FAST GAUSSIAN DISTRIBUTION BASED ADABOOST ALGORITHM FOR FACE DETECTION

Tuan M. Pham[1], Hao P. Do[2], Danh C. Doan[2], Hoang V. Nguyen[2]

[1]University of Science and Technology - The University of Danang; pmtuan@dut.udn.vn
[2]Hippo Tech Vietnam; {haodophuc, danhdoan.es, nguyenviethoang.25}@gmail.com

Abstract - In the past few years, Paul Viola and Michael J. Jones have successfully developed a new face detection approach which has been widely applied to many detection systems. Even though the efficiency and robustness are proved in both performance and accuracy, there is still a number of improvements that we can apply to enhance their algorithm. This paper inherits face detection framework of Viola-Jones and introduces two key contributions. First, the modification is used to apply integral image so that features are more informative and help increase detection performance. The second contribution is the new approach to utilize AdaBoost that uses Gaussian Probability Distribution to compute how close to the mean positive and negative distributions are, then classify them more efficiently. Furthermore, by experiments, we also prove that a small fraction of a feature set is far enough to develop a good strong classifier instead of the whole feature set. As a result, the memory required as well as the time for training is minimized.

Key words - face detection; Gaussian distribution; AdaBoost; Haar-like pattern; weak classifier

1. Introduction

In recent decades, along with the rapidly advanced improvement in technology, face detection has now become the most popular topic that can be applied to many fields in industries or in real life. Algorithms for face detections are developed quickly and become more enhanced to support complicated applications like multi-view face detection [1-4], occluded face detection [3, 5], pedestrian detection [6, 7], ... In this paper, we inherit the work of Viola-Jones [8] which has been proved successful in accuracy as well as in performance.

Thanks to their great work, the number of practical real-time applications and systems are built for face detection or related topics. We will propose some new methods for feature extraction and implementation so that the system can train and detect images faster than previous one from Viola-Jones and it also utilizes less memory storage. Besides, new Haar-like patterns are proposed to improve the efficiency of detection.

There are two main contributions in our face detection systems and they are briefly introduced below and in detail in next sections.

First, we utilize integral image representation as the main component to quickly compute feature values. Nevertheless, it is more efficient when applying non integer-sized pattern as described in section 3. In this way, given a feature, we can obtain much more information that is necessary for classification process. In this step, Viola-Jones system pre-calculates feature values and stores them in hard drive. This method is useful in training process because all information is already computed. In contrast, the significantly long time is used for this calculation and working with hard drive. Instead of following their method, we introduce another approach. Given the size of a data set, size of image as well as the features patterns, a lookup table used for feature indexing can be generated separately before training procedure proceeds. We have to compute a specific feature value when needed. It is more efficient than the previous work [8] not only in performance but also in memory consumption.

The second enhancement is how AdaBoost [9] is applied. Viola-Jones used threshold to classify positive and negative example images. Multiple weaker classifiers are combined to form a strong one which can divide data perfectly when learning, but in testing, it might fail. The detection quality depends much on the correctness of the AdaBoost function to classify data. In case the positive and negative distribution overlap, it is apparently difficult to choose a good threshold between those regions. In this paper we apply Gaussian probability distribution for the classification task. Why we use and how we apply this method to AdaBoost process is described in this section, and pseudo-code is also provided. Besides, the number of operations for Gaussian is less than using threshold, thus the training and detection time is reduced.

a. Overview

We start by review Viola-Jones systems and point out some functions that we need to improve. The review is in section 1. In section 2, we describe and explain how we choose and implement our new algorithms and why they are effective in face detection system. After that, the experiments and results are clearly shown in Section 3.

2. Review of Viola-Jones Algorithms

a. Haar-like patterns

In every vision system, the efficiency and accuracy depend strongly on the features it uses and the quality of those features. Feature design and its calculation is the key to the success of a computer vision or machine learning system. To extract features from example images, Viola-Jones used Haar-like rectangle features in their systems as shown in Figure 1.

