International Journal
Information Technology Applications (ITA)

Volume 7, Number 2, December 2018

AN EFFECTIVE FACE DETECTION ALGORITHM FOR CLIENT-SIDE WEB-BASED SOLUTIONS

Zuzana Képešiová, Štefan Kozák

Abstract:

This paper studies a real-time face detection problem for online web applications and its solution. The main aim of this paper is to describe functionality of effective face detection algorithm and analyze existing solution in order to improve current situation. To obtain satisfying results Viola-Jones object detection framework is chosen, which uses feature selection in combination with Integral images, AdaBoost training and Cascading classifiers. This algorithm is highly popular among existing javaScript frameworks for face or another objects detection for its extreme speed of image processing while achieving high detection rates and multiple objects detection. As analyzed answer for solving stated problem a javaScript tracking.js is chosen.

Keywords:

Computer vision, face detection, Viola-Jones framework, AdaBoost training, cascading classifiers, tracking.js.

ACM Computing Classification System:

Computer vision, machine learning algorithms, web applications.

Introduction

In this era, a computer vision, a field that covers high level understanding of digital images and videos, that seeks automatization of human visual system tasks from the engineering perspective [1], is on its rise. Since more and more refinements and simplifications of every day human lives are relying on computer technology, computer vision has its place in this field too. Face detection or, also more generally said, object detection is one of fundamental blocks for computer vision.

Face detection is a computer-based technique to distinguish human faces in digital images, still like photographs or moving as a video, in order to extract or caption the human face. This technology is used in variety of applications, i.e. biometrics. Face detection, object detection and gesture detection itself is getting into a larger awareness nowadays, since its wide range of use. Not only biometrics, but also interactive games, intelligent gadgets or part of face recognition for security are in the center of interest.

In this paper we will focus on face detection mainly. A face detection is considered as a specific part of object-class detection. In object-class detection the task is to find the locations and sizes of all certain class related objects. Eye regions, eyebrows, iris, nostrils and mouth corners combined in explicit mutual proportions are creating a human face.

These algorithms are focused mainly on frontal human faces, but also a small rotation can be considered. In this case, each pixel of a photography or a video record is compared to a processed database of human faces.

Any facial changes in the database, such as tone of skin, light in the room, different proportions of the face, smile, grin and many more are considered and processed before and during training algorithm run. There are different approaches of normalizing images such as each face candidate is normalized to reduce both the lightning effect and head rotation or skipping normalizing and continue with defining with preprocessed similar features.

1 Problem formulation

The main problem to be solved in this paper is finding and describing an effective algorithm to find face on a visual input, image or video, that is quick and lightweight enough that it can be used in online web applications. What looks as an easy task for a human, is not an easy task for a computer since computer has to be programmed to take exact orders and define mathematically how a face looks like. Every person differs and also there are many factors, that change the exact model, i.e. face proportions, deformations, expressions, captured angle etc. At this point, however, it is reachable to apply an algorithm that is bounded by certain rules.

Fig. 1. A typical input image [2].

In Fig. 1., we can see a typical input image to a face detection algorithm. This image consists of group of people of different races, sexes, angles of faces (frontal to approximately 20 degrees) and contrast composition of face itself and its surroundings like hair and hair color and background.

Another challenge is to develop a lightweight algorithm, that would be enough robust to provide very high detection rate, also known as true-positive rate, and very low false-positive rates and provide a real-time detection for online solutions not to halt CPU of a basic user.

We seek an algorithm, that need not be used only on faces itself, but also on their segments, as eyes or mouth for closer lookups, what basically means, that algorithm can be trained not only to detect faces, but objects themselves.

