Enhanced Knowledge in Sciences and Technology Vol. 2 No. 1 (2022) 303-312
© Universiti Tun Hussein Onn Malaysia Publisher's Office

EKST

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/ekst
e-ISSN : 2773-6385

Face Detection with Haar Cascades Method

Yap Jia Hui[1], Lee Siaw Chong[1*]

[1]Department of Mathematics and Statistics, Faculty of Applied Sciences and Technology,
Universiti Tun Hussein Onn Malaysia, 84600 Pagoh, Johor, MALAYSIA

*Corresponding Author Designation

DOI: https://doi.org/10.30880/ekst.2022.02.01.033
Received 02 January 2022; Accepted 19 April 2022; Available online 1 August 2022

Abstract: Face detection is a process of finding and identifying human faces in digital images. It is widely applied in social media, digital services, security systems, robotic and other applications. A human frontal up-right face detector had constructed in this paper by Haar Cascade method using MATLAB. Input images are often scaled to obtain a variety of different size faces, rather than scaling the detection window. The integral image method is also used to reduce the computation time of features value. The relationship between the scale factor with accuracy and computational time had been studied in this paper. In overall, the scale factor increase will cause the false detection, detection rate, and computational time decrease. The scale factor does not affect false detection and computational time much when scale factor up to 1.1 and above. The suggested scale factor in this paper is 1.1 to achieve higher accuracy and shorter computation time.

Keywords: Human Face Detection, Haar Cascade Method, Viola Jones Detector, MATLAB

1. Introduction

Face detection, which is also known as facial detection, is an action of finding the location and the scale of any face from an image, a video, or a real-time application. It is widely applied in social media, digital devices, security systems, robotic and other applications. For example, the auto-focusing function on the camera, the face unlocks technology use face detection as the first step to detect whether any human face presents [1]. Face detection is a complex task for a machine as all human faces are different. In this paper, a human face detection system was developed for various purpose: face unlock system, face recognition attendance system.

Haar Cascades method is one of the face detection methods. It was introduced by [2]. There are many face detection methods introduced before the Haar Cascades method, but most of these methods were unsuitable to be applied in real-world situations due to the various unconstrained conditions, including the large variations in pose, facial expression, lighting conditions, etc. [3]. Fortunately, these conditions were handled by the Haar Cascades method.

**Corresponding author: sclee@uthm.edu.my*

Next, the basic knowledge about digital image and Haar Cascade Method for understanding on how face detection works. After that, the deeper explanation of the Haar Cascade Method had shown in part 2. The Haar Cascade Method introduced by [2] including the integral image method, normalization, detection window scaling, integration on multiple detections, and so on. Then, the output of the detector created had explained, and the results and discussions about the benchmarking of false detection, detection rate, and computational time against the scale factor.

1.1 Digital image

Face detection only works with digital images, which are the images that saved in an electronic device. One of the digital image types is the raster type, which means the image is formed by a series of pixels. Each pixel is a tiny square picture with a color that represented by a number.

1.2 Haar-like features

When a 24×24 pixels grayscale image is given for face detection, the image will be scanned with a 24×24 pixels detection window. Two types of detection windows are depicted in the upper row of Figure 1; the black-white rectangle in each of the detection windows is known as the Haar-like features. The lower row represents the face feature captured by the detection windows. The face features captured are

- eyes (middle column of Figure 1); by comparing the intensities of the eyes with the cheeks
- nose (right column of Figure 1); by comparing the intensities of the eye regions with the intensity across the bridge of nose

This is due to the region of human eyes is often darker than the region of the nose and cheeks. Other face features are captured by different Haar-like features. The Haar-like feature is also called as the rectangle feature.

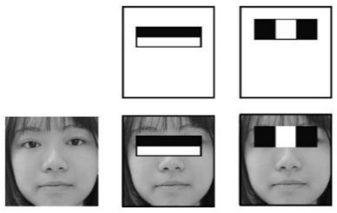


Figure 1: Example of scanning an image with detection window

When scanning the input image by Haar-like features, the sum of pixels value under white rectangle is subtracted from the sum of pixels value in black rectangle, and the value computed is the feature value. If the feature value calculated is less than the threshold determined in training process, the system will conclude that no feature exists as the feature given in that window.