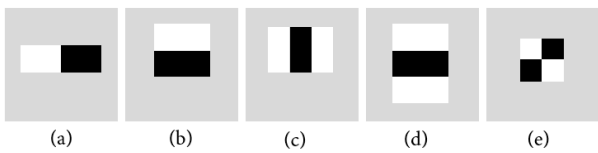

Figure 1. *Haar-like rectangle patterns in Viola-Jones system*

Feature value for a certain rectangle is the difference between white and black pixel values. If doing this task by common methods, it would take the complexity of $O(HW)$ where H and W are height and width of a pattern.

Integral image representation is one of their contributions in their paper. It is applied to rapidly compute feature values. Its formula is described below:

$$ii(x,y) = \sum_{0 \le x' \le x, 0 \le y' \le y} i(x',y')$$

Where $i(x,y)$ is the image intensity at pixel (x,y) and $ii(x,y)$ is the value of integral image at pixel (x,y).

By using integral image method, we are able to calculate any rectangular sum by pre-computed referenced rectangles. Thus, the complexity is approximately *O*(*1*).

In their detection system, they trained and tested by 24x24 Grayscale PNG images. For each image, they applied those 5 rectangle features. Heights, widths and positions of each rectangle are also varied. Because information of images and feature patterns are given, the number of features which can be applied to an image is known. There were 43200, 43200, 27600, 27600 and 20736 features for each rectangle of category (a), (b), (c), (d) and (e) respectively, thus 162336 features in total.

b. AdaBoost algorithm

From that, there were a huge number of features corresponding to each image sub-window. Due to this fact, it is still a lot of work even when those features are calculated quickly and efficiently. However, by using a very small set of features, detection system can form an effective classifier.

Thanks to the invention of AdaBoost [9], this method can be used to select the essential features as well as to train for a strong classifier. AdaBoost is an algorithm for constructing a strong classifier from a linear combination of weak classifier.

$$F(x) = \sum_{t=1}^{T} a_t h_t(x)$$

Where x is an example (19x19 image in our system), $h_t(x)$ is a weak or basis classifier. Normally, the set $\mathcal{H} = h(x)$ is finite.

A weak classifier is a function of a feature (*f*), a threshold (θ) and a polarity (p) that denotes the direction of the inequality:

$$h(x,f,p,\theta) = \begin{cases} 1 & p.f(x) < p.\theta \\ 0 & otherwise \end{cases}$$

For a weak learner, we do not expect the best classification. After each round of AdaBoost, a weak classifier with a smallest weighted error is chosen:

$$\hat{h}_t = \arg \min_{h_j \in H} \varepsilon_j = \sum_i w_i \left| h_j(x_i,f,p,\theta) - y_i \right|$$

Where y_i is the correct label for example x_i, y_i is 1 if x_i represents a face, otherwise it is 0.

In additions, every example is re-weighted so that it is emphasized in the next training round. Clearly, an example which is incorrectly labeled in the current round will have the greater weight compared to correct ones.

$$w_{t+1,i} = w_{t,i} \beta^{1-e_i}$$

Where $e_i = 0$ if x_i is correctly classified, otherwise it is 1, and $\beta = \frac{e_i}{1-e_i}$

AdaBoost is very simple to implement and efficiently extract good features from a very large set. One of the disadvantages of AdaBoost algorithm is that over fit is the result of choosing a very complex-training model, turning this to the key challenge to applying this method.

Graphic visualization of AdaBoost process after each round is shown in Figure 2 below. In this example, we need to detect and classify blue dots from red ones. We apply AdaBoost to this problem and try to find the best weak classifier to classify these two regions in each round.

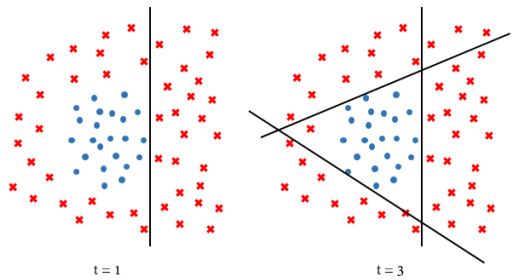

Figure 2. *Visualization for AdaBoost process after t=1 and t=3*

After completing the first round, 1 weak classifier is chosen as described by the black straight line. The total detection quality is now very low. However, by combining 3 weak classifiers, the accuracy is significantly improved.