2 Viola-Jones method

Viola-Jones method is a framework firstly proposed by Paul Viola and Michael J. Jones in 2001. This framework has its vision to provide a solution to process images extremely rapidly while achieving high detection rates. The main idea behind this is image representation called Integral Image which provides detector a high computation. [3]

Converting image to an integral image is done by expressing each pixel by a sum of all pixels above and to the left of the concerned pixel. Such a conversion is presented in Fig.2. [4]

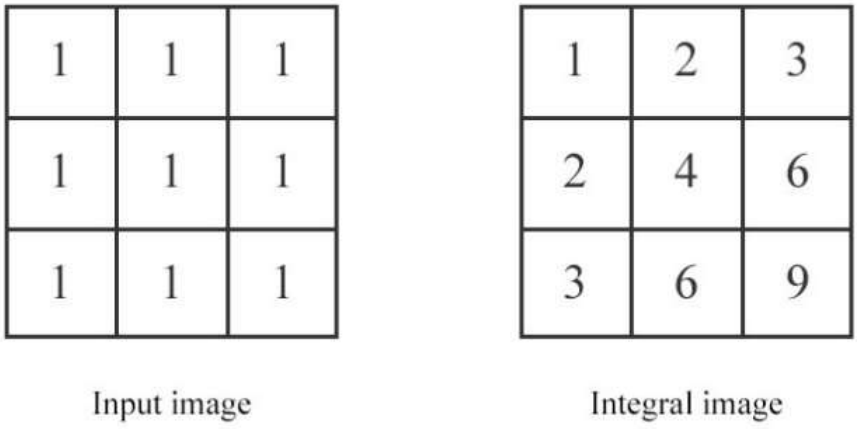

Fig. 2. Image conversion to summed-area table, also known as Integral image.

As the second step, when image is represented as summed-are table, AdaBoost learning algorithm is used to build an efficient classifier to select a small amount of visual features that are crucial for detecting an object using feature selection.

The third and final step is combination of classifiers into a cascade. This technique promises fast discard of background of the image and focus more on face promising features, [3].

A Features

In the above discussed face detection process, images are classified based on the value of simple features, which are encouraged more than using individual pixels directly. Such a technique is used mainly because it is faster way than a pixel-oriented system.

Originally, there were three kinds of features used to select, two-rectangle feature, three-rectangle and finally a four-rectangle feature.

Two-rectangle feature consists of two regions and it is a difference between the sum of the pixels within the two of them. When it comes to three-rectangle feature, than it computes the sum within two outside rectangles subtracted from the sum of central rectangle. Four-rectangle feature computes the difference between diagonal rectangle pairs. All those three kinds of HAAR like features are visible in Fig.3. These types are base types, but can be also vertically or horizontally rotated, what makes more feature types.

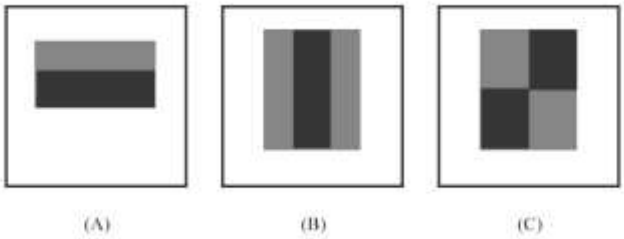

Fig. 3. Three types of rectangle features. The sum of pixels from the grey area is subtracted from the sum of pixels in the black area. Two-rectangle features are visible in (A), three-rectangle features in (B) and four-rectangle features are shown in (C).

The base resolution of a detector is 24 x 24, what offers total of approximately 160 000 different features to be constructed, what basically outnumbers total 576 pixels contained in the detector. [3]

Each feature is computed by subtracting the sum of light rectangles from the sum of dark rectangles. These features are simple in their true nature for an advanced object detection, but on the other hand they provide sufficient computational speed, that is wanted and a key feature for real-time online face detection. [3]

Later on, another type of rectangle features was introduced, since based on experiments, only three types were not efficient enough to detect non-upright faces and non-frontal faces with high accuracy. This was a time, when diagonal features were presented. These diagonal features are represented as four overlapping rectangles, that create a new shape. They are calculated in the same way as previous features, [5].