Furthermore, if the input image is larger than 24×24 pixels, then the image will be scanned from top left corner to the bottom right corner through every row and column with a 24×24 pixels detection window. Then, scan the image at multiple detection window scales each a factor of 1.25 larger than the last.

To detect human face from image, more Haar-like features is needed as there are a lot of features on human faces. These features needed to be divided into multiple layers of classifier to construct a Haar Cascade.

1.3 Haar Cascades

The Haar Cascade that created by Viola and Jones has 38 layers of cascaded classifiers. The first five layers have 1, 10, 25, 25 and 50 features respectively, and the remaining layers have more features compared to its previous layer. The total number of features in the Haar Cascade is 6061, which selected from 180,000 features. The selection is done by AdaBoost algorithm that used to boost the classification performance by combining a set of weak classifiers (Haar-like features) in series to form a strong classifier. Figure 2 is a schematic depiction of a Haar Cascade.

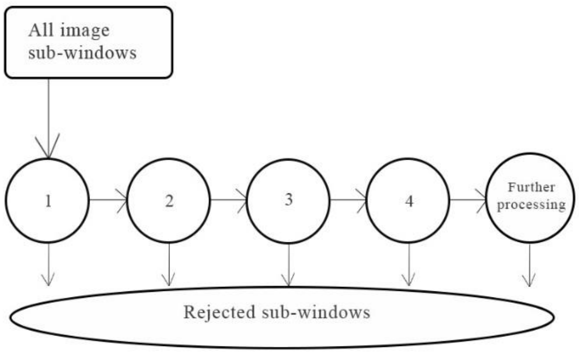


Figure 2: Haar Cascade

After inputting an image for face detection, a detection window with a basic resolution of 24x24 pixels start scanning the image from the top left corner. An image sub-window with a basic resolution of 24x24 pixels is formed by cropping the images with the same size as detection window. After the image sub-window scanned by the detection window in the first classifier, if the image sub-window consists of the feature in the first classifier, then this sub-window will proceed to the second classifier. Next, if the sub-window consists of all 10 features in the second classifier, then it will be proceeded to the third classifier, until the sub-window is rejected or passed through all the classifiers, then repeat for the next image sub-window and different scaled of detection window.

If a sub-window passed through all the classifiers, then the detector considers that the sub-window contains a human face. Otherwise, when the sub-window is rejected in any classifier, it will not proceed further. From the sample by [2], around 60% of non-faces rejected and 100% of faces detected in the first classifier of face detector. In this case, this structure of detector cascade can save a lot of work and time in a face detection process.

A face detector was created with Haar Cascades method using MATLAB. After that, the relationship between the scale factor with accuracy and computational time were studied by 120 frontal upright face images with the size of 384×288 pixels.

2. Haar Cascades Method

2.1 Luminosity method

Haar Cascade method only works with grayscale images. So, for any image inputted into the detector, the system will recognize whether it is either grayscale or color (RGB) image by referring to the channel number of the image. An RGB image is composed of 3 channels to represent red, green, and blue components in RGB space, while grayscale image has only one.

MATLAB reads images and returns the image representation output as $M \times N \times A$ array, where $M \times N$ is the resolution of the image, and A is the number of channels. In this case, $A = 3$ represents the image is RGB, converting it into grayscale by luminosity method with $Eq. 1$ is necessary for further processing.

$$Y = 0.2162\ R + 0.7152\ G + 0.0722\ B \qquad Eq. 1$$

The R, G, B represent the pixels value in red, green, blue channels respectively. The different weights of red, green, blue color channels in $Eq. 2.1$ are due to eyes are more sensitive to green, then red, then blue [4]. That means the grayscale image produced by normal averaging of three-color channels is not perceptually equivalent to the brightness of the original color image.