In Viola-Jones system, they used AdaBoost and for each weak learner, tried to find the optimal threshold classification function. Supposing that there are *N* image examples and *K* features for each image, so they had *KN* combinations for feature and threshold. For the data set used by Viola-Jones, *K* was 162336 features and *N* was 6977 images to train. In a training round of AdaBoost procedure, it needs to iterate the whole data set to evaluate the training error for a feature/threshold combination. It means, the complexity required for each round was *O*(*NKN*) to find a weak classifier. By setting the number of weak classifiers to *M*, the total complexity to train their system is *O*(*MNKN*). With *M* = 200, at least 1.58×10^{15} operations needed to be processed in any training machine. Even when working with a super computer, it is still a tremendous procedure and takes a significantly long time to finish.

To improve the system as well as reduce the training time, they proposed the modified algorithm. With a specific feature, they could find the optimal threshold by using current example weights without generating all possible combinations of this feature and every image example. To apply this algorithm, with each feature, examples should be sorted by their feature values, and the complexity for this process is *O*(*Nlog*$_2$*N*). Thereafter, it only requires *O*(*N*) to find the optimal threshold for current feature. Hence, the complexity to this sub-task is *O*(*max*(*N*, *Nlog*$_2$*N*)) = *O*(*Nlog*$_2$*N*). This algorithm led to the reduced complexity of *O*(*MKNlog*$_2$*N*) and sustainable decrease in the number of operations to 2.89×10^{12}.

Pseudo-code for Viola-Jones' algorithm is shown as below:

1. Giving example image $(x_1, y_1), \dots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
2. Initializing weights $w_{t,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
3. For $t = 1, \dots, T$:
   a. Normalizing the weights $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_j^n w_{t,j}}$
   b. Selecting the best weak classifier with respect to the weighted error
   $$\varepsilon_t = \min_{f,p,\theta} \sum_i w_i |h(x_i, f, p, \theta) - y_i|$$
   a. Defining $h_t(x) = h(x, f_t, p_t, \theta_t)$ where f_t, p_t, and θ_t are the minimizers of ε_t
   b. Updating the weights:
   $$w_{t+1,i} = w_{t,i} \beta^{1-e_i}$$
   Where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta = \frac{e_i}{1-e_i}$
4. Combining strong classifier

3. Proposed Method to Improve Viola-Jones system

The main purpose of this paper is to propose the new method that can perform detection faster than the traditional Viola-Jone system, so we choose the same Haar-like rectangle features in their system. Besides, we also introduce our new features which are more efficient for face detection systems when the complexity level increases, such as detecting rotated faces.

3.1. New feature selection

In Viola-Jones system, they used integer-size of rectangle. In our research, we again use those rectangle but with non-integer size. With this method, features can be more informative, thus the detection performance is higher than that by the Viola-Jones system. Figure 3 shows some examples of new non integer-sized rectangle. This is 2x2 sub-window from an image and the height of rectangle feature is 3.

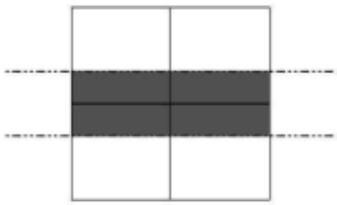

Figure 3. *2x2 sub-window image with non- integer-sized feature*

Because the size is a non -integer number, feature values are represented by floating-point number. It results in new difficulties when systems use complicated pattern of features. For this problem, we also figure out the approach which can quickly compute the feature values in few operations with the complexity of *O(1)*. Compared to the traditional feature calculation, processing time is now nearly the same.

To apply new rectangle features, users only need to clearly specify new pattern before training their models. Below is an example for pattern (d) in Figure 1. The matrix shows color map of features with 1 for white and -1 for black.

