Optimizing Viola-Jones Face Detection For Use In Webcams

By Theo Ephraim and Tristan Himmelman

Abstract

This paper will describe the face detection algorithm presented by Paul Viola and Michael Jones in their 2003 article titled, “Robust Real-Time Face Detection”. We will present the method and highlight the key innovations which make it successful. Then we will examine the problem of face detection for use in webcams. This will let us make several assumptions which allow speed optimizations to be made; the goal being to enable slow devices to achieve real-time face detection.

Introduction

The problem of detecting the presence and locations of faces in arbitrary images is a difficult one, but it has received significant attention over the last 20 years. The task is a specific instance of the object recognition problem in computer vision. Although much research has been done on this topic, object recognition is a complex problem and it is by no means solved. Current methods work quite well in controlled settings, but under real world conditions, where lighting is inconsistent, occlusions are common, and object pose is not controlled, the leading methods do not perform nearly as well. Regardless of the setting, object recognition algorithms are computationally expensive and thus, much of the current research done in this field is on the optimization of these algorithms for speed.

Face detection is an extremely active subset of object recognition because it has many applications, especially in the security and surveillance field. In our security obsessed world, government bodies are extremely interested in what surveillance tasks can be automated and in turn are investing heavily in this field. These tasks include tracking the movement of people in video feeds using static or moving cameras and facial recognition systems in which face detection is the first step. There are also numerous applications in the consumer realm, for instance auto-focusing on faces in digital cameras, Wii-style gaming, and head-tracking webcams. For the majority of these applications to be successful, detection must happen in real-time. This adds another level of difficulty to an already complicated problem.

The task of detecting a face in an image is not an easy problem because many difficulties arise and must be taken into account. For starters, faces will generally occupy very little area in most images and they are usually located arbitrarily. This means that face detection algorithms must search over all areas of any given image to be successful. Some methods do a preliminary scan over the image in an attempt to find the areas of interest early on. Furthermore, algorithms must take into account the fact that faces vary greatly in many aspects such as size, complexion and how they are accessorized. Further, faces in images can look very different depending on orientation and pose. For example, a face seen from a profile perspective will have a completely different set of defining characteristics than a face seen head on. There is currently no simple solution to address these issues, but one technique that is having some success is to train a detector using a set of images that spans many of these variations. Finally, another aspect that introduces a lot of difficulties is occlusions. Occlusions usually happen very unpredictably and are thus very hard to address.

Viola and Jones proposed a popular face detection method in their article “Robust Real-time Face Detection”. It is the current method of choice in the Open Computer Vision Library (OpenCV). Their method is based on simple features that encode image properties. Individually, the features are not very powerful, but they can be computed very rapidly. When the features are combined, they can detect faces quite accurately. In their article, Viola and Jones describe how one can use machine-learning techniques to construct sets of meaningful features that will detect faces. There are many other

proposed methods for handling the face detection problem, however most of them approach the problem in a similar fashion, using different ways of encoding the image properties such as statistical models and colour-based features and different machine-learning techniques.

The Viola-Jones method is quite fast already, but if we're able to make assumptions about the images that are being processed, optimizations can be made to further speed up detection. This paper will outline the Viola-Jones method and describe ways in which it can be optimized for use in a consumer oriented webcam environment. To make our work accessible, we have decided to create our face detection framework in Adobe Flash because it requires very little set up and includes built-in tools for accessing webcams.