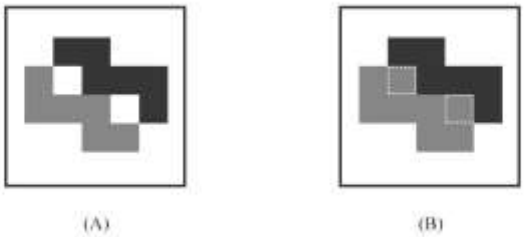

Fig. 4. Example of diagonal features related to the enclosed detection window. Basic type of diagonal feature is shown in (A) and filter constructed from 4 rectangles is visible on (B).

1.) Integral image

Rectangle features used for selection can be calculated quite quickly by using an interpretation of integral image, or also called summed-area table. The integral image at position x, y contains the sum of all pixels above and to the left of the given location:

$$II = \sum_{x' \leq x, y' \leq y} I(x', y') \quad (1)$$

where II(x, y) is the integral image and I(x, y) is a representation of the original image. The integral image can be computed in one pass over the original image by using the following pair of recurrences:

$$s(x, y) = s(x, y - 1) + I(x, y) \tag{2}$$

$$I(x, y) = II(x - 1, y) + s(x, y) \tag{3}$$

where *s(x, y)* is a cumulative row sum, *s(x. -1) = 0* and *II(-1, y) = 0*.

Using the integral image any rectangular sum can be calculated in four array references like those showed in Fig.5.

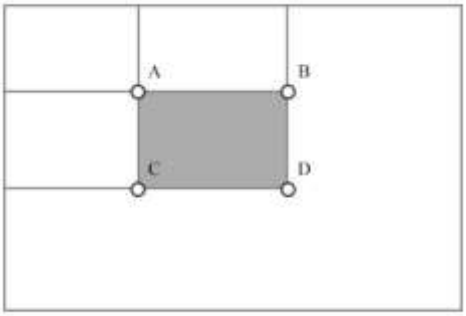

Fig. 5. The sum of Grey rectangle can be calculated as D – (B + C) + A

2.) Feature selection

Regarding to faces, every face consists of isolated structures as eyes, nose and lips which vary from person to person. Face detection algorithm is based on those features that measures face directly and robustly and is capable of distinguish those differences between personal variation.

Each feature mentioned before describes a concrete property at a certain position, scale and aspect ratio and therefore is assigned a weight. Filters are there for choosing which variation is still considered as a part of a face and which is not. This problem covers the learning algorithm to determine which features are effective enough to distinguish a face from other objects i.e. ratio of eyes, nose and mouth should be considered as an effective feature, while on the other hand hair itself shouldn't since its variability in haircuts.

B Learning algorithm

In order to effectively find features, that differentiate characteristics of a face from other pieces of images, the challenge was to find an efficient learning algorithm. As mentioned before, there are approximately 160 000 rectangle features associated with each sub-window, a number, that is much larger than the number of pixels. While there is a great number of possible features, only a small number of them combined can compose a beneficial classifier. Therefore Viola-Jones framework is based on modified AdaBoost learning algorithm developed by Freud and Schapire in 1996 [6]. It's used to find the right features and also to construct a classifier.

In original form, AdaBoost is a learning algorithm for boosting the classification performance of a simple learning algorithm by creating a strong classifier by combining of a collection of weak, simple, classifying functions. A weak learn function is called weak for its simplicity and mathematically described as follows:

$$h(x, f, p, \theta) = \begin{cases} 1 & \text{if } pf(x) > p\theta \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where x is a 24x24 pixel sub-window, *f* is applied feature, *p* is the polarity and Θ is a threshold responsible for classifying x as positive or negative.

The best classification function to classify the training data is not expected in this case. Rather than that, a weak learner is boosted by solving a sequence of learning problems. Round after round of learning, the examples are re-weighted in order to improve ones, that were classified by previous weak classifier incorrectly. The final form of a strong classifier is a perceptron, a weighted combination of simple classifiers in combination with threshold. [3]