1	*1*	*1*
-1	*-1*	*-1*
1	*1*	*1*

By using the color map above, we can quickly compute the feature value for each rectangle with non-integer size.

The size and position of a feature can vary correspondingly to the 19x19 image. Given the size of a pattern, we can manage the number of arising features. This point leads to another improvement for our system, that is pre-calculating and storing for feature values are no longer required. Back to Viola-Jones approach, they have to use hard drive to store the whole set of feature values. Apparently, the way costs a tremendously long time to get the training data available. Besides, it consumes huge memory storage which is now not essential in our system.

There are 29241, 29241. 23409, 23409 and 29241 features for category (a), (b), (c), (d) and (e) respectively.

Thanks to the constraint of rectangle sizes, we separately make a lookup table from the given information about those rectangles. It means that when it requires any single feature value, our system can immediately find the exact feature as well as its size and position. After that we easily compute them by our formula as we have mentioned before. To generate this lookup table, we just apply brute-force algorithm to iterate and find feature information.

3.2. Gaussian distribution as classification function

Even though the training time is significantly reduced, it is still a long time. In our research and experiment, we propose a new way to train our detection system by applying Gaussian probability distribution instead of finding optimal threshold for each feature.

Starting with the point that positive and negative distribution of Haar-like feature for an image are very hard to classify by a single threshold. By combining multiple weak classifiers with many thresholds, the number of operations exponentially increases without guaranteeing the increase of detection accuracy. Besides, it can result in over-fitting problem on the training process.

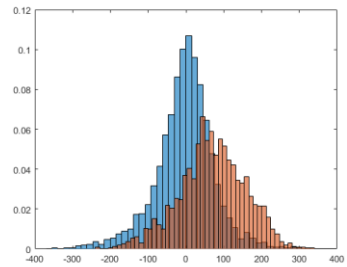

Figure 4. *Histogram of a specific feature for face and non-face images*

Figure above shows histogram of feature values for face and non-face images computed from a specific feature. Blue region denotes feature values for face images, and non-faces are drawn in orange. The x-axis is the feature values; meanwhile y-axis shows the frequency after normalization of each value. From the figure, it is clear that 2 histograms are overlapped, leading to the difficulty to select a threshold. Moreover, in those situations, a weak classifier's performance is poor but the training time is longer and finally the testing result is poor as a consequence.

In this paper, we propose the AdaBoost algorithm that uses Gaussian distribution of feature values. Gaussian distribution is one of the most important probability distributions for continuous variables and it is really useful in natural sciences. From theory, the averages of samples of a variable from independent distributions converge in distribution to the normal. In other words, it becomes normally distributed when we have enough observations. Below is the formula for a Gaussian distribution of a single real-valued variable x:

$$f_g(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Where μ is the mean and σ^2 is the variance of the sequence of feature values for the data set with a specific feature.

To overcome this overlapping problem, we apply Gaussian distribution to calculate and compare the difference to 2 means of positive and negative distributions. The classification function is applied as follows:

$$h(x,f,\mu_p,{\sigma_p}^2,\mu_n,{\sigma_n}^2) = \begin{cases} 1 & f(x|\mu_p,{\sigma_p}^2) > f(x|\mu_n,{\sigma_n}^2) \\ 0 & otherwise \end{cases}$$

Where $f(x|\mu,\sigma^2)$ is the Gaussian function that is mentioned above. x is a feature value of an example image. $\mu_p, {\sigma_p}^2, \mu_n,$ and ${\sigma_n}^2$ are means and variances for positive and negative distributions respectively.

Our method proceeds as follows. For each feature, all corresponded values to image examples are computed at a given feature. Next , we calculate means and variances for 2 distributions. After that, formula of Gaussian distribution is applied to find the distance to the means before comparison. If an image example is more likely to be a face, so the closer it is from the means of a positive region. Otherwise, it is closer to negative distribution. With this method, the difficulty of overlapping is overcome because the means of distributions are all always separated.