2.2 Haar Cascade model

To build a face detector, an XML file that contains the Haar Cascade model is needed. In this paper, 'haarcascade_frontalface_alt.xml' had been used, which was trained by Rainer Lienhart. This XML file is used in OpenCV system, which provides a real-time computer vision library and is widely used nowadays. There are several differences between this RL Cascade model with Viola Jones model. Its basic detection window size is 20×20 pixels, and it consists of 22 layers of cascaded classifiers, with different Haar feature sets with Viola Jones.

2.3 Image scaling

Referring to [2], they scaled the detection window with a recommended scale factor of 1.25, to obtain a variety of different size faces. However, large cache of memory is needed to achieve fast memory access for integral image due to the large size of detection window. This is difficult to achieve because of hardware limitation. Hence, a scaling image technique has been used in this paper as it does not need large cache memory for fast memory access and it is easy to implement using hardware [5], while the detection window remains as 20×20 pixels. Figure 3 shows the detection window, maintains the same size all the time, traverses the scaled image for each scaled increment.

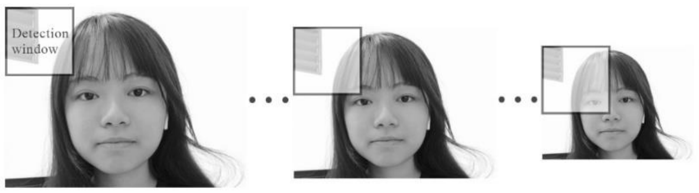


Figure 3: The detection window traverses the scaled image for each scaled increment

The disadvantage of scaling the image rather than scaling the detection window is it may cause the loss of features as the resolution of the image decreases as well as affects the accuracy of the detector [6]. The scale factor for face detection must be greater or equal to 1.0001, because the detection window's size must be increased in every increment.

To resize the input image, we need the number of pixels rows and columns, which are integers. But the result of dividing current input image size with scale factor might be a value with decimal numbers. Hence, the calculated new size of image should be rounded off to get the integer number as $Eq. 2$, where N is the current increment, an integer that is greater than zero.

$$\begin{matrix}\text{New scaled}\\ \text{image size}\end{matrix} = round\left(\frac{\begin{matrix}\text{Initial size of}\\ \text{input image}\end{matrix}}{\text{scale factor}^{N}}\right) \qquad Eq. 2$$

2.4 Integral image

During the scanning process, the feature value should be computed rapidly. The computational time is longer if the feature value is computed by adding all the pixels value. To speed up the process of face detection, this paper had used the integral image method to calculate the pixels value sum quickly.

The integral image is an intermediate representation for the input image. The value of integral image at location (x, y) consists of the pixels value sum that above and left of the location of x, y as

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \qquad Eq. 3$$

where $ii(x, y)$ is the integral image and $i(x, y)$ is the original image.

Figure 4 represents the value of the integral image at point (x, y) is the sum of all the pixels value above and left of (x, y).

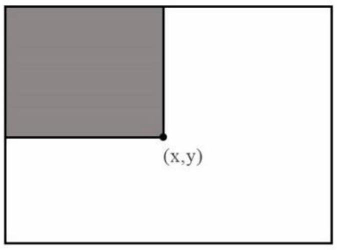


Figure 4: The value of integral image

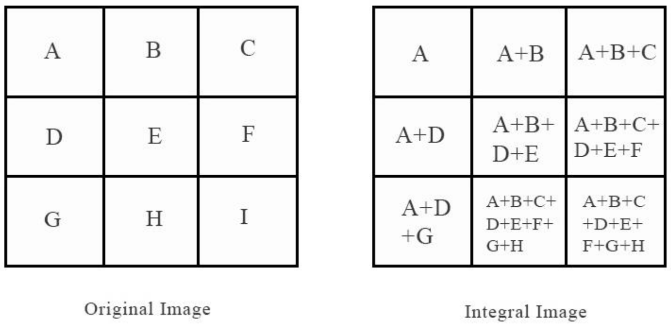


Figure 5: Example of calculation of integral image

Figure 5 shows an example of original image and its respective integral image. The integral image can be formed by these formulas:

$$s(x, y) = s(x, y-1) + i(x, y) \qquad Eq. 4$$
$$ii(x, y) = ii(x-1, y) + s(x, y) \qquad Eq. 5$$

Where $s(x, y)$ is the cumulative row sum, and $s(x, -1) = ii(-1, y) = 0$.