It is proved by Freund and Schapire, that the training error of a strong classifier goes to zero exponentially with the number of rounds. Since AdaBoost is round based, the algorithm assigns to each example x_i a weight w_i which is initialized to 1⁄N where N is a number of all samples. After every round, one weak classifier is chosen, while the requirement for selecting one is, that its error rate is less than 0.5. In this case, input vectors are image pairs:

$$x_i = (I_1^i, I_2^i) \tag{5}$$

and a weak classifier is composed of one feature with rectangle filter on each input image in the pair. Therefore let the algorithm is modified as follows: If

$$h_j(I_1, I_2) = \left|\phi_j\left(I_1^i\right) - \phi_j\left(I_2^i\right)\right| \tag{6}$$

where ϕ_j is a scalar function of an image called "filter" then:

$$f_j(x_i) = f_j\left(I_1^i, I_2^i\right) = \begin{cases} \alpha & \text{if } h_j(I_1^i, I_2^i) > t_j \\ \beta & \text{otherwise} \end{cases} \tag{7}$$

where α and β are weights and t_j is a threshold that combined with rectangle filters create classifier that separates the positive examples from the negative ones.

On every round of AdaBoost is chosen a weak classifier ϕ_j and t_j for which:

$$\epsilon_j = \sum_{\substack{i:\, y_i = +1 \\ h(x_i) \le t}} D_i + \sum_{\substack{i:\, y_i = -1 \\ h(x_i) > t}} D_i \tag{8}$$

is minimized. The first sum specifies sum of weights of the examples that are false negatives and the second one is the sum of weights for examples that are false positives. Therefore minimizing the summing of false negatives and false positives minimizes also the weighted error.

After successful finding of optimal threshold and filter by minimizing the weighted error, adequate values for α and β are calculated. Minimizing is considered as good criteria for selection of weak hypotheses, which can be used to calculate efficient values α and β as follows:

At first, sum should be split into a sum of negative and sum of positive examples:

$$Z = \sum_{i:\, y_i = +1} D_i e^{-f(x_i)} + \sum_{i:\, y_i = -1} D_i e^{f(x_i)} \tag{9}$$

$$Z = \sum_{\substack{i:\, y_i=+1 \\ h(x_i)>t}} D_i e^{-\alpha} + \sum_{\substack{i:\, y_i=+1 \\ h(x_i)\le t}} D_i e^{-\beta} + \sum_{\substack{i:\, y_i=-1 \\ h(x_i)>t}} D_i e^{\alpha} + \sum_{\substack{i:\, y_i=-1 \\ h(x_i)\le t}} D_i e^{\beta} \tag{10}$$

By simplifying the formula, we get:

$$Z = W_+^+ e^{-\alpha} + W_+^- e^{\beta} + W_-^+ e^{\alpha} + W_-^- e^{\beta} \tag{11}$$

where W_+^+ are true positives, W_+^- are false negatives, W_-^+ are false positives and W_-^- are true negatives.

Z respectively to α can be minimized by:

$$\frac{\partial Z}{\partial \alpha} = W_-^+ e^{\alpha} - W_+^+ e^{-\alpha} = 0 \Rightarrow \alpha = \frac{1}{2} \log\left(\frac{W_+^+}{W_-^+}\right) \tag{12}$$

and also similarly for β:

$$\frac{\partial Z}{\partial \beta} = W_-^- e^{\beta} - W_+^- e^{-\beta} = 0 \Rightarrow \beta = \frac{1}{2} \log\left(\frac{W_+^-}{W_-^-}\right) \tag{13}$$

These equations are used to set α and β every round of AdaBoost.

- Given example images $(x_1, y_1) \ldots (x_n, y_n)$ where y_i = 0 for negative and 1 for positive examples.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for y_i = 0, 1 respectively, where *m* and *l* are the number of negatives and positives respectively.
- For *t* = 1,..., *T* :
  - Normalize the weights, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$
  - Select the best weak classifier with respect to the weighted error

$$\epsilon_t = \min_{f,p,\theta} \sum_i w_i \left| h(x_i, f, p, \theta) - y_i \right|$$

  - Define $h_t(x) = h(x, f_t, p_t, \theta_t)$ where f_t, p_t, and θ_t are the minimizers of ϵ_t.
  - Update the weights:

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

where e_i = 0 if example x_i is classified correctly,
e_i = 1 otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$

- The final strong classifier is:

$$C(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \ge \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = \log \frac{1}{\beta_t}$

Fig. 6. Pseudocode of the modified AdaBoost algorithm [3].

