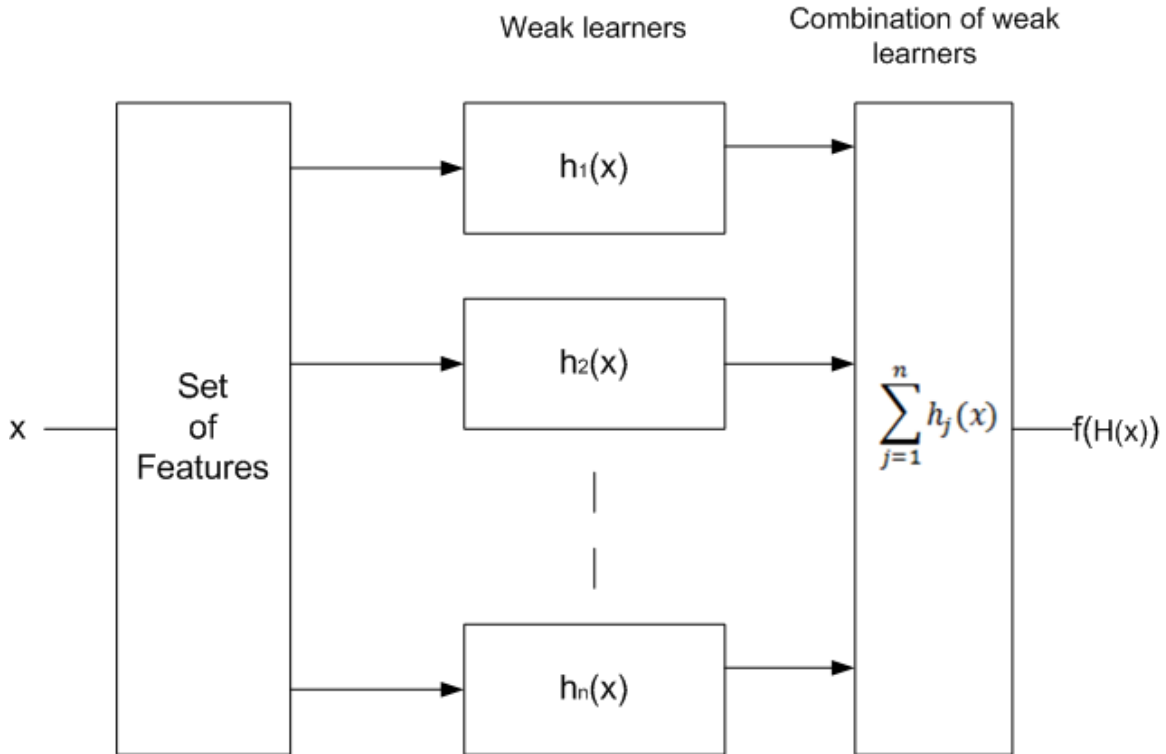


Figure 2.2: Ensemble of Classifiers

The weak learners or base learners perform only slightly better than a random guessing. Though the accuracy of these weak learners is not so accurate, combining many of these simple predictions could result in accurate classification. This combination or weighted averaging of the weak learners is called as boosting which really improves or boosts the accuracy of the weak learners [5]. Many works for improving the boosting algorithms resulted in the AdaBoost algorithm introduced by Freund and Schapire [6]. This AdaBoost algorithm simulates the weak learner for  $N$  number of iterations (or boosting rounds) and picks a best weak classifier in each iteration, while reducing weights to correctly classified examples and giving more weight to misclassified examples. The algorithm is explained in detail in Section 2.3.

## 2.1 Feature Extraction

### 2.1.1 Haar-like features

Haar-like features [7] has been used most often as a feature extraction method due to its simplicity and acceptable performance. These intensity-based features are calculated by using different size of rectangular features as shown in Figure 2.3. The integral image [7] computation makes it easy to extract haar-like features. These features are boosted by AdaBoost classifier and have led to good results for face recognition in real time [7]. The detailed procedures for computing these features are discussed in 4.2. This particular feature extraction method is used in the thesis; however other methods are also investigated.

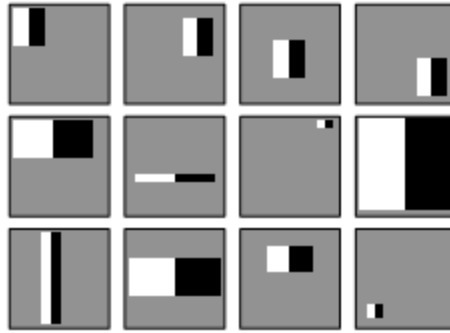


Figure 2.3: Subset of Haar-like feature types

### 2.1.2 Histogram of Oriented Gradients:

Histogram of Oriented Gradients [HOG] [8] is a method of feature extraction. The HOG is similar to edge detectors and SIFT but it is applied differently. HOG uses a single scale block scanning densely along the whole image, while SIFT is multi-scale. Step-by-step description of HOG explaining with illustrations is given in [8], displayed in Figure 2.4. These HOG features extracted from the images are used to classify an object. The HOG uses the orientation histograms from each block as a feature vector, where blocks are formed by the dense grid of cells in the images. In short, the procedure to obtain HOG is to take first derivative of the input image, compute the magnitude of gradient, compute gradient orientations, obtain weights to vote for the orientation histogram and finally compute orientation histograms. There are four variants of HOG encoding, namely Rectangular HOG (R-HOG), Circular HOG(C-HOG), Bar HOG and Centre-Surround HOG. Only Rectangular-HOG is discussed in this thesis. R-HOG, which is called so because the shape of the blocks considered are either square or rectangular.

Obtaining the orientation histograms is the main task in HOG. Orientation histograms with pre-specified bins are calculated for each cell. For each pixel, the gradient magnitude is used as the weight to the corresponding orientation bin. Simple way of voting is to consider the nearest orientation bin but it results in aliasing. Hence trilinear interpolation as described in [8] is considered to vote among the orientation bins. Orientation bins are spaced over  $[0,180]$  which are 'unsigned', where the sign of each gradient is ignored. Orientation bins are spaced over  $[0,360]$  which are 'signed', where the sign of each gradient is considered. 'Signed' seems to be good choice for human detection; it needs to be verified for pose recognition. The pseudo code for the Rectangular HOG is described in Table 2.1.

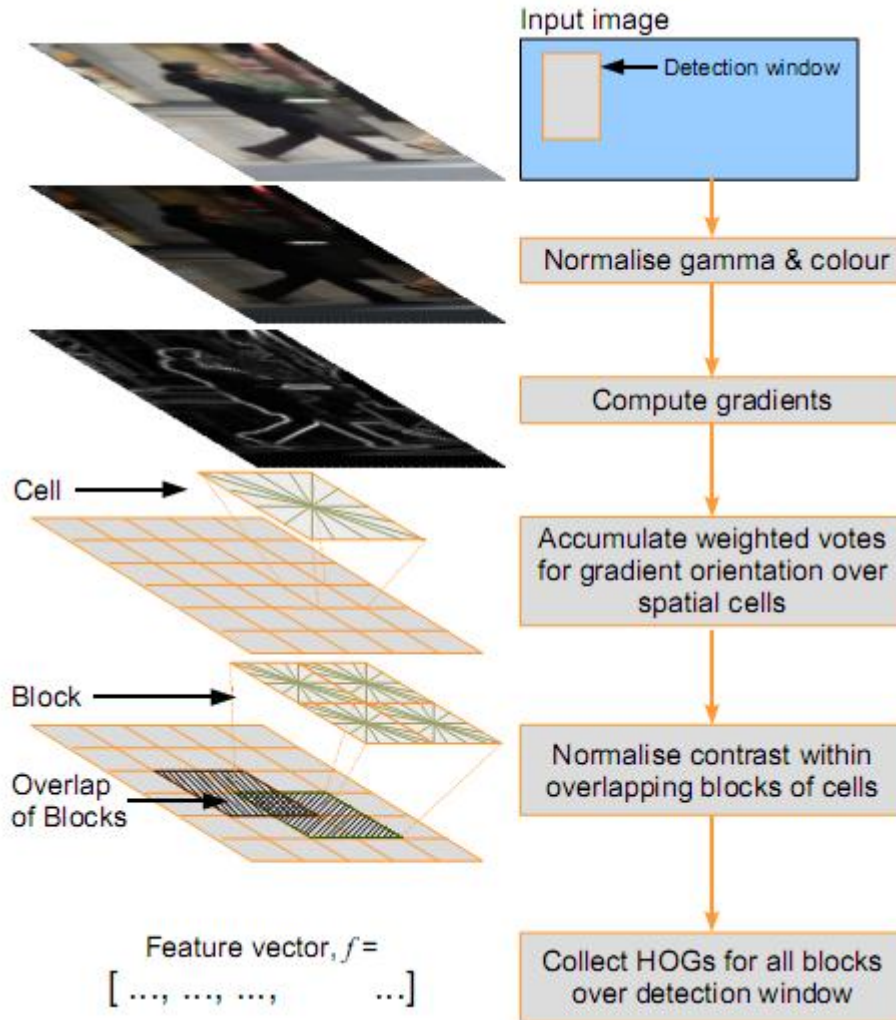


Figure 2.4: An overview of Static HOG feature extraction [8]

---

**Pseudo code:** (R-HOG)

**Input:**

- Input Image  $I$
- HOG Descriptor parameters
  - $Cell = \eta \times \eta \text{ Pixels}$
  - $Block = \zeta \times \zeta \text{ cells}$
  - $Orientation \text{ bins}(\beta)$ ,
  - $Overlap(\varrho)$

**Output:** Feature vector  $F = \{f_1, f_2, \dots, f_k\}$ ,

where,  $k = \text{number of blocks}$ ,  $\text{size}(f_i) = \zeta\zeta\beta$



**Steps:**

- Take the first order derivative of Image  $I$  in X-axis and Y-axis
    - $I_x = h_x * I$       $h_x = [-1 \ 0 \ 1]$
    - $I_y = h_y * I$       $h_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$
  - Compute image gradient magnitude  $I_m$  from  $I_x, I_y$ 
    - $I_m = \sqrt{(I_x)^2 + (I_y)^2}$
  - Compute gradient orientation  $I_\theta$  from  $I_x, I_y$ 
    - $I_\theta = \tan^{-1}(I_y/I_x)$  in the range  $[0, \pi]$
  - For all Blocks  $b_1, b_2, \dots, b_k$ :
    - Compute weights to vote the orientation histogram
      - Obtain a gaussian window  $g(b_i)$  with  $\sigma = 0.5 * \zeta \eta$
      - Apply the gaussian window to each block to obtain weights
 
$$W_i = I_m^{b_i} * g(b_i)$$
    - For each pixel, use weights  $W_i$  to vote the  $\beta$ -bin orientation histogram using trilinear interpolation method described in [1]
    - Obtain  $v_i$  from the above step of size  $\zeta * \zeta * \beta$
    - Apply L2-norm normalization to  $v_i$ 
      - L2-norm,  $f_i = v_i / \sqrt{\|v_i\|^2}$
- Hence  $F = \{f_1, f_2, \dots, f_k\}$  is the HOG feature vector.

**Table 2.1: Pseudo code of R-HOG [8]**

HOG [8] originally was proposed for human detection. Qiang Zhu et al. [9] has also used HOG as feature extraction method for human detection. Computing of HOG features is enhanced by using integral HOG. Linear SVM is used as the weak learner for AdaBoost. Cascade of AdaBoost is used to reduce the false detection.

## 2.2 Weak learners

The weak learners as shown in Figure 2.2 is a simple classifier whose estimation needs to be little better than the random guessing. The weak learner or the base learner was introduced by Freund et al. [6] in which the weak learner is boosted by use of AdaBoost. The selection of a simple and efficient weak learner is necessary. This is because AdaBoost is used over these weak learners which simulate these weak learners for many iteration. In each iteration, AdaBoost selects a best weak learner and combines all of these best classifiers resulting into a strong classifier.

## 2.2.1 Decision tree based weak learner

### 2.2.1.1 Decision Stumps:

The decision stumps introduced in [10] is commonly used as a weak learner in AdaBoost. The decision stumps are single level decision trees which restrict the classification to a binary case. Therefore, for multi-class classification, C4.5 or CART (Classification And Regression Trees) type of decision trees can be considered. The pseudo code of the stumps, modified for using as a weak learner in AdaBoost is shown Table 2.2

---

**Pseudo code:** Weak learner (Stumps)

**Inputs:**  $\chi = \{x_1, x_2, \dots, x_m\}$  be the training samples, represented by feature vectors,

$$x_i = \{f_1^i, f_2^i, \dots, f_j^i\} \text{ for } j \text{ dimensions.}$$

$Y = \{y_1, y_2, \dots, y_m\}$  be the corresponding class labels of  $\chi$ , for binary classification  $Y \in [-1, 1]$

$W = \{w_1, w_2, \dots, w_m\}$ , be the normalized weights for the input samples,  $\sum_{i=1}^m w_i = 1$

**Outputs:**  $h_1^\theta: \chi \rightarrow Y$ , weak hypothesis obtained after applying threshold  $\theta$ .

**Algorithm Steps:**

For  $d = 1, \dots, j$ ;  $\theta_d = \min(x) : \max(x)$ ;

- Classify  $(x_i < \theta_d) \in Y[-1|1]$
- Calculate Error  $\varepsilon_{d\theta} = \sum w_i * [(x_i < \theta_d) \neq y_i]$
- Take minimum error  $\varepsilon_d = \min(\varepsilon_{d\theta})$

Threshold  $\theta_{th} = \arg \min_{\theta} \varepsilon_d^\theta$

Classify the test examples using  $(x_{id} < \theta_{th}) \in Y[-1|1]$

---

**Table 2.2: Pseudo code for weak learner 'Stumps'**

### 2.2.1.2 C4.5

Decision tree can be used for classification problems. This involves building a tree from the test examples and splitting it into their respective classes. One of the well-known methods of growing decision tree is C4.5 [11].

The algorithm involves dividing the test examples into a subset of examples which is further subdivided until all the examples in the subset belongs to a single class. The misclassification

impurity is calculated for each node while building a tree. The misclassification impurity  $i(N)$  at node  $N$  is given by [12].

$$i(N) = 1 - \max_j p(\omega_j)$$

where  $p(\omega_j)$  is the probability of examples belonging to class  $j$ . Hence if  $i(N)$  is 0, then all the examples belong to one class, so no further branches will be drawn from that node. The decision tree building process starts with selecting a node. In case of a 2D datasets, which is considered in this work, the node is simply a threshold that splits the examples into two regions by having a decision boundary parallel to one of the axes. The choice of this threshold is done through an exhaustive search over all dimensions such that it maximizes  $\Delta i(N)$ .

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R)$$

where  $N_L$  and  $N_R$  are left and right descendent of node  $N$ , and  $\Delta i(N)$  specifies the probability of correctly classified examples. A test data is then passed from the root node along the respective branch to the end leaf which classifies it to its class. The pseudo code of the algorithm is described in Table 2.3.

#### The Pseudo code:

Training phase:

Let ' $x_i$ ' be the training patterns in 'd' dimensions belonging to  $\omega_j$  classes.