Integral image does not require the whole pixels value in respective feature for calculation of total pixels value within the region. To calculate the total number of pixels within region D in Figure 6, four values only are needed. The value of integral image at location 1 is the sum of pixels value within region A. The value of integral image at location 2 is the sum of pixels value within region $A + B$, $A + C$ for integral image at location 3, and $A + B + C + D$ for integral image at location 4. Then, the total pixels within region D can be calculated immediately by $4 + 1 - (2 + 3)$.

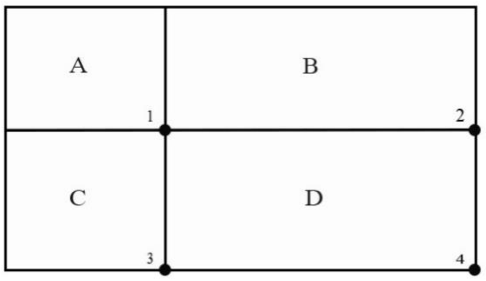


Figure 6: The integral image calculation

In this case, only six values are required during the subtraction process of pixels number under white region from pixels number under the black region for each two-rectangle feature, eight values needed for three-rectangle feature, nine values needed for four-rectangle feature.

2.5 Normalization

Different illumination variations of the images will affect the pixels value of the images. Hence, the possible range of feature value is very wide. From [7], a 12×18 pixels of Haar feature can produce a feature value of -2000 to $+2000$. The huge range of feature values is not conducive to the quantitative evaluation of feature values. To undergo this problem, after a feature value had been calculated, it should be normalized. The normalized feature value is always in the range of [0,1].

Firstly, the sum of pixel value and the sum squares of the pixel value of the image sub-window should be calculated.

$$sum = \sum i(x,y) \qquad Eq.6$$

$$sqsum = \sum i^2(x,y) \qquad Eq.7$$

Next, calculate the average value. The w and h represent the number of pixel rows and columns of the image sub-window respectively. In this paper, the image sub-windows are maintained as 20×20 pixels. Hence, both w and h are equal to 20.

$$mean = \frac{sum}{w * h} \qquad Eq.8$$

$$sqmean = \frac{sqsum}{w * h} \qquad Eq.9$$

Then, use the average values calculated to find the normalization factor, $varNormFactor$.

$$varNormFactor = \sqrt{sqmean - mean^2} \qquad Eq.10$$

Finally, normalized feature value, $normValue$ can be calculated.

$$normValue = \frac{\text{feature Value}}{varNormFactor} \qquad Eq.11$$

The normalized feature value should be used to compare with the threshold for further processing.

2.6 Integration of multiple detections

Recall that the detection window will move to different positions of the image to search the human face. At the same time, the input image will be resized to find the human face of different sizes. Since there is no obvious difference between an image sub-window with its neighbor sub-windows, as well as the face detector is less sensitive to small changes in translation and scale, multiple overlapping detection results windows appear near to each human face at most of the time.

From Figure 7, notice that there are multiple outputs around the human faces, and some scattered false positive detections. So, merging the overlapping detection is necessary to obtain only one output for each face and eliminate scattered false positive detections.

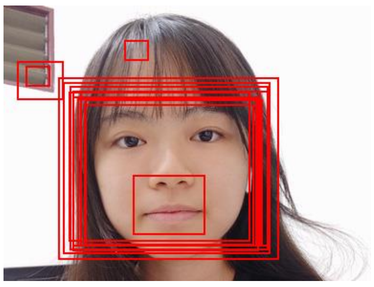


Figure 7: Output before merging the overlapping detection

In this paper, we consider 4 or more detections that are overlapped will declare a final output. In other words, the detection threshold, or merge threshold is equal to 4. Figure 8 shows different merge thresholds will produce different final output detections. Increasing the merge threshold may advance in suppressing false detections by requiring the human face to be detected more times during the multiscale detection phase.