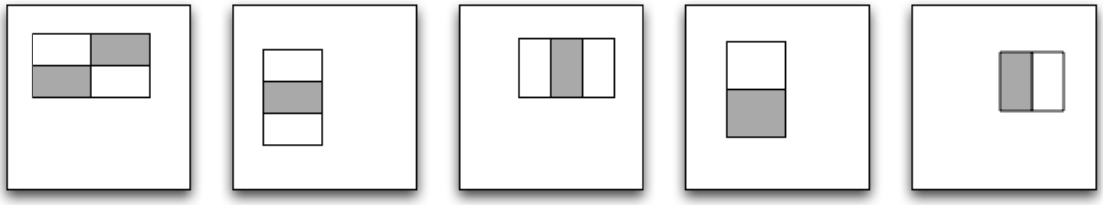
The Viola-Jones Method

The method proposed by Paul Viola and Michael Jones in 2003 in their paper, "Robust Real-Time Face Detection" was a significant step forward in the face detection field. Similar to other previous methods, they used machine-learning algorithms to select a set of simple Haar-like features which they combined into an efficient scalable classifier. Their paper, however, introduced three key innovations that enabled their detector to achieve performance boosts over previous systems. The first was the use of a new image representation called the "integral image" for faster feature computation. The second was the use of the AdaBoost (adaptive boosting) machine-learning algorithm as a means for selecting simple and efficient classifiers. The last was a method of combining classifiers into a "cascade" that quickly eliminates background regions and focuses computational attention on more promising areas of the image.

Haar-Like Features

Haar-like features are rectangular digital image features that get their name from their similarity to Haar-wavelets. Working with individual pixel intensities is computationally expensive so these features, introduced by Papageorgiou et al in 1998, provide a method for encoding image properties in a form that can be computed much more quickly. Simple Haar-like features are composed of two adjacent rectangles, located at any scale and position within an image, and are referred to as "2-rectangle features". The feature is defined as the difference between the sums of image intensities within each rectangle. Viola and Jones also extended this set by defining similar features composed of 3 and 4 rectangles (see Figure 1). Additionally, Lienhart and Maydt defined tilted versions of these features to enrich the possible feature set in an attempt to form better classifiers. These types of features are quite coarse when compared to alternatives such as steerable filters, however, their computational efficiency more than makes up for their limitations.

Figure 1 – Types of Haar-like features (4,3,2 rectangle)



Integral Image

Haar-like features can be calculated extremely quickly by using an image representation called the integral image. The integral image is an application of summed-area tables; a concept introduced by Crow in 1984, usually used in computer graphics. The integral image can be defined as:

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

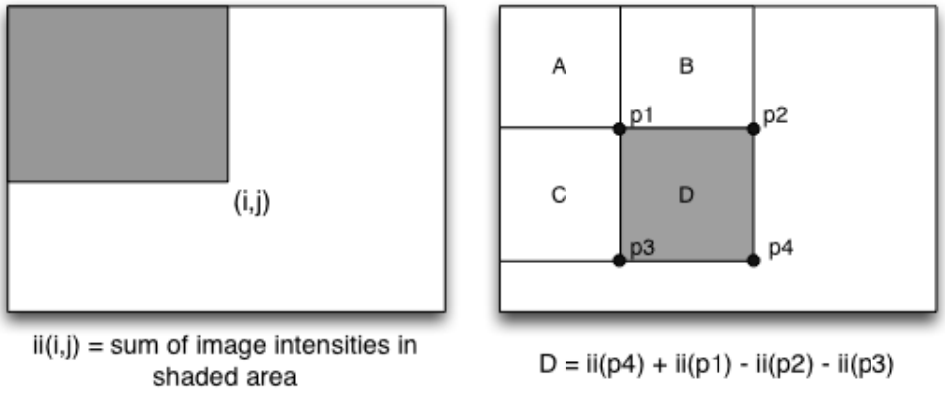
Where ii(x,y) is the integral image and i(x,y) is the original image intensity. The integral image can be calculated in a single pass using the following recurrences:

$$s(x,y) = s(x,y-1) + i(x,y)$$

$$ii(x,y) = ii(x-1,y) + s(x,y)$$

Where s(x,y) is the cumulative row sum and we have the following base cases: s(x, -1) = 0 and ii(-1,y) = 0. Using the integral image, each feature can be calculated in *constant* time. The sum of any rectangle can be calculated in only four memory references.

Figure 2 – Integral image and rectangle feature calculation



Furthermore, because Haar-like features use rectangles that share corners, the number of total memory references needed to compute a feature can be further reduced. For example, 2, 3 and 4-rectangle features require only six, eight and nine memory references respectively.