- Build tree recursively
  - Sort all examples under node  $N_i$  in all dimensions
  - for  $x_{di}$ ,  $((i - 1)$  values of  $x_d$ )
    - calculate  $i(N^{x_{di}})$  and  $\Delta i(N^{x_{di}})$
  - find best  $x_{di} = \max(\Delta i(N^{x_{di}}))$
  - Create decision node  $N$  based on  $x_{di}$
- Repeat above step until  $N_{end}$ , while  $i(N_{end}) = 0$ , which means all examples under  $N_{end}$  belongs to one class.
- Label  $N_{end} = \operatorname{argmax}_j p(\omega_j)$

Test phase:

- Classify a test data using the tree obtained from the train process
  - For  $m_i$  find Node  $N_{end}$  that it belongs to.
  - $h(m_i) =$  class label of  $N_{end}^{m_i}$

Table 2.3: Pseudo code of C4.5 weak classifier

### 2.2.2 Perceptron-based Classifier

The perceptron is a neural network approach based on combinations of weighted inputs to produce a function which classifies the input. Figure 2.5 shows a typical perceptron, where  $y_i$  are input examples and  $w_i$  are their corresponding weights.

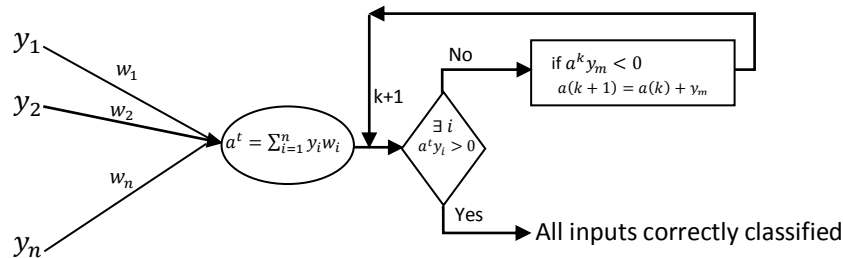


Figure 2.5: An example model of a perceptron

The weak learners discussed in 2.2.1 are performed by splitting the input space parallel to the axes. While in this method, a linear discriminant function  $g(x)$  is used to classify the inputs. The perceptron criterion function decides whether or not all the examples are correctly classified according to the following equation,

$$J_p(a) = \sum_{y \in Y} (-a^t y)$$

where  $J_p(a) > 0$ . It is possible only when the weighted vector  $a$  is on the hyperplane that classifies both classes. If it is not the case, then the vector  $a$  is updated such that it moves close to the decision boundary. The update of the weighted vector is given by

$$a(k+1) = a(k) + \sum_{y \in Y} y$$

The pseudo code of the perceptron-based classifier is summarized in Table 2.4.

#### Pseudo code:

- Let  $y_i$  and  $z_i$  are training and test examples with 'd' dimension and  $w_i$  are their corresponding weights. Class 0 and class 1
- Training;
  - Initialize  $a_d = \sum_{i=1}^n y_{di} w_i$ ,  $T = \text{max iterations}$ ,  $t = 0$ 
    - do,  $t = t + 1$ ;
      - if  $a^t y_m < 0$  [ $y_m = \text{misclassified example; usually random}$ ]
        - update a;  $a(t+1) = a(t) + y_m$
      - until  $\exists i, a^t y_i > 0$  or  $t = T$
    - Return a

- Classifying Test examples
  - if  $a * z_i > 0$ 
    - $z_i \in \text{class 1}$
  - else  $z_i \in \text{class 0}$

Table 2.4: Pseudo code of perceptron-based classifier

## 2.3 Boosted Classifiers

The Boosted classifier is defined as those classifiers which use the concept of boosting the weak learners to improve the classification performance. Instead of building a single efficient classifier, simple weak learners are combined to make a strong classifier with efficient predictions. The basic idea of boosting is to combine simple guesses into a strong prediction. Let  $h_1, h_2, \dots, h_T$  be a set of hypothesis obtained from weak learners, then the combination of these results in a strong hypothesis given by

$$H(x) = \sum_{t=1}^T \alpha_t h_t(x)$$

where,  $\alpha_t$  denotes the weight of the weak hypothesis and it is assigned during the boosting iteration  $t$ . The AdaBoost(Adaptive Boosting) [6] is the first and popular algorithm using boosted classifiers for binary classification. AdaBoost has been used for real time face recognition [7], human detection [9] and many other application. In this section, we will discuss about the original AdaBoost, its theoretical analysis, and some of its variants. Also multiclass AdaBoost will be studied.

### 2.3.1 AdaBoost

The AdaBoost algorithm is an efficient classification algorithm with boosting features. A training set of input samples and its labels  $(x_i, y_i)$  is given as the input to the algorithm. Initial weights to the training samples have a uniform distribution. The AdaBoost runs for  $t = 1, 2, \dots, T$  boosting rounds. For each boosting round, the weak learner computes an initial guess and gives a weak hypothesis  $h_t(x)$ . Before the next boosting round starts, the weights of the training samples are redistributed. The weights of the correctly classified samples are decreased while misclassified samples are assigned with increased weights. The motive of AdaBoost is to concentrate on hardest examples and improve the classification rate. After T boosting rounds, the final hypothesis is the majority voted combination of the entire weak hypothesis.

The generalized version of AdaBoost [3] with more detailed steps is given in Table 2.5

---

**Inputs:**  $\chi = \{x_1, x_2, \dots, x_m\}$  be the training samples, usually feature vectors,

$Y = \{y_1, y_2, \dots, y_m\}$  be the corresponding class labels of input set  $\chi$  ; for binary classification  $Y \in [-1, 1]$

$D(i) = \frac{1}{m}$ , Normalized weights for the input samples.

**Outputs:**  $\mathcal{H}(x)$ , final hypothesis

**Algorithm Steps:**

For  $t = 1, \dots, T$ ;

- Train weak learner using the distribution or weights  $D_t$
- Get weak hypothesis  $h_t: \chi \rightarrow \mathbb{R}$
- Choose  $\alpha_t \in \mathbb{R}$ . Usually  $\alpha_t = \ln \frac{1-\varepsilon_t}{\varepsilon_t}$ ;  $\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$  is the training error
- Update the Distribution:  $D_{t+1}(i) = D_t(i) \exp(-\alpha_t y_i h_t(x_i)) / Z_t$   
Where  $Z_t$  is a normalization factor to make  $D_{t+1}$  to be a distribution.

Output the final hypothesis  $\mathcal{H}(x) = \text{sign}(\sum_{t=1}^T \alpha_t h_t(x))$

---

Table 2.5: Pseudo code of AdaBoost [13]

Using a simple toy example [14], the concept of AdaBoost is explained below.

Let  $+$  and  $-$  be the input example of two classes. The feature response of the input samples is shown in Figure 2.6

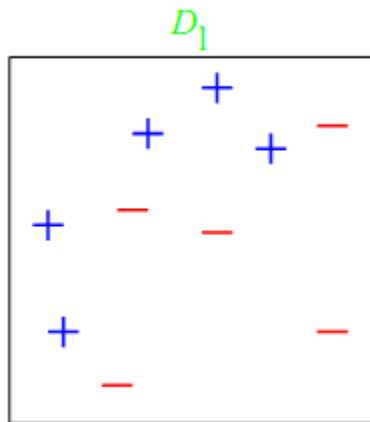


Figure 2.6: Toy example- Feature response [14]

In each boosting round, there are two major steps. One is to find a best weak hypothesis based on low error. Simple weak learner, Stumps is considered for this example. The other step is to update the distribution of the samples for the next round.

### Boosting round 1

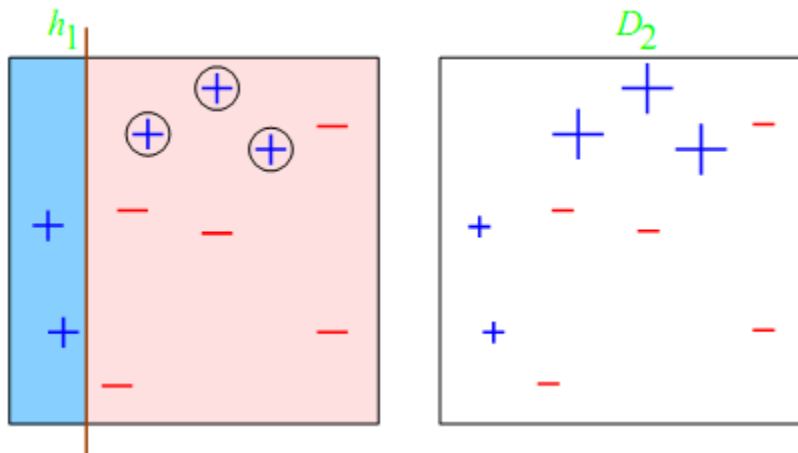


Figure 2.7: Toy Example – Boosting round 1 [14]

- The stumps uses a vertical hyper plane to classify examples, and selects the lowest error  $\varepsilon_1 = 0.30$ , computing weak hypothesis  $h_1$
- The weight of the weak hypothesis is calculated as  $\alpha_1 = \ln \frac{(1-\varepsilon_1)}{\varepsilon_1} = 0.42$
- The selected weak hypothesis misclassifies three examples stated as  $\oplus$ , the weights of these misclassified examples are given more weights while the weight of the other samples is reduced. This is shown in the updated distribution  $D_2$ .

### Boosting Round 2

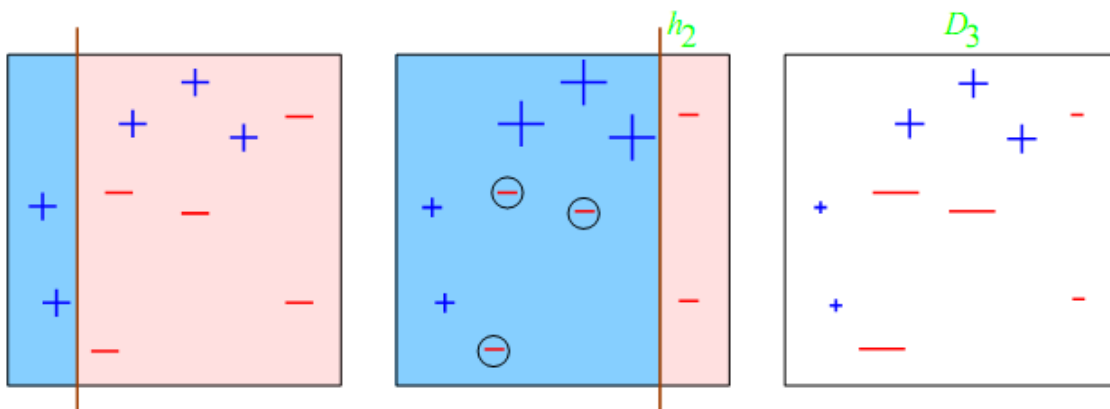


Figure 2.8: Toy Example – Boosting round 2 [14]

- The weak hypothesis  $h_2$  is selected with error  $\varepsilon_2 = 0.21$

- The weight of the weak hypothesis  $h_2$  is computed as  $\alpha_2 = 0.65$
- The weights of the misclassified examples in boosting round 2 are modified and the distribution is modified for next round.

### Boosting Round 3

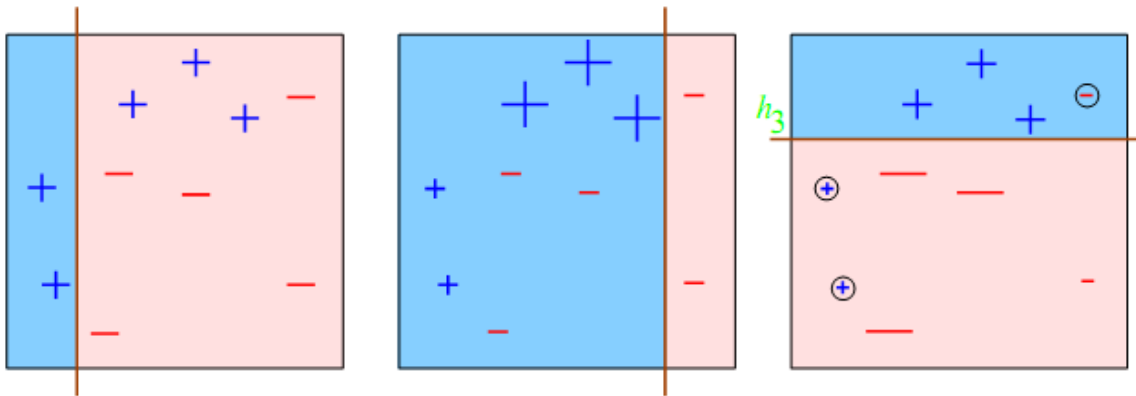


Figure 2.9: Toy Example – Boosting round 3 [14]

- In this round a horizontal hyper plane is used by the weak learner. The weak hypothesis  $h_3$  is selected with error  $\epsilon_3 = 0.14$
- The weight of the weak hypothesis  $h_2$  is computed as  $\alpha_3 = 0.92$ .

### Final Strong Hypothesis

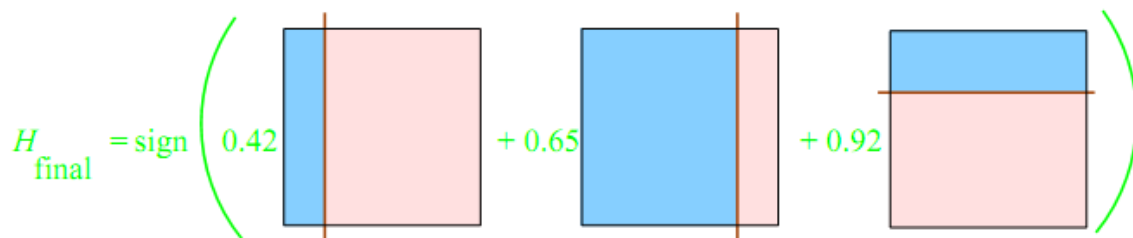


Figure 2.10: Toy Example – combination of weak hypothesis [14]

The final hypothesis is the combination of the weak hypothesis with their weights. This is computed as,

$$H_{final} = \text{sign}(0.42 * h_1 + 0.65 * h_2 + 0.92 * h_3)$$

The final hypothesis looks like in Figure 2.11, thereby classifying all the examples.



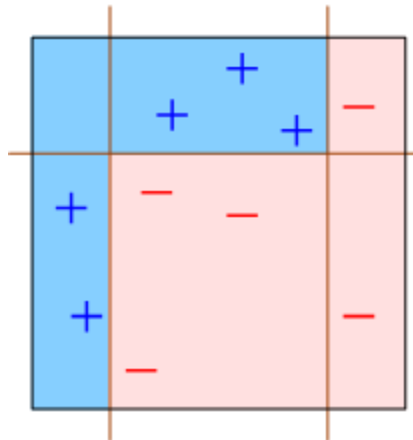


Figure 2.11: Toy Example –Final strong hypothesis [14]

The weights of weak hypothesis:  $\alpha_t$  are weights to the hypothesis  $h_t$  that are inversely proportional to the error of the hypothesis. Therefore, the weighted hypothesis defines their importance or their contribution to the classification. This version of AdaBoost gives only prediction without confidence. The modified one in [15] defines the confidence rated prediction in which the weak hypothesis generates  $h(x) \in [-1, +1]$ , the  $\text{sign}(h(x))$  gives the prediction, and the magnitude  $|h(x)| = 1$  is the confidence measure to the weak hypothesis.