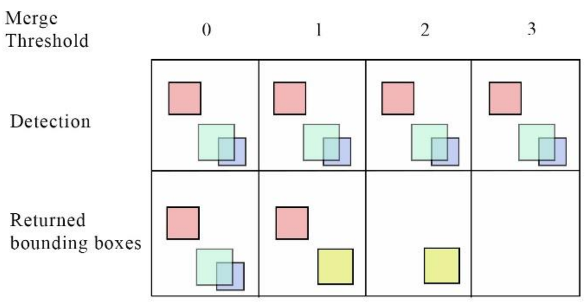
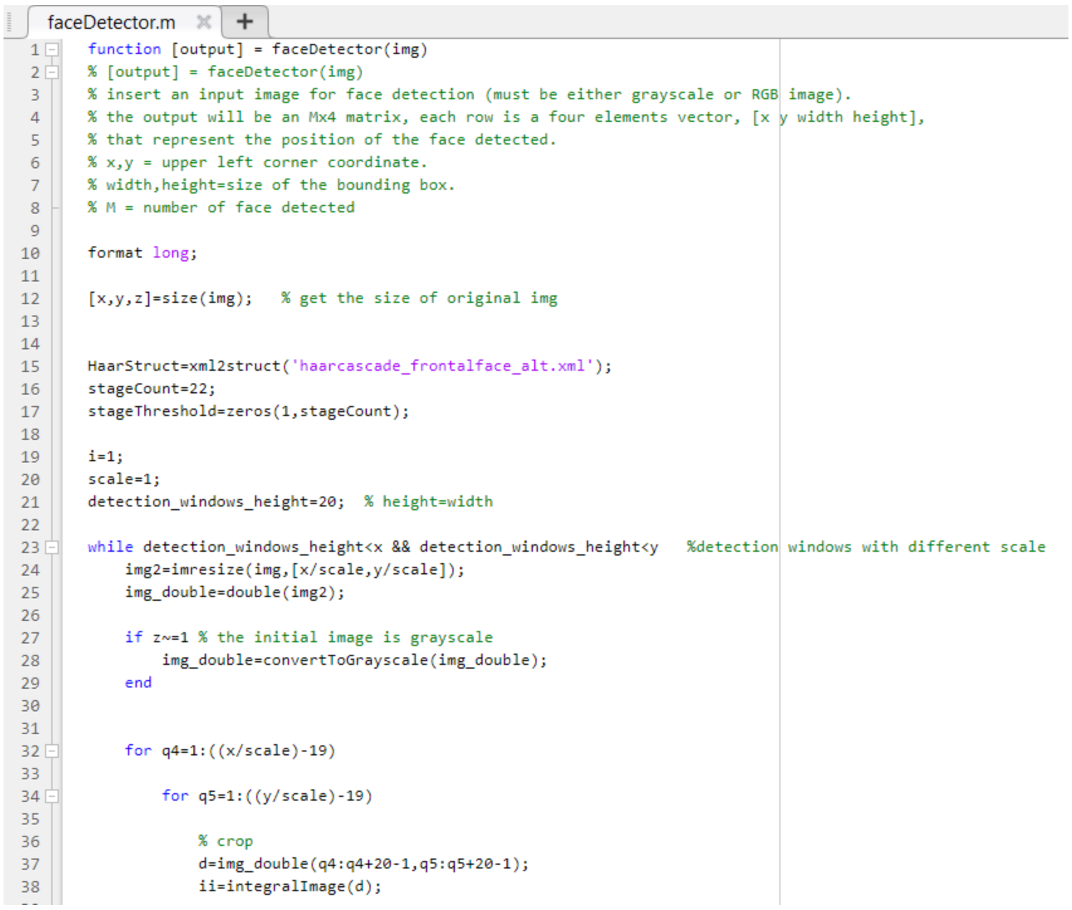


Figure 8: Example of different merge thresholds with their corresponding output detections

Firstly, the set of detections is partitioned into disjoint subsets. The detections that have overlapping bounding regions will be in the same subset. Then, the final detection bounding region equals the average of all detection corners in their corresponding subset.

