Preface

This report introduces and describes the Viola Jones Framework. It is a framework for robust and rapid object detection using a boosted cascade of simple featured. It was firstly introduced by Paul Viola and Michael Jones in 2001. It is a machine learning approach which allows to detect faces in images extremely rapidly and to achieve high detection rates. The images used for the detection have to be in greyscale. The framework is not suitable for coloured images. It consists of three contributions.

The first contribution is a new image representation called "Integral Image" which allows very fast feature evaluation. The second contribution is a method which selects a small number of critical visual features to build a strong classifier. The third contribution introduces a procedure which combines increasingly more complex classifiers in a cascade structure.

Finally the report gives information about experiments and results based on the Viola Jones Framework.

December 2014

Christos Stoilas
Foundations of Computer Vision
Technische Universität München

Table of Contents

Integral Image

Paul Viola[1] and Michael Jones[2]

[1] Mitsubishi Electric Research Labs, 201 Broadway ,8th FL
Cambridge, MA 02139
viola@merl.com
[2] Compaq CRL
One Cambridge Center, Cambridge, MA 02142
mjones@crl.dec.com

Abstract. This contribution introduces and explains in detail the purpose and the use of features and the integral image.

Keywords: feature evaluation, rectangle-features, Summed-Area-Table

1 Rectangle-features

Images are classified based on values of simple features. Generally, features are specific structures in images like points, edges or objects. In this framework, features represent diffenrences in intensities between regions of an image. They are called "rectangle-features" and they are categorized in three types: two-rectangle features, three-rectangle features and four-rectangle features. This object detection procedure is based on features and not on pixels for two reasons. The first reason is that features are more robust than pixels, as they can act to encode the ad-hoc domain knowledge. The second reason is based on the fact that a feature-based system operates faster than a pixel-based system.

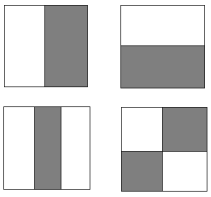

Fig. 1. The three kinds of rectangle-features. From up left to bottom right: Two rectangle feature, two-rectangle feature, three-rectangle feature, four-rectangle feature.

These rectangle features have to be evaluated. In order to calculate the value of each feature, the sum of pixels of the white rectangles of the features are subtracted from the sum of pixels from the grey rectangles. For instance, a three-rectangle feature computes the sum within two outside rectangles subtracted from the sum in a center rectangle.

Features can appear in different locations and in different scales on the detection window as in figure 2 shown. Considering this fact, the number of features in a 24x24 pixel detection window is over 180000.

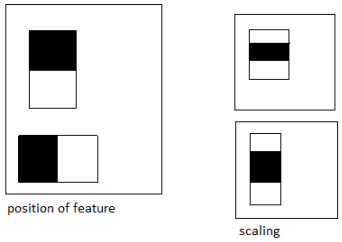

Fig. 2. Rectangle features shown relative to the enclosing detection window.

Rectangle features provide a rich image representation which supports effective learning.

2 Integral Images

2.1 Introduction

Features have to be evaluated as mentioned in the previous section. But summing all pixels one by one for that large number of features would cost very much computation time. Viola and Jones came up with a solution, using integral images for the evaluation. The integral image is a new image representation which allows quick calculation of pixel sums within rectangular cutouts. It is a very efficient way to evaluate rectangle-features. The integral image can be computed from an original image with a few operations per pixel. Once computed, every feature can be computed in constant time.

2.2 Computing the Integral image

When creating an integral image, it is necessary to create a Summed Area Table. In this table going to any point, there is a certain value for this table entry. This value is the sum of all the pixel values above, to the left and of course including the original pixel value of this point.

The value in the integral image at (x, y) is calculated:

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

The value $s(x, y)$ is given by the sum of these four terms. The original pixel value $i(x, y)$ from the original image is added with the values directly above this

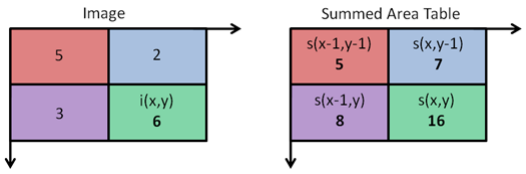

Fig. 3. Calculating the pixel values of the integral image by given values of the original image.
Taken from: "https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/", Visited on 19.12.2014

pixel and directly left of this pixel from the integral image at $s(x-1, y)$ and $s(x, y-1)$ respectively. Then the value $s(x-1, y-1)$ directly top-left of $i(x, y)$ from the integral image is subtracted from this outcome.