### Training Error

The weak learner needs to perform slightly better than the random guessing for the training error to drop exponentially [15]. Let  $\gamma_t = \frac{1}{2} - \epsilon_t$ , then each classifier should be  $\gamma_t \geq \gamma$  for some  $\gamma > 0$ . The training error upper bound for a binary case is given by  $e^{-2T\gamma^2}$  hence the training error drops exponentially with the increase of boosting round  $T$ .

### Generalization Error

The generalization error is the test error computed after the test images are classified by the trained classifier. The test images are those samples which are not used in the training. These test images are used to test the efficiency or classification performance. Generally for any database, data are partitioned into training set and testing set.

The bound on the generalization error [15] is given as,

$$\widehat{\text{Pr}}[H(x) \neq y] + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$$

where,  $\widehat{\text{Pr}}[\cdot]$  is empirical probability of training error on the training samples,

$T$  is the iteration round of boosting,  
 $d$  is the VC-dimension<sup>2</sup> of the base classifier space,  
 $m$  is the size of the samples.

Generally generalization error decreases for increase in boosting iterations [15].

### Boosting Margin

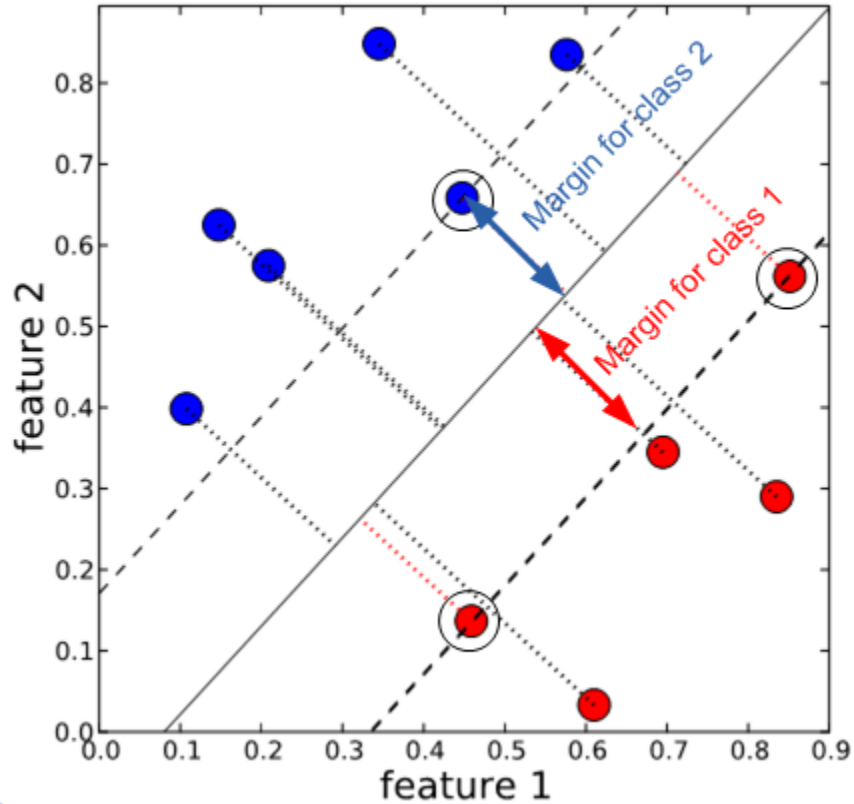


Figure 2.12: Representation of Boosting the margin [16]

Margin [15] is defined as the shortest distance between the training samples and the class boundaries in each class. In other words, it is the distance between the separating hyper plane and the samples as shown in Figure 2.12. Margin for the examples  $(x, y)$  is given as

$$\text{margin}(x, y) = \frac{y \sum_{t=1}^T \alpha_t h_t(x)}{\sum_{t=1}^T |\alpha_t|}$$

AdaBoost increases the margin when the boosting continues even if the error reaches zero [15]. The increase in the margin leads to a reduced generalization error. Hence instead of considering only the prediction, margin should also be investigated to know the stability of the trained classifier. If the margin is small, then the classifier is prone to many errors. However, if the margin is large, then it will reduce the generalization error. AdaBoost has proved to maximize the margin in each boosting rounds. As the iteration of the boosting rounds is increased, the AdaBoost tends to maximize the margin.

### 2.3.2 Multiclass AdaBoost:

AdaBoost is generally designed for binary classification problems, however has been extended for multiclass classification. AdaBoost.M1 [6] and AdaBoost.M2 [6] are the direct extension of AdaBoost for boosting multiclass classification. Another way of extending binary classification to multiclass is reducing it into multiple binary classifications. One such method is AdaBoost.MO which uses coding technique to reduce the multiclass to multiple binary classes.

AdaBoost.M1:

The basic boosting assumption is that a weak learner should perform little better than the random guessing. This criteria is acceptable for binary classification as the probability of classifying between two class is just  $\frac{1}{2}$ . However in case of multiclass classification, this criterion is very strong. For example, in case of 4-class, the probability of classifying is  $\frac{1}{4}$  which is far less than the usual assumption of  $\frac{1}{2}$ . Choosing a good weak learner with special focus on minimizing the error  $\varepsilon_t \leq 1/2$  tends to have the error decreasing exponentially. The pseudo code of the algorithm is given Table 2.6.

**Inputs:**  $\chi = \{x_1, x_2, \dots, x_m\}$  be the training samples, usually feature vectors,  
 $Y = \{y_1, y_2, \dots, y_m\}$  be the corresponding class labels of  $\chi$

$D(i) = \frac{1}{m}$ , Normalized weights for the input samples.

Weak learner:  $h_t: \chi \rightarrow \mathbb{R}$

**Output:**  $\mathcal{H}(x)$ , final hypothesis

**Algorithm Steps:**

For  $t = 1, \dots, T$ ;

- Train weak learner using the distribution or weights  $D_t$
- Get weak hypothesis  $h_t: \chi \rightarrow Y$
- Calculate the error of  $h_t: \varepsilon_t = \sum_{i=1}^m \Pr_i^t [h_t(x_i) \neq y_i]$   
 If  $\varepsilon_t > 1/2$ , then set  $T = t - 1$  and abort loop
- Set  $\alpha_t = \frac{\varepsilon_t}{(1-\varepsilon_t)}$
- Update the Distribution:  $D_{t+1}(i) = D_t(i)\alpha_t^{1-[h_t(x_i) \neq y_i]}$   
 Normalize  $D_{t+1} = \frac{D_{t+1}}{\sum_{i=1}^m D_{t+1}(i)}$

Output the final hypothesis  $\mathcal{H}(x) = \arg \max_{y \in Y} \sum_{t=1}^T \left( \log \frac{1}{\alpha_t} \right) [h_t(x_i) = y_i]$

Table 2.6: Pseudo code for AdaBoost.M1

The choice of weak learner for a multiclass classifier needs to be chosen carefully. Decision stumps used for binary classification cannot be directly applied for the multi-class classification. However, it can be used when it is reduced to multiple binary classification, where for each binary classification, stumps can be used as usual. The other weak learner though can outperform this are c4.5, CART. Any algorithm that can satisfy the weak learning condition can also be used.

AdaBoost.M2 is another extension of AdaBoost, which uses pseudo-loss criteria instead of usual prediction error as the boosting criterion for weak learners. This pseudo-loss is modified in every iteration and sent to the weak learner apart from the modified weights. The weak learner is designed to minimize the pseudo-loss under the constraint  $\varepsilon_t < 1/2$ . The weak hypothesis gives the estimation  $[0,1]^k$  which means for each case, the prediction is done for all labels, and label that is close to 1 defines the final predicted class of that case. This method is somewhat similar to 'one versus all' strategy, where each class is trained to classify itself from all other classes.

## 3 Equipment

### 3.1 Thermal Imaging

**A** Thermal Imaging is the device that captures the thermal emissivity of an object. Contrary to what a visual camera capturing the reflected light of an object, a thermal camera captures the temperature of the object. The idea is to capture the thermal image of face for different poses, which are invariant to light. Meanwhile the corresponding visual image should also be captured. In this project, a thermal IR camera “Fluke Ti45FT-20<sup>1</sup>” was used for this purpose, which is able to capture both thermal image and visual image.

#### 3.1.1 Thermal IR Camera

The Fluke Ti45FT-20 IR camera is IR FlexCam Thermal Imaging equipment with IR-Fusion. The TIR (hereafter referred for Thermal IR imaging camera) is used in this thesis to capture both visual and thermal image. The TIR has a built-in visual lens and an infrared lens shown in Figure 3.1. This enables one to capture concurrently both the visual and thermal images from a same scene. Though the temperature range of the thermal image was automatically calibrated, it can also be changed offline by Fluke SmartView™ software. There was no specific capture environment created, instead TIR was taken to individuals around the Chalmers campus when their face image was captured. Totally, five poses were considered i.e., Frontal, Right, Left, Up and Down. The TIR with right pose captured is shown in Figure 3.1.



Figure 3.1.a: Thermal Camera - Back view



Figure 3.1.b: Thermal Camera - Back view

The detailed specification of the captured images and manually cropped images for the training is given in Table 3.1.

<sup>1</sup> Thanks to Department of Power for lending the thermal imager.

<sup>2</sup> Thanks to Masih and Yixiao who equally contributed in building this database.


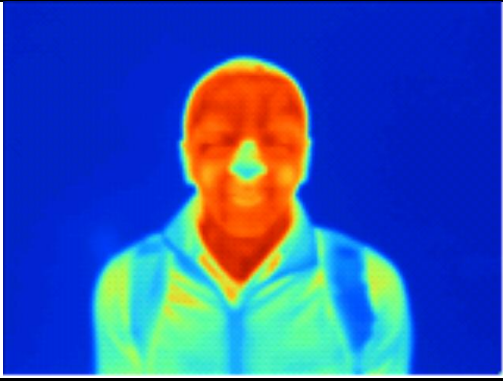
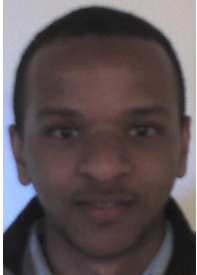
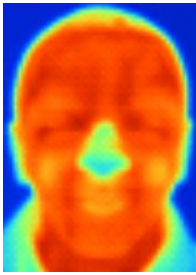
Type of Image	Visual Image	Thermal Image
Original Image		
Size	1280x1024	640x480
Cropped image		
Size (pixels)	271x389	158x210

Table 3.1: Original image and the capture image from the thermal camera

### 3.1.2 Creating Chalmers Visual-Thermal Database

With the help of TIR camera, the Chalmers Thermal-Visual Database was created<sup>2</sup> during the thesis work. The images were captured from different locations around the campus at different time. Since the capture environment was not confined, the database covers whole real-time scenarios with lot of variations, e.g., varying lighting conditions, dissimilar background, with glasses & hats, indoor, day and night etc. The whole process of capturing images for this database took two weeks to complete. All the captured images were cropped manually to remove the background and irrelevant information and make it easy for the training. The cropped images are then normalized to a fixed size. Details and sample images of the created database is listed in the Table 3.2.

<sup>2</sup> Thanks to Masih and Yixiao who equally contributed in building this database.





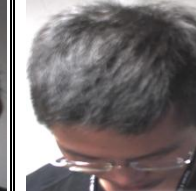
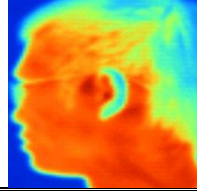
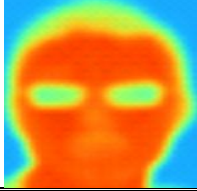
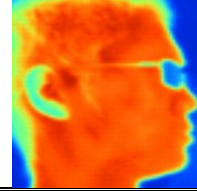
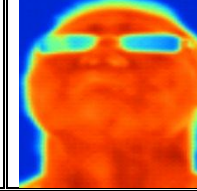
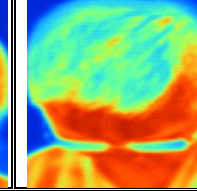
Pose	Right	Front	Left	Up	Down
No. of samples	500	506	500	456	460
Cropped Visual image					
Cropped Thermal image					

Table 3.2: Different pose captured for Chalmers Visual-Thermal Database

### 3.2 Self-made NIR Webcam

The Thermal IR camera which was described in the previous section is very costly to use for a commercial application. Due to this, the TIR is only used in high-end fields like military and research purpose. The basic objective of using NIR images is to make the object visible both during the day and night. Instead of choosing the thermal spectrum, coming little closer to the visible spectrum gives rise to Near-Infrared spectrum. The Near-Infrared (NIR) falls in the infrared spectrum closer to the visible spectrum while the thermal infrared lies in the other end of the infrared spectrum. The whole spectrum is shown in Figure 3.2

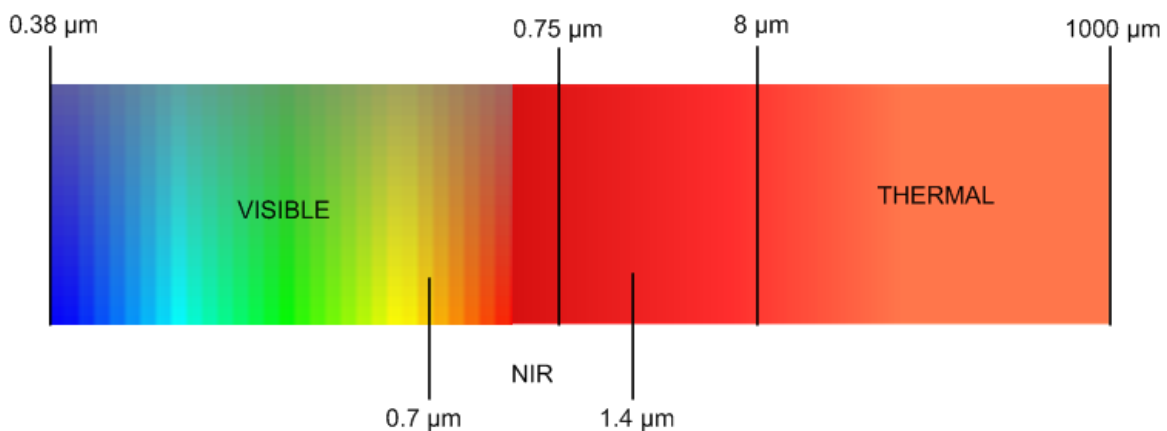


Figure 3.2: Spectral range of visual and thermal

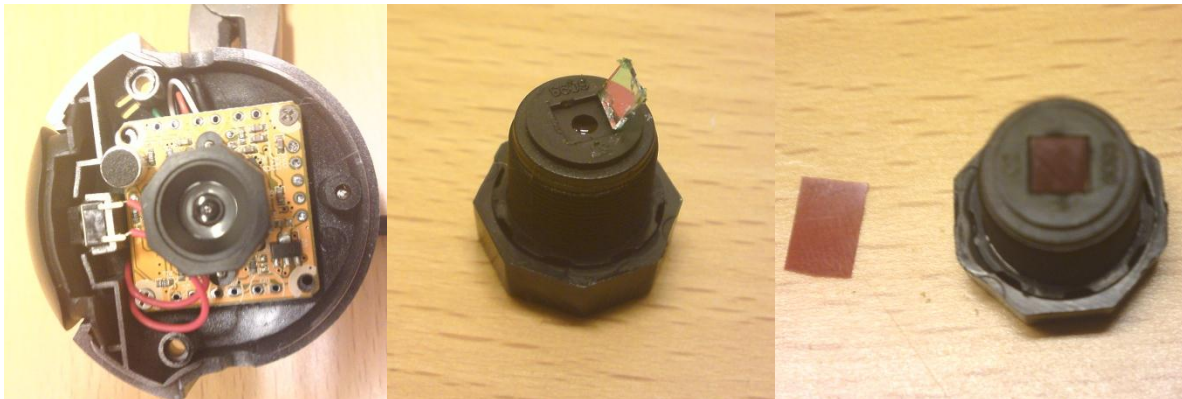
The human eye is sensitive only to the visible spectrum, as the name suggests it. Although Near-infrared exists in normal environment, it is invisible to human eyes. This can be easily viewed by an infrared camera, which can capture the infrared light from any source. Hence, instead of capturing the thermal emissivity of the objects by the TIR, a simple self-modified web camera is

used to capture the Near-Infrared light and provide infrared-like images. This is an alternative method providing an infrared image which is invisible light with low cost equipment.