3. Results and Discussion

A face detector system had created by MATLAB. Figure 9 shows the beginning part of the face detector code. A trained cascade model in XML format, 'haarcascade_frontalface_alt.xml', which created by Rainer Lienhart, was used in this paper.



```
function [output] = faceDetector(img)
% [output] = faceDetector(img)
% insert an input image for face detection (must be either grayscale or RGB image).
% the output will be an Mx4 matrix, each row is a four elements vector, [x y width height],
% that represent the position of the face detected.
% x,y = upper left corner coordinate.
% width,height=size of the bounding box.
% M = number of face detected

format long;

[x,y,z]=size(img);   % get the size of original img


HaarStruct=xml2struct('haarcascade_frontalface_alt.xml');
stageCount=22;
stageThreshold=zeros(1,stageCount);

i=1;
scale=1;
detection_windows_height=20;  % height=width

while detection_windows_height<x && detection_windows_height<y   %detection windows with different scale
    img2=imresize(img,[x/scale,y/scale]);
    img_double=double(img2);

    if z~=1 % the initial image is grayscale
        img_double=convertToGrayscale(img_double);
    end


    for q4=1:((x/scale)-19)

        for q5=1:((y/scale)-19)

            % crop
            d=img_double(q4:q4+20-1,q5:q5+20-1);
            ii=integralImage(d);
```

Figure 9: Beginning part of the face detector code

The output of the detector is a $M \times 4$ matrix. M represents the number of faces detected. Besides that, each row is a four-element vector: $[x, y, width, height]$, where x, y represent the upper left corner coordinate, and $width, height$ represent the size of face detected. To annotate the face detected, a rectangle is added with the same size as the face detected, at the location where the face founded as Figure 10.

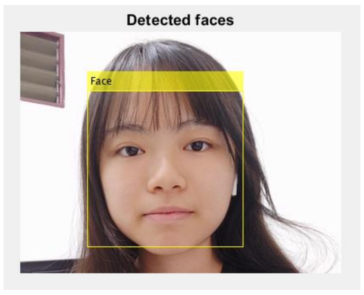


Figure 10: Output of the face detector on a test image

Next, the scale factor of the detection window had adjusted to different values to understand how the scale factor influences the detector's accuracy and computational time. Table 1 shows the results obtained. False detection is the number of false positive detection. Detection rates are obtained by

$$\frac{\text{number of true positive detection}}{\text{number of human faces in sample}} \qquad Eq. 12$$

Computational time are computed automatically by MATLAB.

Table 1: Detection rates, false positive detection, and computational time for various scale factors of 120 human face images

Scale factor	1.3	1.25	1.1	1.08	1.05	1.03	1.01
False detection	0	1	7	16	23	48	132
Detection rates	45.5%	54.5%	70.2%	72.7%	81.0%	87.6%	97.5%
Computational time (for each image, sec)	0.0190	0.0198	0.0342	0.0394	0.0564	0.0869	0.2801