After finishing the current classification, error is computed to select the best feature of the current AdaBoost round. Clearly, the weights of image examples are re-computed to emphasize incorrect classification for later training.

Our procedure is described in the pseudo-code below. The only difference between our algorithm and Viola-Jones' is the classification function with Gaussian method.

1. Giving example image $(x_1, y_1), \dots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
2. Initializing weights $w_{t,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
3. For $t = 1, \dots, T$:
   a. Normalizing the weights $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_j^n w_{t,j}}$
   b. Selecting *K'* features from full set of features
   c. For each feature:
      i. Computing feature values for every example
      ii. Calculating means and variances for positive and negative distributions.
      iii. Selecting the best weak classifier that minimizes the error:

$$\varepsilon_t = min_f \sum_i w_i \left| h\left(h(x_i, f, \mu_p, {\sigma_p}^2, \mu_n, {\sigma_n}^2\right) - y_i \right|$$

      iv. Defining $h_t(x) = h(x, f_t)$ where f_t is the minimizer of ε_t
   d. Updating the weights:

$$w_{t+1,i} = w_{t,i}\beta^{1-e_i}$$

   Where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta = \frac{e_i}{1-e_i}$
4. Combining strong classifier

In our method, it takes linear time to compute mean and variance for each positive or negative distribution with complexity of *O*(*N*) and all simple operations. *O*(*MKN*) is the total complexity for our algorithm. However, due to the use of Gaussian distribution, the floating-point operations are obligated. Indeed, exponential operation for floating-point numbers is far complicated than simple arithmetic ones. In our system, this expression is used to find e^x for classification function $f\left(x|\mu_p, {\sigma_p}^2\right)$. Although the current complexity is *O*(*MKN*), it costs even longer time than *O*(*MKNlog*$_2$*N*) does if those operations are not well computed. Thus, for this step, instead of exponentials we use inverse operation which is natural logarithm $ln(x)$. By our experiment, it takes much less time to compute $ln(e^x)$ compared to e^x. By simplifying expressions to find Gaussian function, the number of operations is also remarkably reduced.

If an image is a face, ratio of Gaussian functions for two distributions should be greater than 1:

$$\frac{\frac{1}{\sqrt{2\pi{\sigma_p}^2}} e^{\frac{(x-\mu_p)^2}{2{\sigma_p}^2}}}{\frac{1}{\sqrt{2\pi{\sigma_n}^2}} e^{\frac{(x-\mu_n)^2}{2{\sigma_n}^2}}} > 1$$

or,

$$\frac{\sqrt{{\sigma_n}^2}}{\sqrt{{\sigma_p}^2}} > e^{\frac{(x-\mu_p)^2}{2{\sigma_p}^2} - \frac{(x-\mu_n)^2}{2{\sigma_n}^2}}$$

Because all elements are non-negative numbers, the inequality remains unchanged when taking square of both sides:

$$\frac{{\sigma_n}^2}{{\sigma_p}^2} > e^{\frac{(x-\mu_p)^2}{2{\sigma_p}^2} - \frac{(x-\mu_n)^2}{2{\sigma_n}^2}}$$

Moreover, natural exponential is a one-to-one and increasing function, we can apply natural logarithm to the inequality:

$$\ln\left(\frac{{\sigma_n}^2}{{\sigma_p}^2}\right) > \frac{(x-\mu_p)^2}{2{\sigma_p}^2} - \frac{(x-\mu_n)^2}{2{\sigma_n}^2}$$

At this point, we can use the above inequality to make comparison; all of those operations can be calculated in a short time. We have reduced a lot of operations and the computation is now much simpler.

The following shows pseudo-code for our approach:

1. Computing feature values for the whole data set
2. Finding mean μ and variance σ^2 for positive and negative distributions
3. Using natural logarithm to find: $a = \ln(\frac{{\sigma_n}^2}{{\sigma_p}^2})$
4. Finding the value of $b = \frac{(x-\mu_p)^2}{{\sigma_p}^2} - \frac{(x-\mu_n)^2}{{\sigma_n}^2}$
5. Comparing a and b, return 1 if $a > b$, otherwise 0

4. Experiments and Results

a. Experiments

As mentioned before, to train and test this system, we use data set from MIT cbcl Face Data [10] and it contains 19x19 grayscale PGM images.