### 3.2.1 Procedure to convert a conventional webcam to NIR webcam

Two webcam of same type is used in this thesis to capture the visual and infrared images. One of them is conventional webcam for capturing visual image, while the other is self-modified to capture the NIR image. Since there is a need for external source of infrared illumination, an infrared lamp is used. This lamp illuminates the infrared light, which is captured only by the modified infrared webcam without having any effect on the visual image that is captured by the other visual band webcam. The following procedure is performed to convert an ordinary webcam into an infrared webcam, following the suggestion [17].

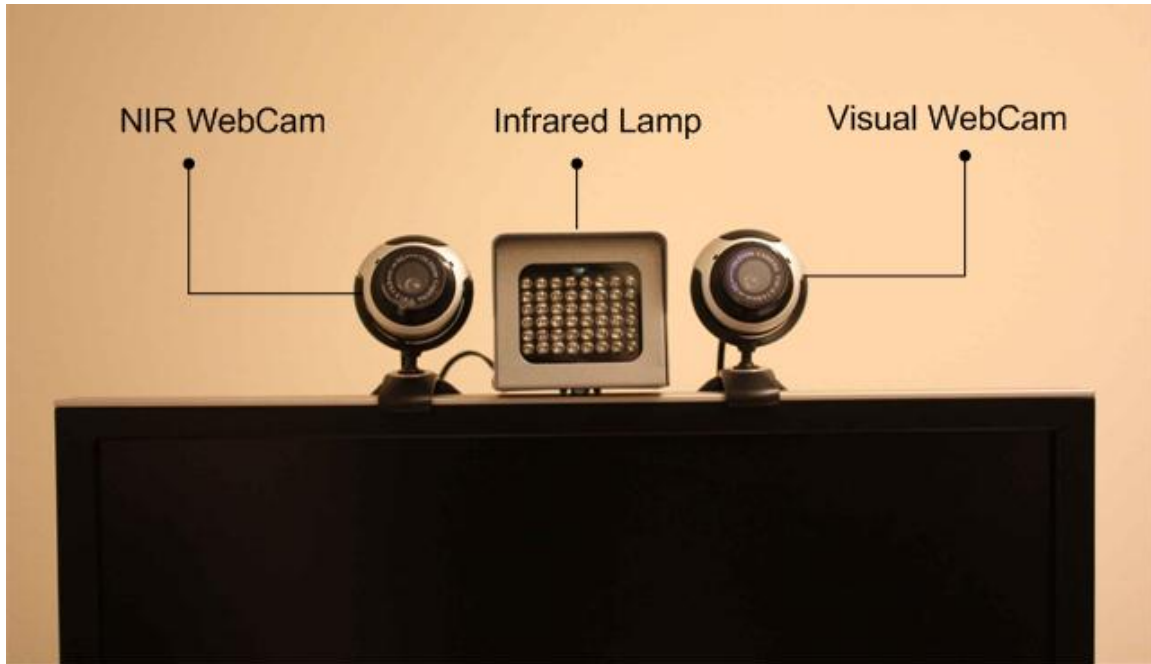
- Unscrew the webcam and remove the camera lens
- There is an IR filter over the lens, which needs to be removed to make it capture the infrared light.
- After removing the IR filter, replace it with a new developed photo film, so that it avoids the entry of visible light. That's it! An infrared webcam is ready to capture the infrared light images.



**Figure 3.3: Step to make NIR webcam (a) lens of webcam (b) Lens with infrared filter removed (c)Infrared filter replace with photo film**

It is worth mentioning that there needs to be an infrared illumination alongside the scene so that the modified webcam can capture the image. Hence an infrared lamp is used which emits the infrared light which is lit besides the webcam to emit the illumination. Another unmodified webcam is used for capturing visual image. Both the webcam needs to be placed such that both capture the same scene to avoid image registration issues. The whole experimental setup is shown in Figure 3.4





**Figure 3.4: Visual and NIR webcam Setup**

Although this technique is very simple and cheap, there are some limitations. The image captured with the modified NIR webcam is distorted with lot of noise. Another major issue is to setup the webcams such that both of them capture the same scene, providing little work to image registration. Also the webcam captures a low resolution image. To overcome these limitations a single device with built-in visual and infrared camera is chosen. Kinect is used for this purpose, which is explained in 3.3

### **3.2.2 Demo Experiments**

Sample images were captured from both the webcam in two different scenarios. One with good lighting condition termed as 'day' and other with bad lighting condition termed as 'night'. During day, the image captured by the visual image webcam and NIR webcam are acceptable for classification purpose. During night, when all the lights are turned off, the quality of the visual image is very poor. The image could be hardly seen with naked eye and hence it cannot be classified correctly. However the NIR images are good even in such dark room, hence enabling the system to classify even in such low light conditions. Some of the sample images captured in different lighting conditions are listed in Figure 3.5.



Figure 3.5: Image capture from Visual and NIR webcam: a) Visual Image (Day) b) NIR image (Day) c) Visual Image (Night) d) NIR Image (Night)

### 3.3 KINECT

The Kinect [18] is an add-on from Microsoft for Xbox game console, which enables a user to be controller itself. The Kinect has inbuilt RGB camera, Infrared camera and infrared light projector. This enables the device to track the user's activity and use them to interact to games. Some of its inbuilt features used with Xbox are face recognition, motion sensing, gesture recognition, voice recognition etc.

#### 3.3.1 Computer Vision Applications Using Kinect

Though primarily released to use for gaming purpose, but due to its varied features, the Kinect has been used in many computer vision applications. Some of them are listed in an overview.

- KinEmote is kinect software for windows enabling the user to interact windows with hand [19]
- Doctors are using Kinect in the operation room to navigate the MRI scan image etc. [20]
- Delta robots are controlled by Kinect to follow the user's gesture [21], and also quadrotor is controlled using Kinect [22]

- The Holographic display using Kinect [23] is built with the help of a LED projector which projects the different perspective of the image based on the location of the user. The location of the user is tracked by Kinect which controls the perspective projection.
- 3D tele-presence system with real time 3D capture [24] using multiple kinect makes the user to interact with anyone in 3D as such the person is beside them.
- Wi-Go [25]: project that helps the disabled person to do the shopping with ease. This is done by enabling the shopping cart to follow the person using Kinect. This makes the person to just roll his wheel chair and keep the shopping items in the cart that is autonomously following him. Though this system was demonstrated for shopping but could be extended to anything where ever there is a need for the disabled person to carry things.
- Controlling the robotic arm using kinect [26] which tracks the user's arm and the robotic arm follows the same action.
- NAVI (Navigational Aids for the Visually Impaired) uses a helmet mounted kinect to navigate the person [27] has been a good use of kinect using it for removing the boundaries of the visually impaired. This helps the visually impaired person to guide them in moving in indoor environment. It gives a message whenever there is obstacle or so. In short it can be said as a GPS system for the blind.
- Humanoid teleoperation [28] with kinect helps the robot to follow the actions performed by the user remotely.
- A mobile autonomous navigation robot [29] using kinect sensors for detecting objects providing wide use from remote learning, collaborative work, homecare, caregiver support etc.
- Optimal camouflage tried with kinect [30] makes the person to be invisible, just one of the adventurous kinect hacks.
- Real time 3D tracking and reconstruction from Microsoft research project KinectFusion [31] has really advanced the use of kinect from real gaming application to using its individual features for other use.
- True 3D shopping making the shoppers to try out their desired cloths and accessories with having them to wear it out. KinectShop [32] has developed a televise shopping, where kinect detects the person and uses the model to allow them to try any accessories giving them a live preview.

### 3.3.2 Interfacing Kinect to PC

The Kinect which was released to work with Xbox was hacked [33] to interface with PC using third-party drivers. This opened ways to use its exhaustive features for developing computer vision programs. Noticing the overwhelming response and use of kinect features for many applications, Microsoft released official versions of Kinect SDK [34] .

The Kinect shown in Figure 3.7 has following main components, RGB camera, infrared camera and infrared projector. The Kinect captures the visual image using the RGB camera, while infrared camera is used to capture the infrared image. Depth image is also calculated using the

infrared projector and the infrared camera. There is tilt motor that helps the kinect to tilt and keep the tracked object in scene.

### Interfacing with PC

There are two ways of connecting Kinect to PC. One way is to use the official Microsoft SDK while the other is to use the third party drivers. The infrared image cannot be captured with the former, hence the latter is used. In particular we are interested to directly interface with MATLAB. Though there are many ways to connect it to PC, but the method used in the thesis is described below.

- Requirements: To get it started, following are the check list,
  - Kinect
  - Windows 32-bit (tested with Win 7 and Win XP)
  - Microsoft Visual Studio 2010
  - MATLAB 32-bit (tested with R2011a)
  - Drivers namely OpenNI Binaries [35] , SensorKinect [36] , and OpenNI Compliant Middleware Binaries (NITE) [35].
  - Kinect for MATLAB (a wrapper functions of OpenNI APIs to interface Microsoft Kinect) [37]
- Installation procedures:
  - Install MATLAB and Microsoft Visual Studio 2010 in 32-bit Windows machine
  - Connect the Kinect (the LED wont glow now).
  - Install the drivers OpenNI, SensorKinect, and Nite (in this order).
  - When the drivers are installed correctly, the LED in kinect will glow green
  - If Kinect LED does not glow, manually select the drivers and update it. The updated device manager will look like in Figure 3.6.

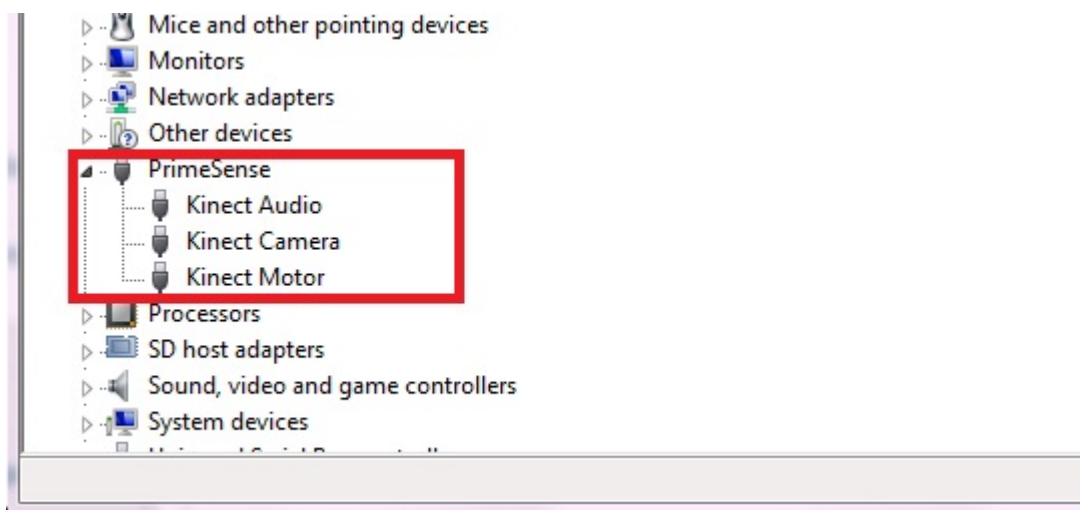


Figure 3.6: Installed Kinect drivers

The Kinect is now interfaced with PC, and using the Kinect MATLAB, now we can extract the desired images. Kinect is used in this thesis to obtain the visual image from the RGB camera, infrared image from the infrared camera, depth image from the depth sensor (infrared camera and the infrared projector). Once the images are extracted it is now given to the classifier for training or testing. Since there is not a database of different pose in infrared to train, the captures images from kinect cannot be used as of now for testing. Also lack of time restricts from creating a new database. However it creates a new path for the future works.



Figure 3.7: Different components in Microsoft Kinect

### 3.3.3 Demo Experiments

The Kinect is now ready to capture the images. These images can be used to train or test the classifier. Some of the sample images captured by the Kinect are shown in Figure 3.8

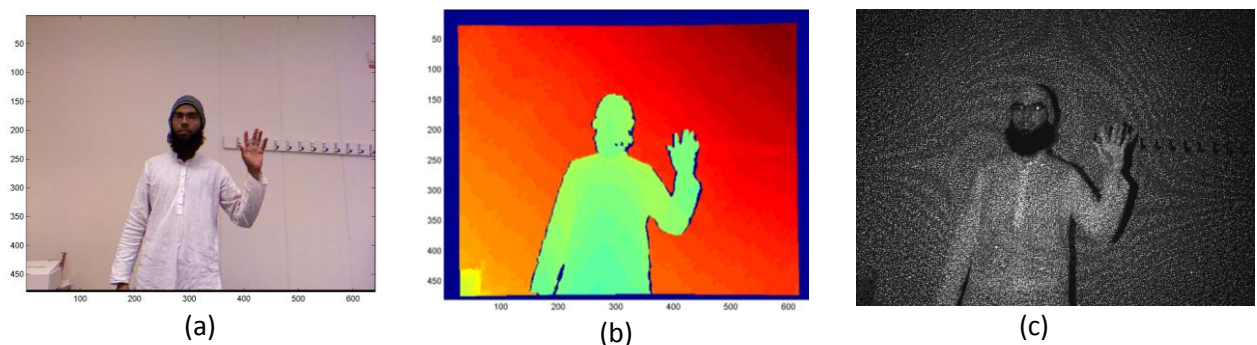


Figure 3.8: Different images captured from Kinect a) RGB image, b) Depth Image, c) Infrared Image

## 4 Classification of Object Poses from 2D Image

### 4.1 Overview

The methodology used in this thesis work is a modified version of AdaBoost [13] enabling one to classify object poses through fusion of IR and visual object information. The classification procedure may be split into two main blocks, namely feature extraction and the classifier. The whole functional block of classification is shown in Figure 4.1.