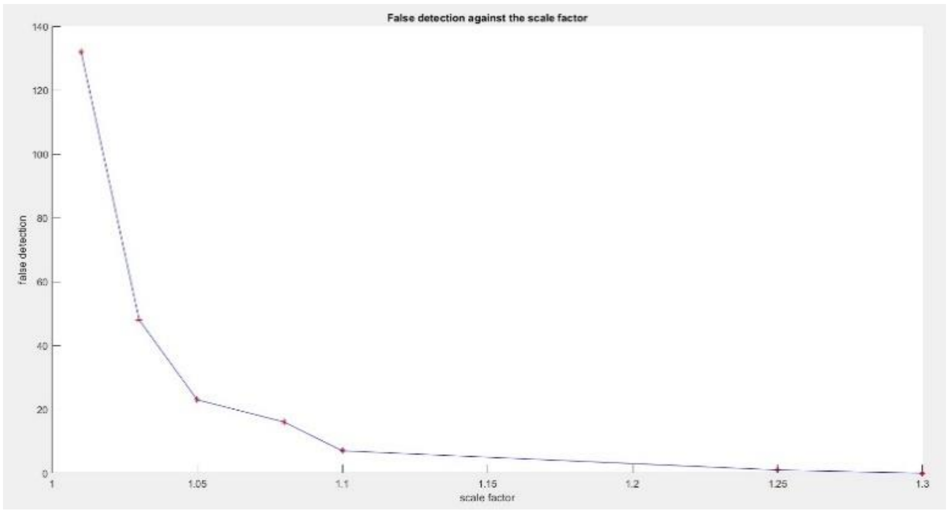


Figure 11: Graph of false detection versus scale factor

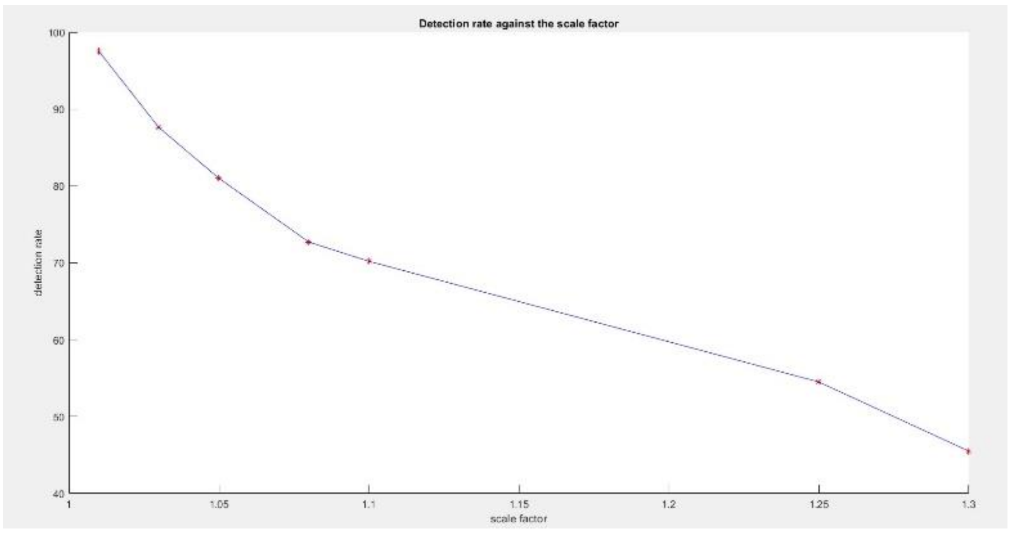


Figure 11: Graph of detection rate versus scale factor

From Figure 10 and Figure 11, notice that the false detection and detection rate decreases as the scale factor increase. This is due to fewer overlapping detections when the scale factor increases, then the probability of missing face detection is higher, as well as the false detection. Furthermore, the scale factor does not affect much on false detection count when it comes to 1.1 and above.

From Figure 12, the detector computational time is inversely proportional to the scale factor. In overall, increasing in scale factor will reduce the computational time, due to the fewer image sub-windows to be scanned by the detector. Additionally, the rate of change in computational time drops significantly only when the scale factor is smaller than 1.1.