So, this procedure works recursively, as to evaluate a certain region, it is necessary to have calculated values of previous regions. By adding all pixel values from the original image gives the value 16 which is equal to the value of $s(x, y)$, so the algorithm works correctly.

For calculating only some rectangle (for example rectangle D in figure 4) there can be used a formula which uses four array references:

$$i(x', y') = s(A) + s(D) - s(B) - s(C) \tag{2}$$

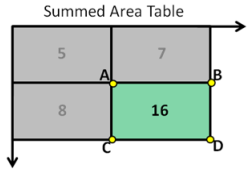

Fig. 4. Value of $s(D)$.
Taken from: "https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/", Visited on 19.12.2014

References

1. Viola, P., Jones, M.: Rapid Object Detection using a Boosted Cascade of Simple Features. Accepted Conference On Computer Vision and Pattern Recognition 2001
2. https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/ Visited on 19.12.2014

Boosting

Paul Viola[1] and Michael Jones[2]

[1] Mitsubishi Electric Research Labs, 201 Broadway ,8th FL
Cambridge, MA 02139
viola@merl.com
[2] Compaq CRL
One Cambridge Center, Cambridge, MA 02142
mjones@crl.dec.com

Abstract. This contribution introduces the purpose of the Adaboost algorithm. Furthermore, it explains in detail how this algorithm works.

Keywords: training examples, classifiers, threshold, weak learner

1 Introduction

Adaboost is an offline procedure and improves the accuracy of a learning algorithm. The idea of this procedure is to select and to combine a small number of weak classifiers to build a strong classifier in order to reduce computation time. The number of features which have to be evaluated is very large. Computing all these features would spend too much computation time. So the use of the Adaboost algorithm is to select some critical features for the face detection. A classifier is formed by one or more features and decides whether a region of an image has to be considered or not. A weak classifier has only one feature as an input and can classify correctly with a probability which is slightly better than 50%.

2 Adaboost algorithm

The database contains training exmples. Each example has properties (x_i, y_i) where x_i indicates the number of the example and $y_i = 0, 1$ for negative and positive examples respectively. The weights of the examples have also to get initialized: $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ where m, l are the numbers of the negative and positive examples respectively. The Adaboost algorithm performs a number of steps for a certain number of iterations. The number of iterations depends on the number of features the final strong classifier will have. Starting with the algorithm:

For $t = 1...T$:

- Normalizing the weights: $w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$, in order to make all weights dependent from each other.

- Foreach feature j, training a weak classifier using the weak learner. The weak learner is designed to select the single feature which best separates the positive and negative examples. The weak learner also determines the optimal classification function, such that the minimum number of examples get misclassified. The classifier h_j consists of a threshold θ_j and a parity p_j indicating the direction of the inequality sign:

$$h_j(x) = \begin{cases} 1, & p_j \cdot f_j < p_j \cdot \theta_j \\ 0, & otherwise \end{cases}$$

  The classifier can take two values, 1 or 0, depending on whether it recognized a face on the specific image or not. Evaluating the error of each constructed classifier is necessary: $\epsilon_j = \sum_i w_i \cdot \mid h_j(x_i) - y_i \mid$
- Choosing the classifier h_t with the lowest error.
- Updating the weights: $w_{t+1,i} = w_{t,i} \cdot \beta_t^{1-e_i}$,
  where $e_i = 0$ if example image x_i is classified correctly, otherwise $e_i = 1$ and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$, in order to emphasize the misclassified examples.

These steps are performed by Adaboost for all iterations. Afterwards Adaboost has selected T weak classifiers which can be combined to build the strong classifier:

$$h(x) = \begin{cases} 1, & \sum_{t=1}^{T} \alpha_t \cdot ht(x) \geq \frac{1}{2} \cdot \sum_{t=1}^{T} \alpha_t \\ 0, & otherwise \end{cases}$$

where $\alpha_t = log\frac{1}{\beta_t}$ represents a weight for each weak classifier.

The features which are selected by Adaboost are trained to detect faces, not other objects, as the database includes images which contain only faces.