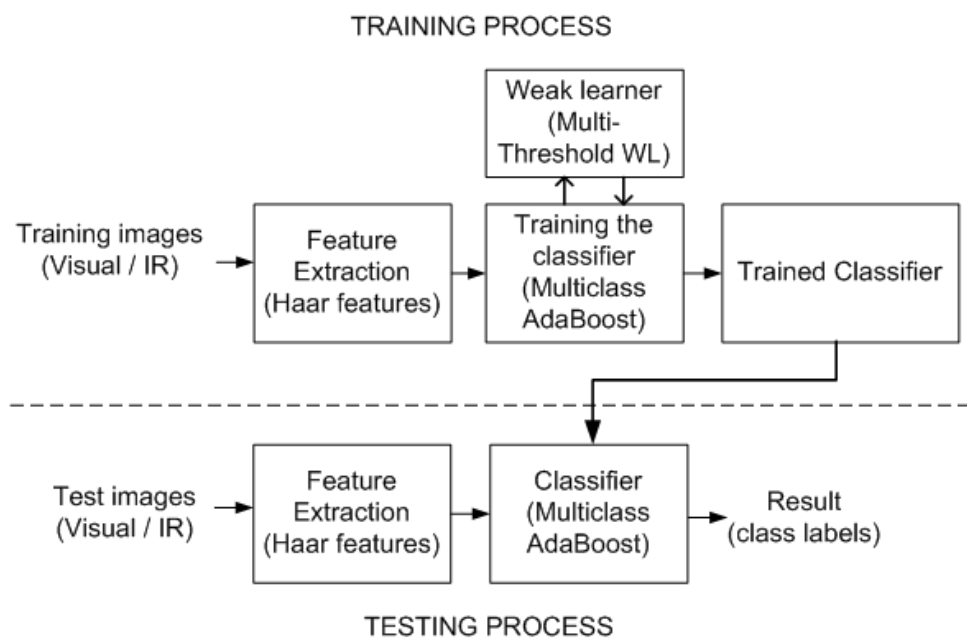


Figure 4.1: Block diagram of object classification

### 4.2 Feature Computation

The classification procedure starts with extracting unique features from the image and uses it for training the classifier. Classifying any image with respect to pixels, would be a difficult task, hence features are used, which is unique to different kind of image.

The feature extraction method described in [7] is used with addition of a new feature type (type V). The feature types shown in Figure 4.3 are rectangular window functions that are convoluted with images to obtain a feature. For each feature type, the sum of all the pixels in black region is subtracted from all the pixels in the white region. This procedure is simplified by use of integral image [7] and have been clearly explained in [38].

### 4.2.1 Computation of Integral Image

The integral image is calculated by replacing its pixel value by sum of entire pixels until its position. This is explained with the help of Figure 4.2

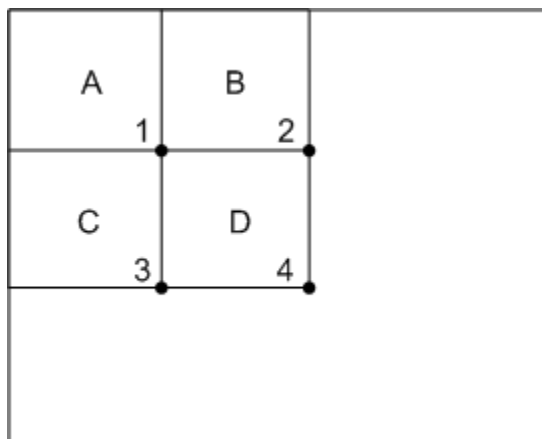


Figure 4.2: Integral image calculation [7]

For a the above image 'I', the pixel at position '1','2','3','4' denotes the sum of all the pixels in region A, A+B, A+C, A+B+C+D respectively. This is done for all the pixels of the image. To cross check it, the last pixel value will be the sum of all pixel of the image. The final obtained image is the integral image. By using this integral image, the computation of the feature value will become easier. The procedure for calculation the feature is given below,

Let  $I$  be the actual image and  $I(x, y)$  be the pixel intensity at position  $x$  and  $y$ . For any pixel in image  $I(x, y)$ , integral image is given by  $I_n(x, y) = \sum_{y=1}^y \sum_{x=1}^x I(x, y)$ . To calculate sum of any region in the image  $I$ , For example, to calculate the sum of pixels in the region D shown in Figure 4.2 is  $D = 4 + 1 - (2 + 3)$ .

For any feature type in Figure 4.3 the difference of the sum of pixels in the black region and the sum of pixels in the white region gives a feature value. This is calculated for all the images which results in a feature vector for one haar-like feature with specific type, size( $w, h$ ) and location( $x, y$ ). The type defines anyone of the type shown in the Fig 4.3. The size denotes the width and the height of the rectangular region. The location denotes the placement of the rectangle in the image. Different type of feature is shown in Figure 4.3.

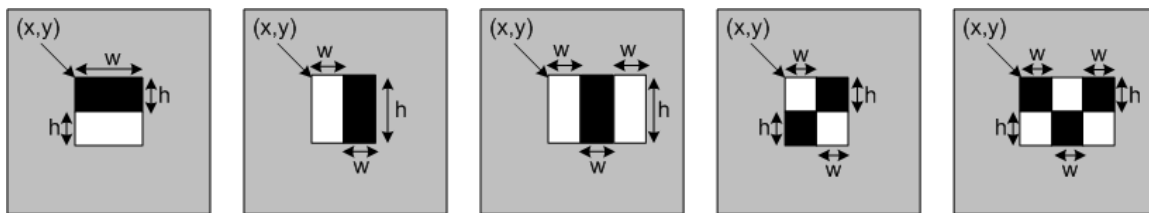


Figure 4.3: Different feature types. (a) Type I (b)Type II (c)Type III (d)Type IV (e)Type V (new)

### Sample feature computation

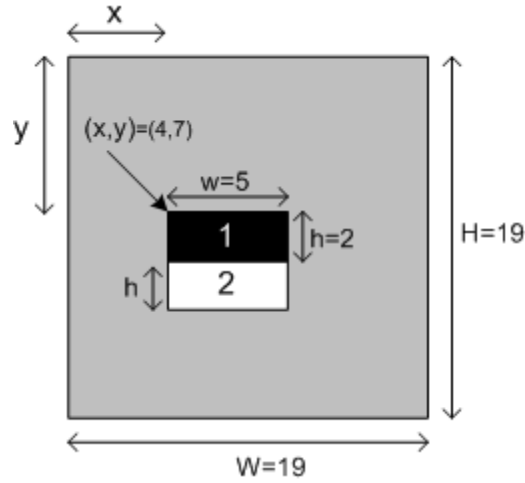


Figure 4.4: Sample feature parameters

Let  $I$  be the image and  $I_n$  be the integral image. Selecting the feature type I, with  $W=19$ ,  $H=19$ ,  $x=4$ ,  $y=7$  as shown in Figure 4.4. Let  $R_1$  be sum of pixels in black rectangular region and  $R_2$  be the sum of pixels in the white rectangular region.  $F(x, y, w, h)$  be its corresponding feature. Hence the  $R_b$  is given by

$$R_1(4,7,5,2) = \sum_{x=4}^{4+5-1} \sum_{y=7}^{7+2-1} I(x, y)$$

Therefore,

$$R_1(x, y, w, h) = \sum_{x=x'}^{x+w-1} \sum_{y=y'}^{y+h-1} I(x', y')$$

$$R_2(x, y + h, w, h) = R_1(x, y + h, w, h)$$

The above computation is done from the image  $I$ , but it does a heavy computation. Hence Integral image is used now to simplify the computation for the same feature type defined in Figure 4.4

- Define  $R$  a zero vector of size  $WH$
- W.k.t  $D = (4 + 1 - 2 - 3)$  hence defining the pixel positions 1,2,3,4 and replace the corresponding positive position with 1 and negative position with  $-1$



$$\begin{aligned}
4 &\Rightarrow R((y + h - 1) + (((x + w - 1) - 1) * H)) = 1 \\
1 &\Rightarrow R((y - 1) + (((x - 1) - 1) * H)) = 1 \\
2 &\Rightarrow R((y - 1) + (((x + w - 1) - 1) * H)) = -1 \\
3 &\Rightarrow R((y + h - 1) + (((x - 1) - 1) * H)) = -1
\end{aligned}$$

And the final feature vector for type I can be computed from the expression

$$F_I(x, y, w, h) = R(x, y, w, h) - R(x, y + h, w, h)$$

Similarly for other feature types in Figure 4.3 ,

$$F_{II}(x, y, w, h) = R(x + w, y, w, h) - R(x, y, w, h)$$

$$F_{III}(x, y, w, h) = R(x + w, y, w, h) - \{R(x, y, w, h) + R(x + 2w, y, w, h)\}$$

$$\begin{aligned}
F_{IV}(x, y, w, h) &= \{R(x + w, y, w, h) + R(x, y + h, w, h)\} - \{R(x, y, w, h) \\
&\quad + R(x + w, y + h, w, h)\}
\end{aligned}$$

$$\begin{aligned}
F_V(x, y, w, h) &= \{R(x, y, w, h) + R(x + w, y + h, w, h) + R(x + 2w, y, w, h)\} \\
&\quad - \{R(x, y + h, w, h) + R(x + w, y, w, h) + R(x + 2w, y + h, w, h)\}
\end{aligned}$$

This results in a feature computation vector  $F$  that is used with any integral image to compute the feature easily. The feature is computed as

$$f_{jx} = FI_n$$

The feature types (a, b, c, d) are used in [7] and the feature type (e) in Figure 4.3 is new feature type introduced in this thesis. Therefor for considering all feature types and possible size and location, for an  $19 * 19$  image, there exist 35686 features which are used as weak learners.

### 4.3 Multi-Threshold Weak Learners:

Once the features are extracted, they are used for training. The weak learner is a simple classification algorithm that can classify at least half of them correctly. These weak learners are used to build a strong classifier. The weak learner used in [7] is a single threshold classifier which is used for the binary classification. The same concept is used in this thesis but modified to obtain a multi-threshold classifier which does the multiclass classification. The feature response of one weak learner is shown in Figure 4.5. The Figure 4.5 shows feature response for three different classes for one single feature.

Total features is given by  $F = \{f_1, f_2, f_3, \dots, f_t\}$  ,  $t =$  total no. of features = 35686

and each feature is given by  $f_j = \{f_{jx_1}, f_{jx_2}, \dots, f_{jx_{n*m}}\}$ ,

where,  $n =$  total number of images in each class,  $m =$  total number of class

Hence for a feature  $f_j$ , the weak learner estimate  $h_j$  is defined as

$$h_{jk}(x_i) = P_{jk}(n) \text{ where } \begin{cases} n = 1 \text{ if } f_j(x_i) < \theta_{j1} \\ n = 2 \text{ if } f_j(x_i) > \theta_{j1} \text{ and } f_j(x_i) < \theta_{j2} \\ n = 3 \text{ if } f_j(x_i) > \theta_{j2} \text{ and } f_j(x_i) < \theta_{j3} \\ n = 4 \text{ if } f_j(x_i) > \theta_{j3} \text{ and } f_j(x_i) < \theta_{j4} \\ n = 5 \text{ if } f_j(x_i) > \theta_{j4} \end{cases}$$

where,  $P \in [1:m]$ ,  $m = 5$ ,  $P$  is computed for all the features and it determines the placement of each class in the feature response graph.  $P_{jk}$  is defined for each of feature  $j$  and classifier  $k$ .  $\theta_{ji}$  is the threshold from left to right, that is computed to give the least error. The feature response for one feature  $type = 24118$  and iteration=1 for the classifier  $k=1$ , is shown in the Figure 4.5. The chosen weak learner is able to classify most of the samples belonging to right pose, while classifying half of the samples of other class. Hence this is chosen as the weak learner for the classifier  $k=1$ , which concentrates its classification for right pose.

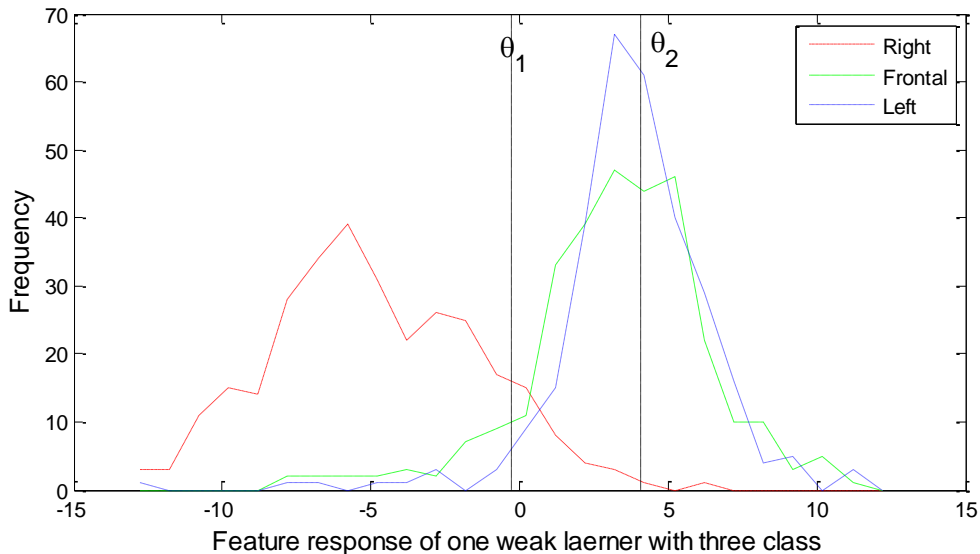


Figure 4.5: Multiple threshold weak learner for three classes showing  $\theta_1$  and  $\theta_2$  (horizontal line)

#### 4.4 Multiclass AdaBoost Classifier

AdaBoost [13] originally proposed for binary classification has also been extended for multiclass classification. But most of the proposed method reduces the multiclass to binary and apply the usual algorithm. Though there has been direct multiclass extension in [39], yet a simple multiclass boosting is proposed. The method uses the simple rectangular features described in [7].

In this method these simple rectangular features are used to obtain a multiple threshold weak learner, unlike single threshold weak learner in [7]. Hence the weak learner by itself is a

multiclass classification. The weak learners are used for boosting such that each strong classifier obtained uses a set of weak learners that are good at classifying one class. Hence each strong classifier is trained to classify one class while the final classification is given by majority vote of all strong classifier. The overview of the method is explained in Section 4.4.1.

The Multiclass AdaBoost classification method involves training 'K' strong classifier, each of which is made to specialize in classifying one class more accurately. Each classifier chooses a weak learner that could classify its class with less error. Hence all the classifiers are trained to perform better in their own class. The final classifier can be a majority vote of all the classifier.

#### 4.4.1 Pseudo code

The pseudo code for the classification of multiple classes is given in Table 4.1.

---



---

#### Pseudo code for Classification using Visual/Thermal Images:

**Inputs:**  $\chi = \{x_1, x_2, \dots, x_l\}$  be the training images,

$F = \{f_1, f_2, \dots, f_n\}$  be the feature set where  $f_j = \{x_{1j}, x_{2j}, \dots, x_{lj}\}$

$Y = \{y_1, y_2, \dots, y_M\}$  be the corresponding class labels of  $\chi$ , where  $Y \in [1, 2, \dots, M]$ ,  $M = \text{No. of Class}$

$w(i) = \frac{1}{l}$ , Normalized weights for the input samples.