C Cascading classifier

The key principle of Viola-Jones object detection framework is to go through the same image many times, while the spotted window is enlarging. In most of the cases of input images there will be more of sub-windows that don't hold any face related information. Therefore, a different approach to solve this problem is considered. Instead of finding face related sub-windows, algorithm should be focused more on discarding non-face sub-windows. This should be a better conclusion, since it's faster to discard a face non-related sub-windows than evaluating sub-windows containing a face related information. In this case a single strong classifier wouldn't be much efficient, because no matter the input, the computational time remains constant. This problem offers a question how to boost the computational time and its solution is presented as cascade classifier.

The final form of the classifier after AdaBoost training is a perceptron, a thresholded linear combination of features. Viola-Jones framework uses a sequence of more and more complex classifiers formed into the cascade in order to reduce false positive rate and also to significantly improve computational efficiency. Simple classifiers are called to reject most of sub-windows, that holds background and any other unimportant sections of an image, and after that increasingly stronger classifiers are used to achieve low false positive rates. The input window is passed from classifier to the next stronger classifier in the cascade until every one of them returns true. Otherwise an input window is considered as non-face and is discarded.

The single stage classifiers allow higher false negatives rates in order to reduce false positive rate, but in this case it's not much of a problem to pass more false positives, since sub-windows are reassessed by stronger classifiers. Because most parts of an image don't look like a face itself, they are very quickly thrown away. Such a process is depicted in Fig.7.

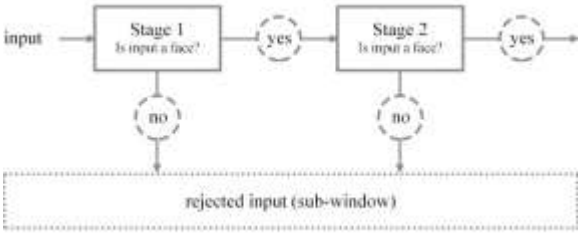

Fig. 7. Cascaded classifier functioning showcase

Conclusion

Considered face detection algorithm is popular for real-time online face detection using video-input from a camera or also for detection from a still image thanks to its low computational load.

One of solution like that is a javaScript project called tracking.js that is capable of recognizing faces, mouths and eyes on the input. The main idea of this solution is to take precomputed patterns for all mentioned objects to be detected and compare it with the input. It can be modified to detect also other objects by providing a representation as a Float64Array variable filled with HAAR like features and a bit of customization of the detector class.

A medium image, approximately maximum of a size 800x800 pixels has to be provided, depending on the computational resources of the user's computer, in order to not halt the CPU and not enforce the site to crash. Therefore, this script is a useful presentation of a real-time online face detection, but needs further improvement for scaling the input image to smaller size in case of high-quality image input.

For video input a different approach should be considered, since for now the estimated face area differs radically by every small change of position of the scanned object. Examples of a face detection results used by tracking.js solution are showed in Fig.8, Fig.9 and Fig.10.

Fig. 8. successful processed and evaluated image with face detected and highlighted [2].

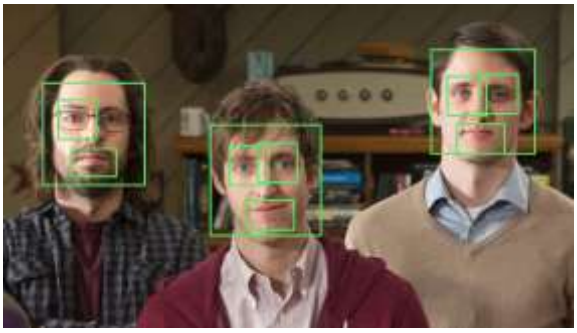

Fig. 9. face, mouth and eyes detection using tracking.js [8]

Results for Fig.8 and Fig.9 are satisfying. Every face on the photo was detected successfully, while on Fig.9 the right eye of the first face wasn't recognized, most probably glasses created noise, while shadow as a noise source should be discarded, because other two faces are similarly contrasted.