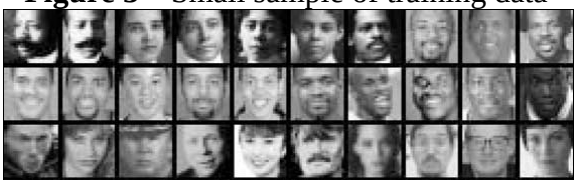
Feature Selection

The set of possible features in a given sub-window is huge. For example, there are over 160 000 features in any given 24 by 24 pixel window using only 2, 3 and 4-rectangle features. This set is much larger than the number of pixels in the window and is many

times overcomplete, however, Viola and Jones hypothesized (and discovered through experimentation) that a very small number of these features can be formed into an effective classifier. The selection of these features is not an easy task. Using a large set of hand-labeled training data and a possible feature set, many different types of machine-learning techniques can be used to train a classification function. The Viola-Jones method uses a variation of the AdaBoost algorithm, formulated by Freund and Schapire in 1995, to select a small set of critical features to form an efficient classifier. The fine details of the AdaBoost algorithm are outside the scope of this paper, but we will present a basic overview of the key concepts.

Regardless of the approach used to train a classifier, the training set must be varied in order to handle real-world conditions. The set used by Viola and Jones is composed of 4916 hand-labeled faces that have been scaled and aligned to the size of the detector, in their case 24 by 24 pixels. The faces include various lighting conditions, skin complexions, and facial variations such as glasses and facial hair. A limit must be set to the amount of rotation (both in plane and out of plane) of the faces within the data set in order to obtain consistent results. If we also want to find rotated faces we can easily train a new classifier on a set of rotated faces, as long as they are within some range. For example, OpenCV includes separate classifiers trained to detect frontal and profile faces. The key point here is that for a classifier to be effective the set it is trained on must contain a good range of facial variations yet the orientation of the faces must be fairly consistent.

Figure 3 – Small sample of training data



No single feature can be used as an effective classification function. Combining several features produces a better classifier, but it is still inconsistent, and thus referred to as a weak classifier. The AdaBoost algorithm creates stronger classifiers by searching the set of all weighted combinations of weak classifiers and selecting the most successful combinations. The newly obtained strong classifier is combined with the optimal threshold which enables it to best separate faces from non-faces. The algorithm strengthens the classifiers it creates by repeatedly re-weighting the test set to favor test cases that were incorrectly classified in the last round of training. Freund and Schapire proved that the training error of a strong classifier approaches zero exponentially with the number of training rounds. It is important to note that due to the greedy selection method used in the training process, the AdaBoost algorithm results in a small feature sets with significant variety.

In our project we did not experiment with the training process. We simply used the classifiers already created for the OpenCV library. However, there are some noteworthy results from Viola and Jones' experimentation that demonstrate the efficacy of this approach. Firstly, initial experiments demonstrated that an effective classifier could be

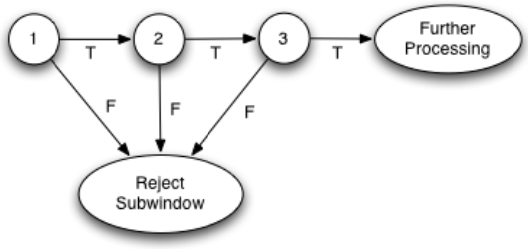
built from only 200 features. This simple classifier yielded a detection rate of 95% and a false positive rate of 1 in 14084, quite impressive but not effective enough for a production application. Secondly, the initial features chosen by the algorithm are easily interpreted by simple observation. They focus on the two simple facial properties: the difference in brightness between the nose and eyes, and the difference in brightness between the eyes and cheeks.

Figure 4 – First features selected by AdaBoost

Attentional Cascade

Perhaps the most important innovation of the Viola-Jones method is the advent of the attentional cascade. The idea is to focus first on the removal of negative regions of the image while including almost all of the positive ones. A simple version of this idea was implemented by Rowley et al. in 1998. Their method was to use two neural networks, one complex network that focused on small regions of the image and detected faces with a low false positive rate, and a simpler faster network that focused on larger regions but had higher false positive rates. The algorithm first uses the faster neural network to select regions of interest before running the slower, more complex network to pick out the faces. The Viola-Jones method extends this approach into a 38-stage cascade of classifiers in order to minimize total computation. The initial stages are composed of fewer, simpler features and are created by adjusting AdaBoost to favor false-positives and maximize detection rates. The later stages of the cascade use more complex classifiers to further reduce overall false-positive rates. Sub-windows of the image are only accepted as faces if they have passed every stage of the cascade. Running the cascade in this specific order allows many regions to be discarded early in the cascade so they will not require any further processing.