Weak Classifier:  $h_j: \chi \rightarrow Y$

**Output:**  $\mathcal{H}_k(x)$  is the final hypothesis, where  $k = 1, 2, \dots, K$

#### Algorithm Steps:

For  $t = 1, \dots, T$ ; boosting iterations

For  $k = 1, \dots, K$  strong classifiers

- Train a weak classifier  $h_{kj}^t$  for each feature  $f_j$
- Choose the best weak learner  $h_{kj}$  minimizing error  $\varepsilon_{kj}^t$

$$\varepsilon_{kj}^t = \left[ \sum_{i \in \{m\}} w_i * \llbracket h_{kj}^t(x_i) \neq m \rrbracket + \sum_{i \notin \{m\}} w_i * \llbracket h_{kj}^t(x_i) = m \rrbracket \right]$$

where,  $m = \text{class } m \text{ for which the classifier } k \text{ is trained}$

$$\llbracket \cdot \rrbracket = \begin{cases} 1 & \text{if satisfies} \\ 0 & \text{otherwise} \end{cases}$$

- Get the weak hypothesis  $h_{kt}: \chi \rightarrow Y$  with error  $\varepsilon_k^t = \min(\varepsilon_{kj}^t)$

- Calculate the Class error for the selected  $h_k^t$

Compute

$$\mathbb{C}_{km}^t = \left[ \sum_{i \in \{m\}} w_i * \llbracket h_k^t(x_i) \neq y_i \rrbracket \right], \forall m = 1, 2, \dots, M$$

- Calculate classifier weights Alpha,  $\forall m \alpha_{km}^t = \log \left( \frac{(1 - \mathbb{C}_{km}^t)}{\mathbb{C}_{km}^t} \right)$

Update the weights of the samples:

$$w_{ki}^t = w_{ki}^{t-1} * \beta_k^{t-1 - \llbracket h_k^t(x_i) \neq y_i \rrbracket} \quad \forall (i \in \{m, m'\}),$$

where,  $m' =$  samples  $\notin m$  but classified as  $m$

$$\text{and } \beta_k^t = \frac{\varepsilon_k^t}{(1 - \varepsilon_k^t)}$$

- Normalize  $w_{ki}^t = \frac{w_{ki}^t}{\sum_{i=1}^l w_{ki}^t}$
- End Loop 'k'
- End Loop 't'

Output the strong hypothesis,

$$\mathcal{H}_{km}(x) = \sum_{t=1}^T \sum_{m=1}^M \alpha_{km}^t * \llbracket h_k^t(x) = m \rrbracket, \quad \forall k, \forall m = 1, \dots, K$$

The final estimation is given as

$$c_m = \sum_{k=1}^K \mathcal{H}_{km}(x) \quad \forall m = 1, \dots, M$$

$$E(x) = \underset{m}{\text{arg max}} c_m(x)$$

**Table 4.1: Pseudo code of Multiclass AdaBoost**

The detailed explanation of the Multiclass AdaBoost about choosing a weak learner, computing classifier weights, re-weighting the sample weights and final estimation are also discussed.

#### 4.4.2 Choosing a weak learner

The Multi-threshold weak learner defined in the 4.3 is used as a weak learner. As we have learnt that for a  $19 * 19$  detector window there are 35686 features. For every iteration, the classifier selects the best weak learner based on low class error. However the error that is used as a criterion to choose the weak learner needs to be defined. In [13] the weak learner minimizes the error,  $\varepsilon_t = \Pr_{i \sim D_i} [h_t(x_i) \neq y_i]$ , which is suitable if a single binary classifier or multiple independent binary classifiers are used. However in this method, the weak learner chosen in the

each strong classifier is a multi-class classifier. Hence the error is calculated based on misclassification of its class and misclassification of other class as its class. In other words, error is summation of false-positive and false-negative. For example, Let  $K=1$ , in which the classifier tends to classify more accurately "Right" samples, then

false positive = Right samples misclassified as Front/Left

false negative = Front/Left samples wrongly calssified as Right

For each boosting iteration 't', classifier 'k' for class 'm' and feature type 'j', the error is given as,

$$\varepsilon_{kj}^t = \text{Misclassification in class 'm'} + \text{Misclassification in other class as 'm'}$$

$$\varepsilon_{kj}^t = \left[ \sum_{i \in \{m\}} w_i * \llbracket h_{kj}^t(x_i) \neq m \rrbracket + \sum_{i \notin \{m\}} w_i * \llbracket h_{kj}^t(x_i) = m \rrbracket \right]$$

where,  $m = \text{class } m \text{ for which the classifier } k \text{ is trained}$

$$\llbracket \cdot \rrbracket = \begin{cases} 1 & \text{if satisfies} \\ 0 & \text{otherwise} \end{cases}$$

Thereafter the weak learner is chosen so as to minimize the above error  $\varepsilon_{kj}^t$

#### 4.4.3 Computing classifier weights

The weights assigned to the classifier  $h_t$  is given by  $\alpha_t$  in [13] and for binary it is set as  $\alpha_t = \ln \frac{(1-\varepsilon_t)}{\varepsilon_t}$ . However in this case of multiclass it needs to be modified accordingly. The weight  $\alpha_t$  is computed for each of the strong classifier and the error  $\varepsilon_t$  is defined for each class separately.

For the classifier 'k', the independent class error  $\mathbb{C}_{km}^t$  is calculated for M classes, and is given by

$$\mathbb{C}_{km}^t = \left[ \sum_{i \in \{m\}} w_i * \llbracket h_k^t(x_i) \neq y_i \rrbracket \right], \forall m = 1, 2, \dots, M$$

Using the above error, weights to the classifier  $h_{kt}$  is assigned. For the classifier 'k' trained to classify class 'm', if the corresponding class error  $\mathbb{C}_{km}^t$  is less, then high weights is assigned to the classifier for classifying class 'm'. And thus the corresponding  $\mathbb{C}_{km}^t$  relates to the weight of the strong classifier for classifying class m.

$$\forall m \alpha_{km}^t = \log \left( \frac{(1 - \mathbb{C}_{km}^t)}{\mathbb{C}_{km}^t} \right)$$

Taking an example scenario, Let  $k = 2$  is the classifier trained to classify class 2 more accurately, hence if the error  $\mathbb{C}_{22}^t = 0.1$ , then, for this strong classifier, the weight for classifying class 2 will be,

$$\alpha_{22}^t = \log\left(\frac{1-0.1}{0.1}\right) = 2.1972$$

Similarly for the same strong classifier  $k = 2$ , the weight for classifying other classes is also computed from  $\alpha_{21}^t, \alpha_{23}^t, \alpha_{24}^t, \alpha_{25}^t$ . This is assumed as the confidence of classifying its class.

#### 4.4.4 Update of sample weights:

The weight of the samples in [13] aims at assigning high weights to those which are misclassified. Since it deals with binary classification and has a single classifier a general rule is applied. But in case of Multiclass-AdaBoost, there are multiple strong classifiers. The motive is to make each classifier strongly trained in classifying one class, and also make sure that it does not classify samples of other classes as its own trained class. Hence the samples of the parent classifier and the samples of other classes that are wrongly classified as the child class are considered for weight update given by

$$w_{ki}^t = w_{ki}^t * \beta_k^{t-1 - \mathbb{I}[h_k^t(x_i) \neq y_i]} \quad \forall (i \in \{m, m'\}),$$

where,  $m' =$  samples  $\notin m$  but classified as  $m$ , and  $\beta_k^t = \frac{\epsilon_k^t}{(1-\epsilon_k^t)}$

Considering a simple scenario to explain, let  $k = 1$  be the strong parent classifier,  $m = 1$  be its child class. Two sets of samples are considered for weight update. First set is the samples of child class  $m = 1$ . In this set the weights of samples that are misclassified as  $y_i \neq 1$  are given high weights. The other set is samples of  $class \neq 1$  that are wrongly classified as  $y_i = 1$ . All these samples will be given high weights, to make the classifier concentrates on these samples.

#### 4.4.5 Final Estimation

Once all the strong classifiers are trained, then the final hypothesis is given as,

$$\mathcal{H}_{km}(x) = \sum_{t=1}^T \sum_{m=1}^M \alpha_{km}^t * \mathbb{I}[h_k^t(x) = m], \quad \forall k, \forall m = 1, \dots, K$$

From the above expression, it is noted that each strong classifier  $\mathcal{H}_k$  will compute the confidence for all the classes, and then the final confidence for each class  $m$ , is given as

$$\text{Confidence for class } m: \quad c_m = \sum_{k=1}^K \mathcal{H}_{km}(x), \quad \forall m = 1, \dots, M$$

The final estimation of any samples, is the class having the maximum confidence, given by

$$\text{Final Estimation:} \quad E(x) = \arg \max_m c_m(x)$$

## 4.5 Fusion of Visual and Thermal Classifier

The proposed multiclass AdaBoost is designed for fusion of visual and thermal classifiers. The algorithm is applied both to the visual and thermal image database. Once the classifiers are trained, we obtain individual classifier namely visual and thermal classifier. The individual trained classifier has the confidence for each estimated class. The final estimation is given by sum of its weighted estimations. The block diagram showing the fusion of these two trained classifier is shown in Figure 4.6.

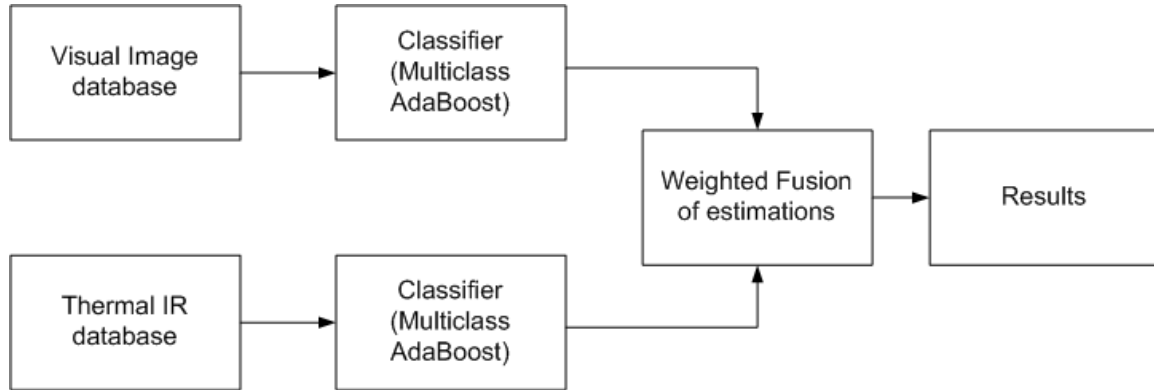


Figure 4.6: Block diagram of Fusion of Visual and Thermal Classifiers

The fusion method applied in this thesis can be considered as decision level fusion. However weights of both the classifiers are used to estimate the final prediction, which improves the classification rate. The pseudo code for the fusion of visual and thermal classifier is given in Table 4.2.

### Pseudo code for Classification using Visual and Thermal Images:

This method involves fusion of both the visual and thermal classifiers and giving a more accurate estimation. The pseudo code is presented below

#### Inputs:

- $\mathcal{H}_{km}^V(x)$  be the strong classifier for visible images,  $\forall k, \forall m = 1, \dots, K$
- $\mathcal{H}_{km}^T(x)$  be the strong classifier for thermal images,  $\forall k, \forall m = 1, \dots, K$

#### Algorithm Steps:

- Compute the individual confidence for each class for visual and thermal images.
  - Confidence for class  $m$  for visual image is given as,

$$c_m^V = \sum_{k=1}^K \mathcal{H}_{km}^V(x) \quad \forall m = 1, \dots, M$$

- Confidence for class  $m$  for thermal image

$$c_m^T = \sum_{k=1}^K \mathcal{H}_{km}^T(x) \quad \forall m = 1, \dots, M$$

- Combine the confidence weights of both the classifier,

$$C_m^{VT}(x) = c_m^V(x) + c_m^T(x)$$

$$C_m^{VT}(x) = \sum_{k=1}^K \mathcal{H}_{km}^V(x) + \mathcal{H}_{km}^T(x) \quad \forall k = 1, \dots, K$$

**Output:** The final estimation is given by,

$$E(x) = \arg \max_m C_m^{VT}(x)$$

**Table 4.2: Pseudo code of fusion of visual and thermal classifier**

The algorithm explained in Table 4.2 is about the fusion of visual and thermal classifier as a general case. Instead of thermal IR classifier, any other classifier like NIR classifier can also be used for fusion. In this thesis work, only visual and thermal classifier is considered. Since there is no available database for NIR face images for different pose, a NIR classifier cannot be trained. Also creating a NIR was a difficult task to do. However some test with NIR image is done with a visual classifier. The experiment that is done using this fusion algorithm is explained in Section 5.



## 5 Experiments and Results

The algorithm described in the 4.1 is used for classification of facial pose. The experiments were conducted using the two databases, namely FERET database and Chalmers Visual-Thermal Database. All the experiments stated below, uses the methodology described in Section 4. The weak learner is the multi-threshold classifier explained in 4.3. Although the weak learner is multi-class classifier, the estimation performance of the weak learner should be at least equal to random guessing. The proposed multiclass AdaBoost is the classifier used for boosting the weak learner. The resultant strong classifier from the visual and the thermal images are combined using a simple fusion algorithm described in 4.5.

### 5.1 FERET Database

The multiclass AdaBoost is applied to the FERET database [40]. The samples images retrieved from FERET database are cropped and resized to 19\*19 pixels. The detail of the parameters used in the simulation is listed in Table 5.1.

<b>Number of class:</b>	3 (Right, Frontal, Left)
<b>Training images in each class:</b>	300
<b>Image size:</b>	19*19
<b>Boosting iterations:</b>	30
<b>Test Images in each class:</b>	100
<b>Type of Image</b>	Visual images

**Table 5.1: Experiment parameters for the classification using FERET database**

The FERET profile images were cropped and used in training. All the images were manually cropped and used for training. Since the database has only visual images, the fusion block is not applied for this database.

The multiclass AdaBoost is trained and tested with only visual images. The comparison of training error and generalization error for all the boosting rounds is shown in Figure 5.1. The generalization error is found to decrease with the boosting iterations.