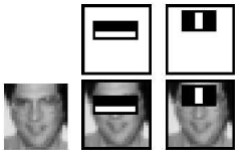

Fig. 1. Two features selected by Adaboost. The first one compares the intensity between the region of the eyes and the region across the upper cheeks. The second one compares the intensities in the eye regions to the intensity across the bridge of the nose.
Taken from: "Viola, P., Jones, M.: Rapid Object Detection using a Boosted Cascade of Simple Features. Accepted Conference On Computer Vision and Pattern Recognition 2001"

References

Viola, P., Jones, M.: Rapid Object Detection using a Boosted Cascade of Simple Features. Accepted Conference On Computer Vision and Pattern Recognition 2001

Detector Cascade

Paul Viola[1] and Michael Jones[2]

[1] Mitsubishi Electric Research Labs, 201 Broadway ,8th FL
Cambridge, MA 02139
viola@merl.com
[2] Compaq CRL
One Cambridge Center, Cambridge, MA 02142
mjones@crl.dec.com

Abstract. This contribution introduces the purpose of the detector cascade. Furthermore, it explains in detail how this detector is constructed and how it works.

Keywords: cascade structure, sub-windows, false negatives, false positives

1 Introduction

In this section, we will introduce and explain in detail the purpose and the use of the Detector Cascade.

The detector cascade is the last part of the framework. It is the method which detects faces in images. This method combines increasingly more complex stages in a cascade structure, while discarding quickly backround regions of an image and focusing on promising object-like regions in order to achieve high detection rates, reduce the false rate and computation time.

2 The Detector

The detector consits of stages. Evaluating a specific image, all sub-windows are going to get evaluated by these stages, starting by the first one. Sub-windows which are not rejected by the initial stage are processed by a sequence of other stages, each slightly more complex than the previous, meaning that each classifier in the stage contains more features than the previous. If any classifier rejects a sub-window, no further processing is performed. If the last stage gives a positive result, the recognition of a face will be confirmed. The cascade structure works similar to a decision tree. The stages are constructed by training classifiers using Adaboost and then adjusting the threshold to minimize false negatives. The purpose of the first stages is to reject the majority of sub-windows. As a result, background regions are going to be discarded very quickly. The rest of the sub-windows as they are more "harder" than typical examples, are getting evaluated by the next stages which are more complex as they contain more features. The stages deeper in the cascade aim to reduce the false positive rate.

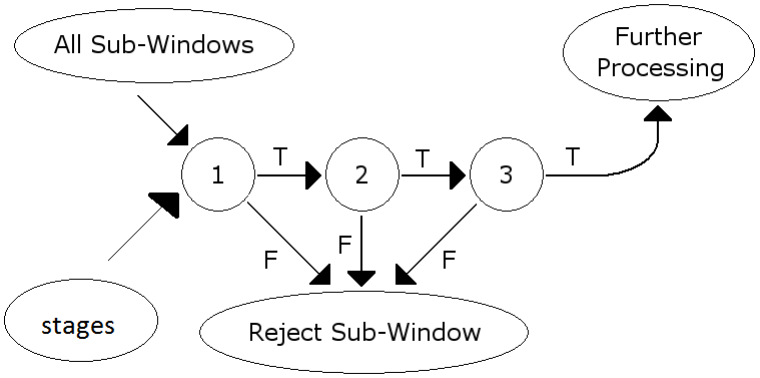

Fig. 1. Simple depiction of the detector cascade.

References

Viola, P., Jones, M.: Rapid Object Detection using a Boosted Cascade of Simple Features. Accepted Conference On Computer Vision and Pattern Recognition 2001

Results and Conclusion

Paul Viola[1] and Michael Jones[2]

[1] Mitsubishi Electric Research Labs, 201 Broadway ,8th FL
Cambridge, MA 02139
viola@merl.com
[2] Compaq CRL
One Cambridge Center, Cambridge, MA 02142
mjones@crl.dec.com

Abstract. This final part gives information about the results of experiments and performances of the Viola Jones Framework.

1 Results

A number of 4916 images were used for the training and their vertical mirror images in order to make the database as wide as possible. The images were normalized in order minimize the effect of different lighting conditions. Moreover, the faces in these images were scaled and aligned to a base resolution of 24x24 pixels.

The detector cascade consisted of 38 stages with a number of 6061 features. The first five stages consisted of 1, 10, 25, 25 and 50 features respectively.

The system was tested on the MIT+CMU frontal face test set. This set consists of 130 images with 507 labeled frontal faces. The detection rate of Viola Jones compared to previous approaches is slightly worse. But a very important advantage of the Viola Jones Framework is its incredible speed. It is at least 15 times faster than any previous approach.

2 Conclusion

The Viola Jones Framework operates very efficiently as it achieves high detection rates and reduces computation time. It is based on a complicated detection dataset which includes faces under a wide range of conditions like illumination, scale, pose and camera variation. Detectors for detecting other objects could be constructed in similar way.

References

Viola, P., Jones, M.: Rapid Object Detection using a Boosted Cascade of Simple Features. Accepted Conference On Computer Vision and Pattern Recognition 2001