Figure 5 – Attentional cascade flow diagram



Algorithm and Implementation

Assuming the training process is complete and a classifier cascade has been created with desired properties, the detection algorithm is fairly straightforward. Simply scan all possible sub-windows of an image at a range of scales, running the cascade on each window. If a sub-window passes the final level of the cascade then the sub-window likely contains a face. It is important to note that the detector is invariant to small changes in scale and location, so there will often be many hits centered around each face. To filter out these duplicated results, overlapping hits can be averaged together to form a single detection. Clearly, running the cascade on every possible sub-window is computationally expensive. Thus the amount we adjust scale and position between tested sub-windows (scanning factor) must be adjusted to achieve a balance between speed and accuracy. Viola and Jones achieved best results using a scaling factor of 1.25 and a scanning factor proportional to the current scale.

In practice, there are a few steps where normalization must occur. First of all, image intensities of both training and test images must be normalized to the same scale. Secondly, when running the cascade on scaled sub-windows, the rectangle sums within each feature must be scaled accordingly. Lastly, due to different lighting conditions, training should be done on variance normalized images, and thus, test windows must be variance normalized as well. Variance can be computed using the following equation:

$$\sigma^2 = m^2 - \frac{1}{N}\sum x^2$$

where σ is the standard deviation, m is the mean, N is the window area, and x is each pixel image intensity. The integral image concept is again useful here as the mean can be easily calculated from the original integral image and a second integral image, that holds the sums of image intensities squared, can be created for use in the summation part of the variance equation. Furthermore, variance normalization can be achieved efficiently through a single multiplication with the feature sum instead of operating on each pixel.

Optimizations

The Viola-Jones method is quite fast, but as low power systems such as mobile devices gain access to cameras, further speed optimizations must be made to achieve real-time detection. In order to make these optimizations without losing accuracy, we must simplify the problem somehow. The following will focus on optimizations that can be made for face detection within in a webcam setting.

Assumptions & Their Implications

(1) Because we are dealing with a webcam, we can assume that it is probably being used for video chatting or simple games.

Thus we already have a basic idea of the types of images being processed. There is probably a single face, or if there are multiple faces, the largest will be the main user of the computer and therefore the one of interest. Therefore, we can limit our detection to a single face and stop processing after a single face is found. Within our scanning process, it makes sense to scan from larger scales down to smaller ones. This is both because the

largest face is mostly likely the one of interest and because scanning at a larger scale is faster because there are fewer sub-windows to process.

(2) Typically, webcams are built into monitors or attached in a comparable location. This generalization gives us an idea of the position of the camera in relation to the user and allows us to make an assumption about the scale and location of faces within the image. There is certainly a limit to how large or small a face might be within an image based on where a user can be in relation to the camera while comfortably using their computer. Thus, based on the specific environment and application, we can set upper and lower bounds on the scales that must be checked. Through informal observation, we determined the size of faces would be within the range of 20-90% of the height of the image (within a 4:3 aspect ratio). Additionally, the user is most likely looking directly at their monitor and is therefore oriented towards the camera with their head upright. This fact justifies the use of a classifier trained on images of frontal, upright faces. Further, if the user's head is oriented otherwise, they are probably looking away from the screen and detection would thus be unimportant.

(3) Usually, we will be processing a continuous video feed. This allows us to use information from previous frames within the processing of subsequent ones. Specifically, the last determined location and the scale of the face of interest. Assumption 1 allows us to stop processing after the single face is found. We can therefore use the last determined location and scale to center our search in order to increase performance. Additionally, movement of the face between frames is more likely to be within the image plane than out of it. Thus we begin our search at the location and scale of last detection, and radiate outwards, first by position, then by scale. To clarify, we check all sub-windows a given distance from the last location at the last scale, increasing the distance until it exceeds the image bounds. We then repeat the process after adjusting the scale value. New scale values are obtained by alternating between choosing larger and smaller values than the last scales checked, adjusting by a given factor. In situations where we have no information from previous frames, assumptions 1 and 2 allow us to formulate a decent guess in that the face is probably around the center of the image at a given scale. Sitting comfortably at a computer, we determined that a face takes up approximately 30% of the image height.