In this procedure, we conduct some experiments to evaluate and compare the processing time between original method with our proposed one. Before preparing lookup table and training, we normalize the whole data set for both train and test images [11] Thus, those images are now in the same standard. Samples of images before and after the normalization process are listed below:

Figure 5. *Original images before normalization*

Figure 6. *Images after normalization*

In the experiment, we implement our algorithm from scratch in Java. After that, the system runs in a normal computer with Mac OS, memory 8 GB 1600MHz DDR3 and processor 2.6 GHz Intel Core i5. The amount of hard drive is not specified due to the fact that we only use Ram to experiment our system. It means, hard drive is not mandatory as in Viola-Jones's.

b. Results

Theoretically, for each round of AdaBoost process, there are totally 134541 features used for testing to choose the best weak classifier. By analysis and experiments, it is unnecessary to test all of those features, instead, a small number of them can be used to reduce the training time but still maintain the accuracy level. From experimenting different numbers of features for training each round, we have found that K' = 5000 features are sufficient for 2 criteria above.

The graph in Figure 7 shows the comparison between choosing K' = 5000 features versus the whole 134541 features. Both of two AdaBoost algorithms that are used Viola-Jones with threshold and proposed method with Gaussian distribution are involved. Viola-Jones' approaches are figured by blue and gray dashed lines.

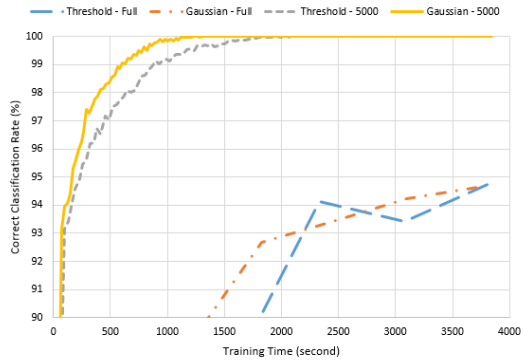

Figure 7. *Processing time and accuracy between 2 methods with different number of chosen features on training image set*

In this experiment, we do not follow Viola-Jones method which requires hard drive to store feature values due to the long processing time with memory storage. Hence, in our proposed method, lookup table is utilized to reduce the training time.

The x-axis is processing time measured in second and the y-axis is the corresponding accuracy by percentage. When Viola-Jones method using threshold is applied to classify face/non-face images, it takes approximately 760 seconds for a weak classifier if we test the whole set of features. In contrast, 30 seconds is needed if this procedure is performed by 5000 features. When applying Gaussian distribution, the processing time decreases. It requires 600 seconds for the full feature set and 22 seconds if 5000 features are chosen.

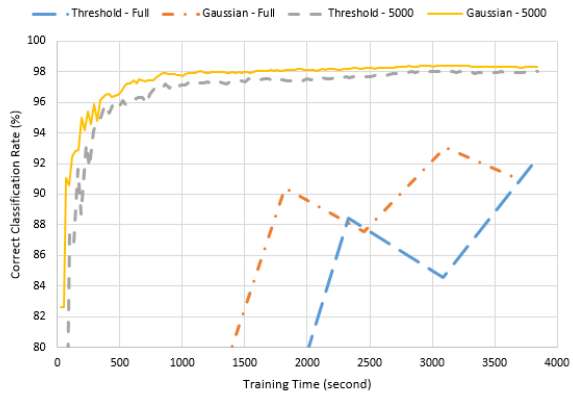

Figure 8. *Result from experimenting with testing image set*

By conducting this experiment, it is proved that the detection system can still obtain the high performance

without choosing the whole set of features. From the above figure, applying Gaussian distribution is better than original Viola-Jones method in both cases. This result is gained by testing with the training image set, we have the similar result with the testing image set and it is shown in Figure 8.