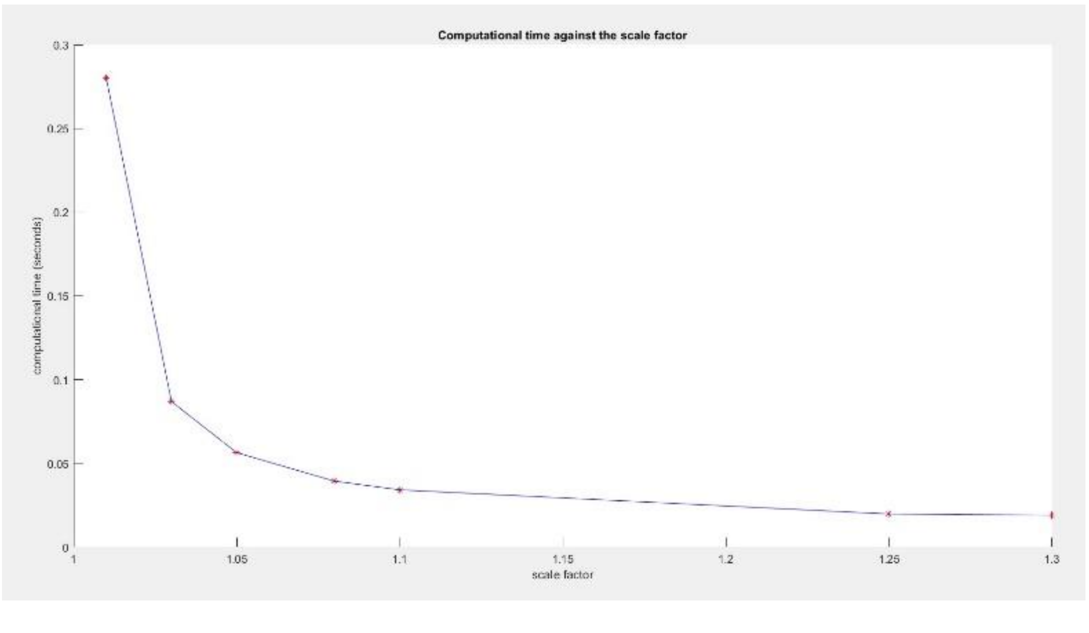


Figure 12: Graph of computational time versus scale factor

A good face detector must have a low false detection, high detection rate, and short computational time. From the results above, the false detection and computational time are decreases as the scale factor increase. But it does not affect them significantly when it is up to 1.1. Hence, the suggested scale factor is larger than 1.1 to obtain low false detection and short computational time. On the other hand, to achieve a high detection rate, the scale factor should be small. In this case, the suggested scale factor in this paper is 1.1.

4. Conclusion

From the results, the scale factor increases will cause the false detection, detection rate and computational time decrease. Additionally, to obtain low false detection, high detection rate, and short computational time, the suggested scale factor is 1.1.

However, this detector can only detect the frontal up-right human face. Hence, further work can be carried out to apply some techniques like the rotated Haar-like feature introduced by [8], or the extend~~ed~~ rotated Haar-like feature from [9].

Acknowledgement

The authors would also like to thank the Faculty of Applied Sciences and Technology, Universiti Tun Hussein Onn Malaysia for its support.

References

[1] Berbar, M. A., Kelash, H. M. & Kandeel, A. A. (2006). Faces and Facial Features Detection in Color Images. In Geometric Modeling and Imaging—New Trends (GMAI'06) (pp. 209-214). IEEE.

[2] Viola, P. & Jones, M. (2001). Robust Real-time Object Detection. International journal of computer vision, 4(34-47), 4.

[3] Zafeiriou, S., Zhang, C. & Zhang, Z. (2015). A Survey on Face Detection in the Wild: Past, Present and Future. Computer Vision and Image Understanding, 138, 1- 24.

[4] Mendoza, F. & Lu, R. (2015). Basics of Image Analysis. In Hyperspectral imaging technology in food and agriculture (pp. 9-56). Springer, New York, NY

[5] Cho, J., Mirzaei, S., Oberg, J. & Kastner, R. (2009). FPGA-based face detection system using haar classifiers. In Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays (pp. 103-112).

[6] Sharma, H., Saurav, S., Singh, S., Saini, A. K. & Saini, R. (2015). Analyzing impact of image scaling algorithms on viola-jones face detection framework. In 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 1715-1718). IEEE.

[7] Sun, S. (2016). Renlian jiance OpenCV zhongde Haar+Adaboost jilian fenleiqi fenjie yi: Haar tezheng jieshao [Face detection Haar+Adaboost cascade classifier decomposition in OpenCV (1): Introduction to Haar features]. Retrieved from https://blog.csdn.net/SMF0504/article/details/52795098

[8] Lienhart, R. & Maydt, J. (2002). An Extended Set of Haar-like Features for Rapid Object Detection. In Proceedings. international conference on image processing (Vol. 1, pp. I-I). IEEE.

[9] Messom, C. & Barczak, A. (2006). Fast and Efficient Rotated Haar-like Features Using Rotated Integral Images. In Australian Conference on Robotics and Automation (pp. 1-6).