Further Improvements

As mentioned earlier, the detector is invariant to small changes in size and location. In the Viola-Jones method, this problem can be dealt with by averaging the locations and scales of overlapping hits into a single, more accurate detection. Obviously the more scales scanned and the smaller the scanning factor, the more accurate and consistent the final averaged detection will be. However, within our proposed improvements, it is vital that processing can stop after a single face is found. This leads to faster, but less consistent results. Now we will propose a method to increase accuracy of results while maintaining the performance improvements gained using our improvements.

Firstly, larger scanning and scaling factors must be found to minimize computation while preserving at least one hit per face. After this initial detection is made, we can improve

the location and scale of the hit as follows. Using a smaller scanning factor, run the cascade on sub-windows moving outwards in each orthogonal direction until detection fails. The face should be roughly centered within the bounds of positive detections. Next, using a smaller scaling factor than was used initially, run the cascade on sub-windows centered at our new estimate, increasing and decreasing scale until detection fails. The average of the scales of accepted sub-windows should provide a more consistent and accurate scale. Additionally, because the sub-windows being checked during this process are likely to be faces, we can run the cascade starting at a higher level. Using this method allows us to avoid scanning many of the negative sub-windows altogether, while still providing consistent and accurate results.

Results & Conclusions

We have implemented both the original algorithm and our improved version using Adobe Flash. We chose this platform because it is easily deployed across multiple operating systems, is accessible over the internet without any installation and provides simple webcam support. Additionally there is a large active development community that may have uses for a native face detector, as current flash-based face detectors actually use C++ and require either installation on the client machine or processing on a server. The result of our choice, and part of the challenge associated with this project, is that flash is quite slow. On the upside, this slow environment provides a good simulation of low-powered devices where this technology may be useful and therefore provides a good testing ground for our algorithm.

Even though there is still plenty of room for improvements to our implementation, initial results are very promising. Using the original method, detection occurs at a rate of 1-2 frames per second which is not quite quick enough to be displayed in real-time. Our improvements to the method, increased performance to a more reasonable rate of roughly 4-5 frames per second. Although in-depth statistical analysis must be done after all possible improvements are made in order to truly assess the effect of our changes, we are confident that our adjustments do provide a significant performance boost.

We have created two interactive demonstrations to display our work. The first uses our improvements to the Viola-Jones method and achieves face detection in real-time. The second demonstration uses the unimproved version of the Viola-Jones method to detect faces in image snapshots. The demos are accessible at the following URLs:
http://theoephraim.com/old/camdemo.html
http://theoephraim.com/old/camdemo2.html

In conclusion, the Viola-Jones face detection algorithm is an extremely powerful method that operates quite quickly to achieve very good results. Even in a slow programming environment, with the help of a few assumptions, it can achieve real-time results. Our modifications to the algorithm could easily be implemented in any setting and should prove to be especially valuable for settings where speed is important. We plan to continue development on our implementation and hopefully create the first successful face

detector written natively in Flash. We are planning to create an open-source face detection toolkit and release it to the Flash community.

Bibliography

Crow, F, “Summed-area tables for texture mapping”, in Proceedings of SIGGRAPH, 18(3):207-212, 1984

Freund, Yoav and Schapire, Robert E. “A decision-theoretic generalization of online learning and an application to boosting”, Journal of Computer and System Sciences, No 55, 1997

Lienhart, R. and Maydt, J., "An extended set of Haar-like features for rapid object detection", ICIP02, pp. I: 900-903, 2002

Papageorgiou et al, “A General Framework for Object Detection”, International Conference on Computer Vision, 1998

Rowley et al. “Neural network-based face detection” IEEE Patt. Anal. Mach. Intell. 20:22-38

Viola, P. & Jones, 1 J. “Robust Real-Time Face Detection”, 2003