After changing the input image to another frontal face results are not that satisfying as before. Image itself has low contrast, but as smaller input as better results are provided. Another problem that this framework generates is multiple detection of objects in a very close area. In the end, there are three types of problems created by tracking.js: buggy object detection in video input, inadequate input image size and multiple detection in a close area.

Problems found it this framework could be get rid of by calibrating code. The essence of buggy face recognition in video is in false negative frames while the frame changed only slightly. The main idea is compute the total weight of difference and evaluate its value.

When it comes to the size of the photo, the main idea is changing the size to acceptable size around 500x500 pixels to not halt CPU by computation right after its upload. By changing the size it doesn't only improve the calculation speed, but also it improves the true positive rate.

Solution of a problem of multiple object detection in very close overlapping objects borders could be storing the size and location of those areas and their comparison.

The main aim in this case is to calibrate the ratio of found features by putting a weight for every found feature in their positions, i.e. overlapping regions of the eyes and mouth in comparison to the face and to the other recognized features like eyes should be placed on the upper face while mouth on the bottom face. Therefore, another set of rules should be considered.

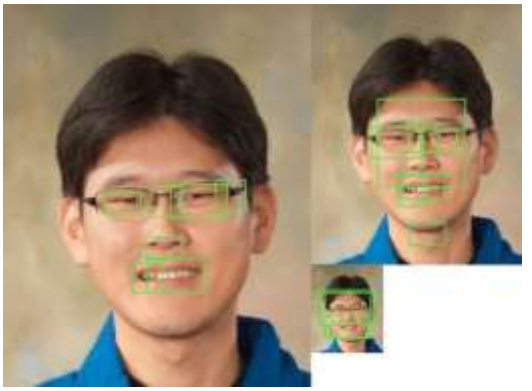

Fig. 10. A difference of recognized face and its features varies significantly by the size of an input image.

Acknowledgement

The work has been supported by Grant No 030STU-4/2017 of the Cultural and Educational Grant Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic.

References

[1] Sonka, M., Hlavac, V., Boyle, V., (2008). *Image Processing, Analysis, and Machine Vision.* Thomson. ISBN 0-495-08252-X.

[2] NASA (2009). NASA Astronaut Group 20. Retrieved December 7, 2017, from https://spaceflight.nasa.gov/gallery/images/behindthescenes/training/hires/jsc2009e215201.jpg

[3] Viola, P., Jones, M.J. (2004). Robust Real-Time Face Detection. *International Journal of Computer Vision.* vol. 57, no. 2, p. 137–154.

[4] Viola, P., Jones, M.J. (2003). Face Recognition Using Boosted Local Features. *Mitsubishi Electric Research Laboratories, Inc.*

[5] Viola, P., Jones, M.J. (2003) Fast Multi-view Face Detection. *Mitsubishi Electric Research Laboratories, Inc*.

[6] Crow, F.C. (1984). Summed-Area Tables for Texture Mapping. *Computer Graphics*. vol. 18, no. 3.

[7] Bishop, C.M. (2006). *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC. ISBN-10: 0-387-31073-8

[8] Tracking.js. Retrieved December 12, 2017, from https://trackingjs.com/

Authors

Ing. Zuzana Képešiová
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava, Slovakia
zuzana.kepesiova@stuba.sk

Currently a student of doctoral studies at Slovak University of Technology in Bratislava. The main focus of her studies is oriented to development of a remote laboratory to monitor and control of mechatronic systems using digital twin and mixed reality.

Prof. Ing. Štefan Kozák, Phd.
Faculty of informatics, Pan-European University, Bratislava, Slovakia
stefan.kozak@paneurouni.eu

Currently at the Institute of Applied Informatics at the Faculty of Informatics, Pan-European University in Bratislava. His research interests include system theory, linear and nonlinear control methods, numerical methods and software for modeling, control, signal processing, IoT, IIoT and embedded intelligent systems for digital factory in automotive industry.