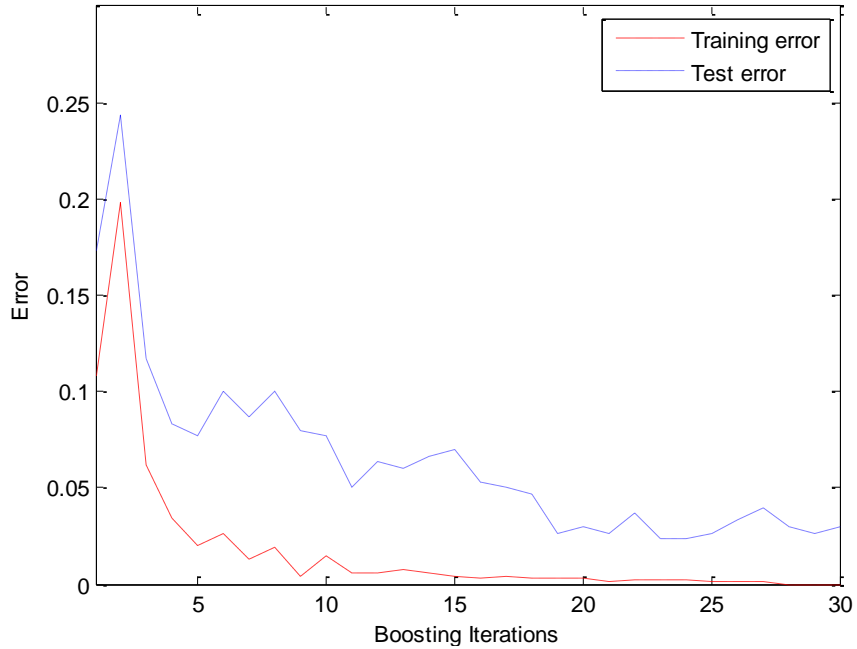


Figure 5.1: Classification error for training and test images along boosting iterations

This experiment is to emphasize on the performance of the multiclass AdaBoost. The concept of boosting helps to improve the classification rate if the classifier runs for a longer iteration.

## 5.2 Chalmers Visual-Thermal Database

The Chalmers Visual-Thermal database has both the visual and thermal samples. First the classification performance is analyzed individually for the visual images and then for thermal images. Finally fusion is applied and compared with the classification rate of only visual and thermal classifiers. The detailed specification of the database is already listed in Table 3.2

### 5.2.1 Classification Using Visual Images

Experiment is conducted for only visual images to evaluate its classification performance. The experiment parameter for classification using only visual images is listed in Table 5.2.

<b>Number of class:</b>	<b>Model 1:</b>	3 (Right, Frontal, Left)
	<b>Model 2:</b>	3 (Up, Frontal, Down)
<b>Training images in each class :</b>		300
<b>Image size:</b>		19*19
<b>Boosting iterations:</b>		30
<b>Test Images in each class:</b>		100
<b>Type of Image</b>		Visual images

Table 5.2: Experiment parameters for the classification using only Visual images

The experiment is conducted for three class classification. Since there are five classes available in the database, two models is trained with three classes each. The model 1 is trained with Right, Front and Left. The model 2 is trained with Up, Front and down. The classification error is given as,

$$\text{Classification Error} = \frac{\text{No. of samples misclassified}}{\text{Total no. of Samples}}$$

$$C_{error} = \frac{N_m}{N_t}$$

The classification error for all the boosting iteration for training and testing is shown in Figure 5.2. From the classification error response, it is evident that the classifier performs well for the test data, and the error drops along the boosting rounds. The individual class error for right, front and left is computed from model 1, while for up, down is computed from model 2 and it is listed in Table 5.3.

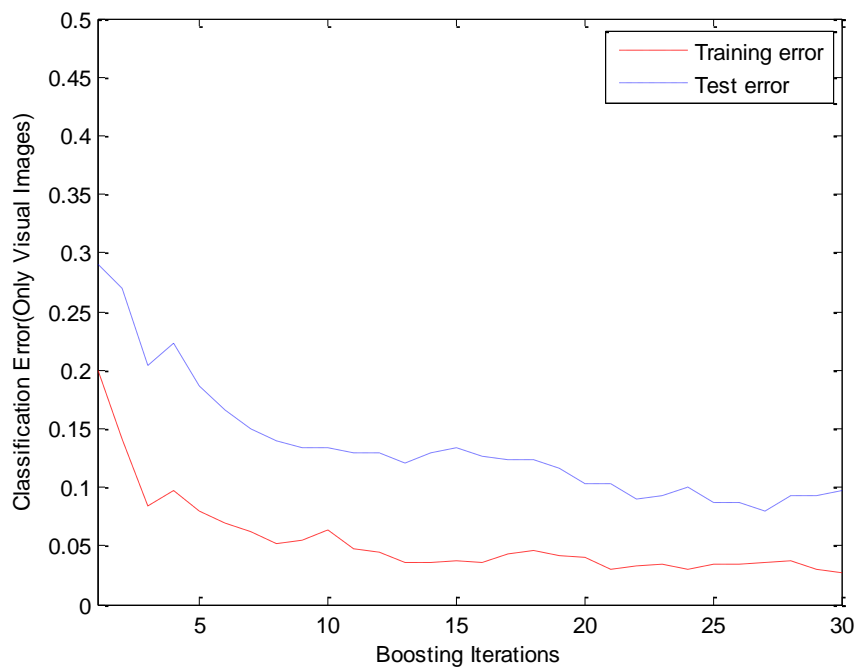


Figure 5.2: Classification Error for Model 1 (Right,Front,Left)

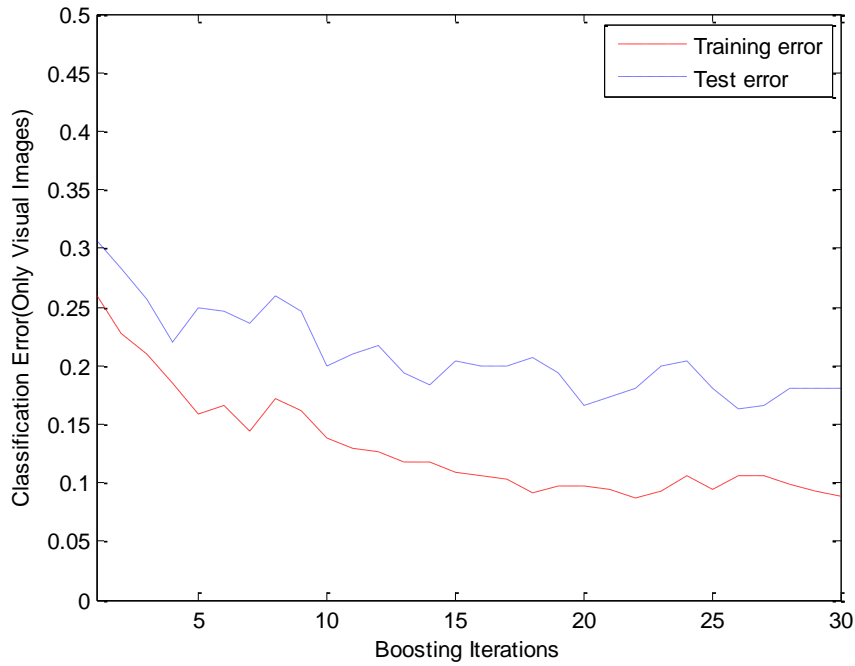


Figure 5.3: Classification Error for Model 2 (Up, Front, Down)

Pose Type	For Training (%)	For Testing (%)
Right	3.3	9.0
Front	1.0	12.0
Left	4.0	8.0
Up	5.0	10.0
Down	11.3	11.3

Table 5.3 Classification Error for each pose

The overall classification rate is given by

Classification rate:  $C_{rate} = 1 - C_{error}$

The overall percentage of classification rate for only visual images given below, which is calculated from the individual class error stated in Table 5.3. False alarm rate for model 1 is also calculated for all the poses and shown in Table 5.3.

	$C_{rate}$ (%)	False alarm (%)
<b>Training</b>	95.08	4.16
<b>Testing</b>	89.4	4.83

Table 5.4: Overall classification rate and false alarm rate

### 5.2.2 Classification using Thermal IR images

The experiment is now conducted with only the thermal IR images. Some of the training images are captured with low lighting conditions, which could result in misclassification in the visual image classification. However, in the case of thermal IR images, this does not affect it. This is evident from the classification error shown in Figure 5.4: Classification Error for Model 1 (Right, Front, Left). The classification rate has improved compared to the classification rate of the visual classifier.

The error response of model 1 in Figure 5.4 shows that the error has dropped to 0.01, while the error response of model 2 in Figure 5.5 shows that the error is 0.2. The reason for the high error in model 2 is because of the usage of similar classes. Front, up, and down classes are considered for this model. Due to some bad samples in these classes, the error rate is increased. Analysis of this issue shows that the Front class constitutes 90% of this error. Hence, the estimation of the Front class is ignored in this model. Only samples of up and down classes are tested with model 2.

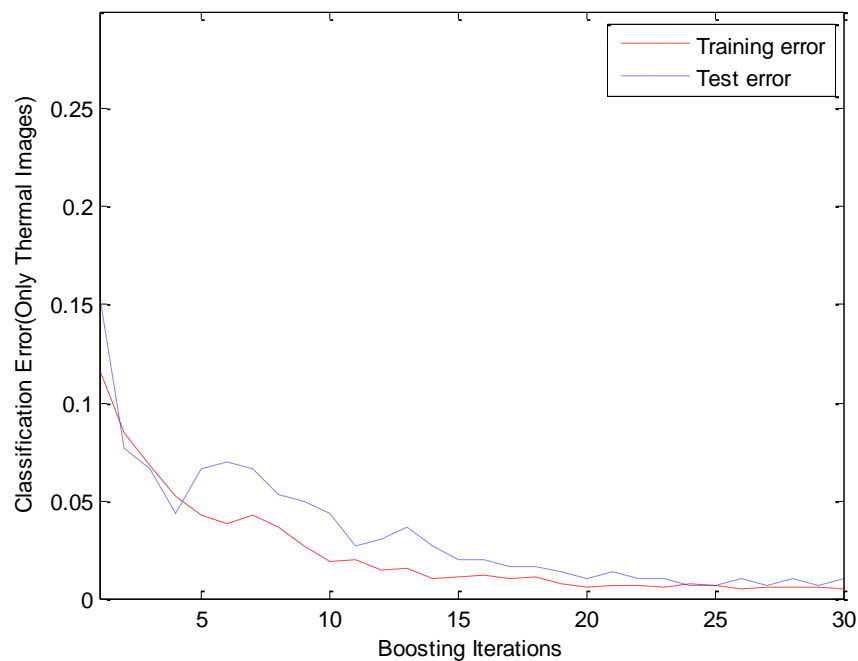


Figure 5.4: Classification Error for Model 1 (Right, Front, Left)

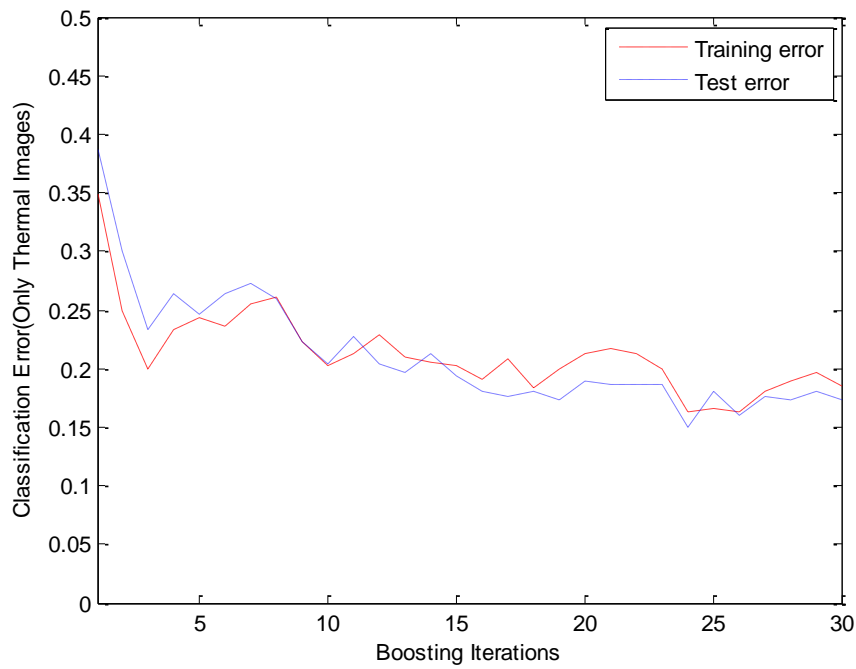


Figure 5.5: Classification Error for Model 2 (Up, Front, Down)

The individual classification error for each class is listed in Table 5.5.

Pose Type	For Training (%)	For Testing (%)
Right	1.0	2.0
Front	0	1.0
Left	0.3	0
Up	5.6	4.0
Down	1.0	1.0

Table 5.5: Classification error for each pose

The overall classification rate and false alarm rate for an only thermal IR image is given in Table 5.6.

	$C_{rate}$ (%)	False alarm (%)
Training	98.42	0.66
Testing	98.4	0.5

Table 5.6: Overall classification rate and false alarm rate

### 5.2.3 Fusion of Visual and Thermal IR Classification

The efficiency of the classifier drastically improves after using both the visual and the thermal images for the classification purpose. Both have unique features to classify the images. Hence the fusion results in better performance than the individual classification results. The classification error response for training and testing is given in Figure 5.6.

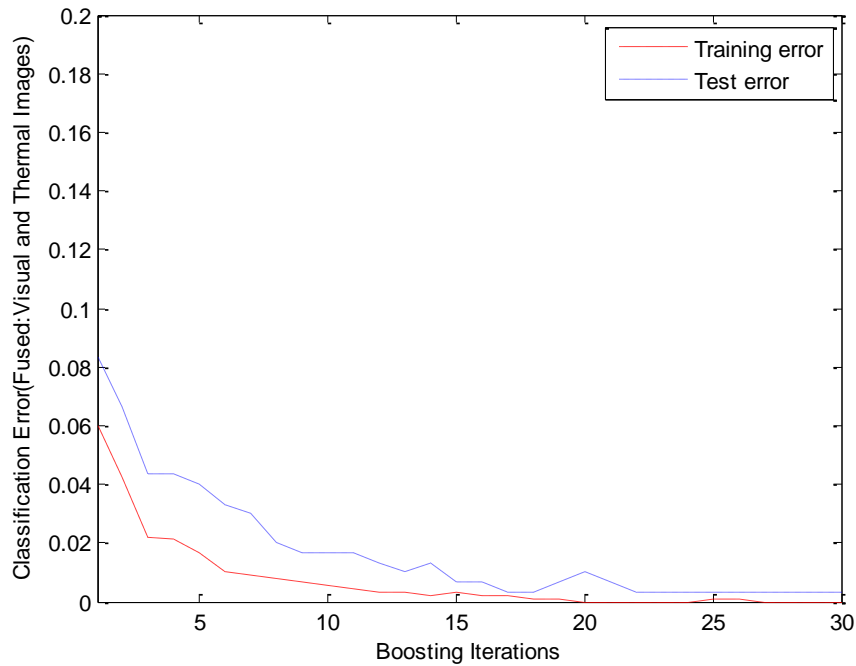


Figure 5.6: Classification Error for Model 1 (Right, Front, Left)

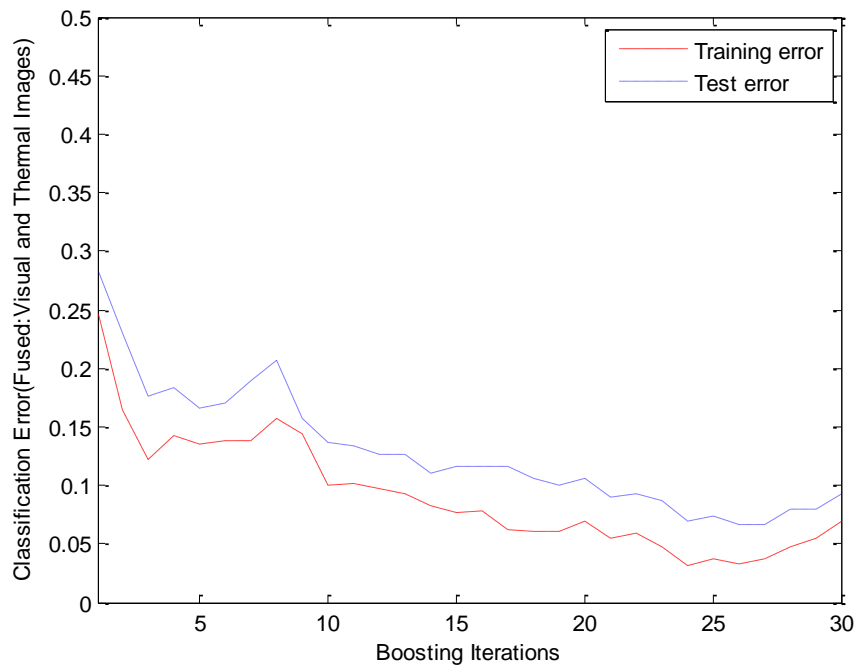


Figure 5.7: Classification Error for Model 2 (Up, Front, Down)

The fusion result for the individual class is listed in Table 5.7.