In those experiment processes, we compute the accuracy of detection in the fixed processing time of about 1 hour. With this period of time, AdaBoost by Viola-Jones' algorithm can produce 5 and 125 weak classifiers for the whole set and 5000 features set respectively. Similarly, 6 and 160 weak classifiers are chosen with Gaussian distribution algorithm.

We also conduct another experiment that sets the fixed value of T - the number of weak classifiers. For T = 200, if we apply Viola-Jones system that pre-compute all feature values and store to hard drive, then use those values for training, it takes 46 seconds to compute and choose 1 weak class. Approximately, 153 minutes or 2 hours 33 minutes is required for the complete training process. By setting the same value for T, but keep using threshold for AdaBoost procedure, our new system requires 30 seconds for each weak classifier even though our computation is more complicated by using floating-point numbers. The total training time is now about 96 minutes or 1 hours 36 minutes, 2/3 of the previous time if we compare that to Viola-Jones system.

The training time is significantly reduced if Gaussian probability distribution is applied. For each weak classifier, the processing time decreases to 22 seconds. Clearly, the training process costs half of the original time with 76 minutes or 1 hour 16 minutes.

By using Gaussian probability distribution, the number of operations is reduced and now the speed of training is 2 times faster than that of the previous work. However, in Viola-Jones method, they had to use hard drive to pre-compute training data and this process took a significant time as described in previous section. If this factor is taken into account, our new method is proved to be far efficient not only in processing time but also in memory usage.

5. Conclusion

In this paper, we propose a new way to apply Haar-like patterns as well as how to use integral image technique for computing feature values. For the same feature, much more informative values can be extracted and hence, detection rate is better as well.

The more important contribution is how we apply Gaussian probability distribution to AdaBoost to improve its performance. By utilizing this function, we can avoid the difficulty to choose optimal thresholds for each round of AdaBoost. Classification problem becomes simpler and more straightforward by determining how far to the mean positive and negative distributions are. Besides, the detection speed is also faster because of classification rate.

Those two contributions have been characterized into the success of our paper. By applying this system or this idea about implementation, face detection system can be run in a normal computer or machine. From the advance in performance, this method can be used in other real-time detection systems in practice.

Acknowledgement

This research was funded by Vietnam Ministry of Science and Technology Research Project in 2017-2018, No. CNTT-10

REFERENCES

[1] Bo Wu, Haizhou AI, Chang Huang and Shihong Lao, "Fast Rotation Invariant Multi-View Face Detection Based on Real Adaboost", *IEEE FGR'04*, 2004.

[2] Paul Viola, Michael J. Jones, "Fast Multi-view Face Detection", *Mitsubishi Electric Research Lab TR-2003-96*, 2003.

[3] Shengcai Liao, Anil K. Jain, and Stan Z. Li, "*A Fast and Accurate Unconstrained Face Detector*", 2015.

[4] T. Mita, T. Kaneko, and O. Hori, "Joint Haar-like Features for Face Detection", *ICCV* 2005.

[5] X. P. Burgos-Artizzu, P. Perona, "Robust Face Landmark Estimation Under Occlusion", *ICCV*, 2013.

[6] B. Leibe, E. Seemann, and B. Schiele, "Pedestrian Detection in Crowded Scenes", *CVPR*, 2005.

[7] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Towards Reaching Human Performance In Pedestrian Detection", *IEEE PAMI*, 2017.

[8] Paul Viola, Michael J. Jones, "Robust Real-time Face Detection", *International Journal of Computer Vision*, 2004, pp. 138-143.

[9] Robert E. Schapire, "Explaining AdaBoost", *In Empirical Inference*, 2013.

[10] CBCL Face Database. Retrieved from http://cbcl.mit.edu/software-datasets/FaceData2.html

[11] Dwayne Philips, "Image Processing in C 2nd". R&D Publications, 2000.

(The Board of Editors received the paper on 03/01/2018, its review was completed on 03/4/2018)