Pose Type	For Training (%)	For Testing (%)
Right	0	0
Front	0	1.0
Left	0	0
Up	1.3	2.0
Down	0.3	0.3

**Table 5.7: Classification Error for each pose**

Fusion of visual and thermal images has improved the classification rate for both training and testing. The overall classification and false alarm rate for fusion of the visual classifier and the thermal IR classifier is given in Table 5.8

	$C_{rate}$ (%)	False alarm (%)
Training	99.7	0
Testing	99.34	0.16

**Table 5.8: Overall classification rate and false alarm rate**

### 5.3 Evaluation

The performance of the fusion of the visual images and the thermal IR images can be evaluated by comparing their individual classification and the fused classification results. The comparison chart in Figure 5.8 and Figure 5.9 shows their performance.

It is clear from the comparison, that for all the different poses, the fusion results in low classification error than the individual classification error. The overall classification rate has increased by using both visual and thermal classifiers. The classification rate improvement is shown in Table 5.9.

	Visual (%)	Thermal IR (%)	Fused (%)
Training	95.08	98.42	99.7
Testing	89.4	98.4	99.34

**Table 5.9: Comparison of classification rate of Visual, thermal and fused classifiers.**



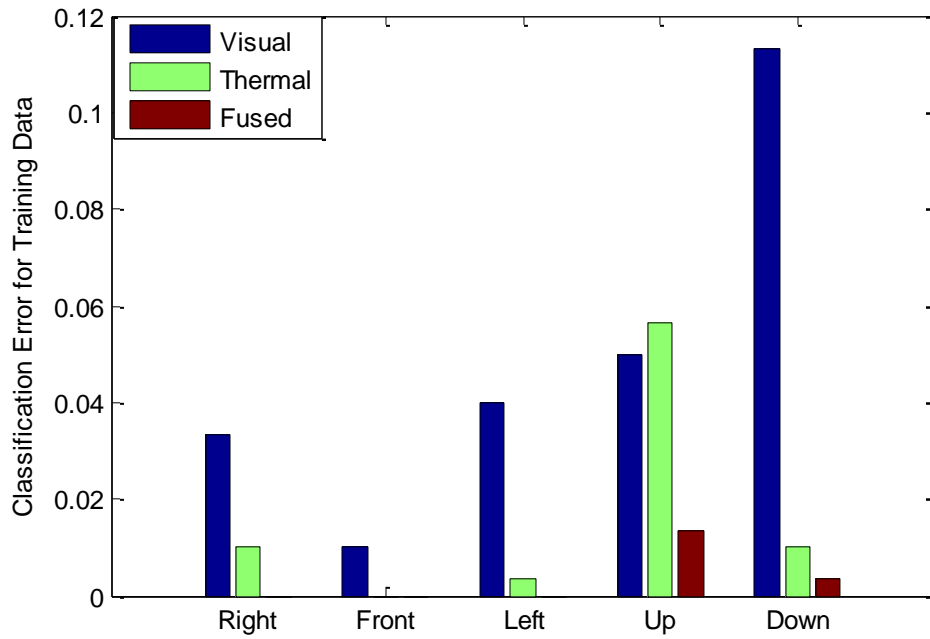


Figure 5.8: Training Error Comparison of Visual, Thermal and Fused images of all pose

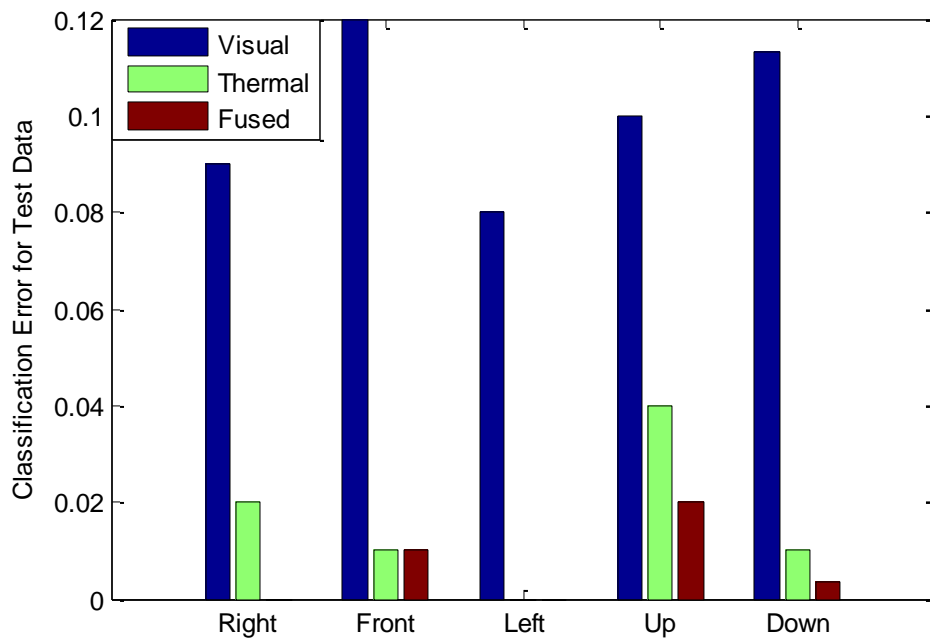


Figure 5.9: Testing Error Comparison of Visual, Thermal and Fused images for all pose

## 5.4 Demo Experiment

The trained classifier model is used to classify the test data. A demo experiment is conducted during the presentation of this thesis. Kinect is used to capture the test image. Only Visual images are considered for this demo. The model is also evaluated with many complex scenarios like occlusion and the results are impressive. Some of the samples estimation of the classifier is shown in Figure 5.10



Figure 5.10: Sample classified image by multiclass AdaBoost

## 6 Conclusion

In this thesis fusion based multiclass classification of visual, thermal IR and NIR images is investigated. The first part is the consideration of images from different spectrum for fusion. For this purpose, there different equipment is experimented. Thermal camera is used for capturing both the visual and the thermal images. The self-made NIR webcam is used for capturing the Near-Infrared images and the visual images. The kinect is interfaced to use its inbuilt camera to obtain the visual image, NIR image and the depth image. In the second part, these images are used for multiclass classification. The proposed multiclass AdaBoost is used as a classifier. The individual classification of the visual and the thermal images are done, and the classification rate is 89% and 98% respectively. The fusion of both these classifier is performed to improve the classification and the resulting classification rate is 99.3%. Thus the fusion indeed improved the classification rate, however due to the restricted use of thermal camera for commercial application, alternative method is suggested. The NIR images captured from the Kinect and the NIR webcam, shows that the these images are invariant to light, and thus has the advantages of thermal images. Thus the fusion of visual and NIR images enhances the classification performance.

### 6.1 Future Work

The experiments and results always pave way to some more improvement. In this thesis, after investigation for the fusion of the visual and thermal IR images, lot of improvements and future work is suggested.

The classifier technique used can be improved by using a strong weak learner. Hence instead of using haar-like features, some intense feature extraction methods like HOG, can be used to improve the performance. The multiclass AdaBoost classifier used in this thesis is constrained to three class due to weak 'weak-learner', this can be extended for 5 or more class classification with the use of a strong 'weak-learner'. The weighted fusion of the individual visual and thermal classifier can be improved by use of feature level fusion. Only useful features of both the visual and thermal classifiers can be fused, and enhance the classification result. The training of the classifier is a time-consuming process. This was evident during the training process for the thesis. One way to avoid it is using online learning. Online learning applied to the current classifier will greatly improve the performance.

The thermal IR cannot be afforded for all the applications; hence NIR was suggested as an option. Due to the unavailability of the NIR database for different poses, the classifier was not tested on NIR database, which can be done in future to evaluate its performance.

Kinect can be used for real-time classification, as it has inbuilt NIR and visual camera. The classifier can be extended to classify for different scenarios like traffic light classification, human behavior classification, expression classification. All the experiments were conducted from still images which can be extended for video. Video tracking using both visual and thermal images can be studied. Also 3D modeling and tracking using multiple kinect can be investigated.

The whole thesis work involves training and testing a static image. The improvement can be made by shifting from images to classification in video. The same algorithm can be applied to video, where each frame needs to be processed individually. The information in the successive frames should be considered for good classification in video. The classification of object can be extended to detection and classification of object. As of now, the classifier can only classify a detected object. Building a detection algorithm into it will have wide usage in many applications.

## 7 Bibliography

- [1] S Singh, A Gyaourova, G Bebis, and I Pavlidis, "Infrared and visible image fusion for face recognition," , vol. 5404, 2004, pp. 585--596.
- [2] M Hanif and U Ali, "Optimized visual and thermal image fusion for efficient face recognition," , 2006, pp. 1--6.
- [3] Y Chen and R S Blum, "Experimental tests of image fusion for night vision," , vol. 1, 2005, pp. 8--pp.
- [4] S Z Li, R F Chu, S C Liao, and L Zhang, "Illumination invariant face recognition using near-infrared images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 627--639, 2007.
- [5] R E Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, pp. 197--227, 1990.
- [6] Y Freund and R Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," , 1995, pp. 23--37.
- [7] P Viola and M Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 57, pp. 137-154, 2002.
- [8] N Dalal, "Finding people in images and videos," *PHD's thesis, Institut National Polytechnique de Grenoble*, 2006.
- [9] Q Zhu, M C Yeh, K T Cheng, and S Avidan, "Fast human detection using a cascade of histograms of oriented gradients," , vol. 2, 2006, pp. 1491--1498.
- [10] W Iba and P Langley, "Induction of one-level decision trees," , 1992, pp. 233--240.
- [11] J R Quinlan, *C4. 5: programs for machine learning.*: Morgan Kaufmann, 1993.
- [12] Peter E.Hart,David G.Stork Richard O.Duda, *Pattern Classification*, Second ed.
- [13] R E Schapire, "The boosting approach to machine learning: An overview," *LECTURE NOTES IN STATISTICS-NEW YORK-SPRINGER VERLAG-*, pp. 149-172, 2003.
- [14] Rob Schapire, "Tutorial on Theory and Application of Boosting," in *Neural Information Processing Systems*, 2007.

- [15] R E Schapire and Y Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine learning*, vol. 37, pp. 297--336, 1999.
- [16] Manfred K. Warmuth, "Tutorial on Survey of Boosting from an Optimization Perspective," in *International Conference on Machine Learning*.
- [17] wikiHow: Make a Webcam into Imfrared Camera. [Online].  
<http://www.wikihow.com/Make-a-Webcam-Into-an-Infrared-Camera>
- [18] Xbox. [Online]. <http://www.xbox.com/en-US/kinect>
- [19] KinEmote - Free XBox Kinect Software. [Online]. <http://www.kinemote.net>
- [20] Xbox Kinect in Operating room. [Online]. <http://sunnyview.sunnybrook.ca/2011/03/fun-and-games-in-or.html>
- [21] Delta-Parallel-Roboter "deltaR". [Online].  
<http://www.mtahlers.de/index.php/robotik/delta-roboter-qdeltarg>
- [22] Controlling the Quadrotor. [Online]. [http://www.idsc.ethz.ch/Research\\_DAndrea/FMA](http://www.idsc.ethz.ch/Research_DAndrea/FMA)
- [23] "Holographic projection" using Kinect SDK. [Online].  
<http://www.youtube.com/watch?v=9xMSGmjOZlg>
- [24] Encumbrance-free Telepresence System. [Online].  
<http://www.cs.unc.edu/~maimone/KinectPaper/kinect.html>
- [25] Wi-Go Project. [Online]. <http://vimeo.com/24542706>
- [26] Teleoperation of a 5-DOF Robotic Arm. [Online].  
<http://embeddedzone.blogspot.com/2011/05/teleoperation-of-5-dof-robotic-arm.html>
- [27] NAVI – Navigational Aids for the Visually Impaired. [Online]. <http://hci.uni-konstanz.de/blog/2011/03/15/navi/?lang=en>
- [28] Improved Humanoid Robot Teleoperation with NAO and Kinect. [Online].  
<http://www.youtube.com/watch?v=TmTW61MLm68>
- [29] iRobot AVA. [Online]. <http://www.irobot.com/ava/>,  
<http://www.youtube.com/watch?v=UmuQYoJJDas>
- [30] Optical camouflage. [Online]. [http://www.youtube.com/watch?v=4qhXQ\\_1CQjg](http://www.youtube.com/watch?v=4qhXQ_1CQjg)

- [31] Real-time 3D Tracking, Reconstruction, and Interaction. [Online].  
<http://research.microsoft.com/en-us/projects/surfacerecon/default.aspx>
- [32] KinectShop – The Next Generation of Shopping. [Online].  
<http://emergingexperiences.com/2011/06/kinectshop/>
- [33] Kinect Hacking Contest. [Online]. <http://www.adafruit.com/blog/2010/11/10/we-have-a-winner-open-kinect-drivers-released-winner-will-use-3k-for-more-hacking-plus-an-additional-2k-goes-to-the-eff/>
- [34] Kinect for Windows SDK. [Online]. <http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/>
- [35] OpenNI. [Online]. <http://www.openni.org/downloadfiles/opennimodules>
- [36] SensorKinect. [Online]. <https://github.com/avin2/SensorKinect>
- [37] Kinect for MATLAB. [Online]. <http://sourceforge.net/projects/kinect-mex/>
- [38] Babak Rasolzadeh and Josephine Sullivan, "DD2427 - Image Based Classification and Recognition," Royal Institute of Technology, Stockholm, Lab Manual 2010.
- [39] I Mukherjee and R E Schapire, "A theory of multiclass boosting," *Advances in Neural Information Processing Systems*, vol. 23, pp. 1714--1722, 2010.
- [40] The Facial Recognition Technology (FERET) Database. [Online].  
[http://www.itl.nist.gov/iad/humanid/feret/feret\\_master.html](http://www.itl.nist.gov/iad/humanid/feret/feret_master.html)
- [41] M Sonka, V Hlavac, and R Boyle, "Image processing, analysis, and machine vision," *CL-Engineering*, 2007.