Exploring the suitability of the Viola-Jones framework for counting people

Chris Ashton
Aberystwyth University
cba1@aber.ac.uk (student no. 110059347)

Abstract

This paper explores the effectiveness of the Viola-Jones algorithm for detecting faces and the suitability of applying the algorithm to the problem of counting people, concluding that a viable solution may be possible by combining classifiers and/or by training classifiers for specific installations.

1. Introduction

Robust Real-Time Face Detection (Viola and Jones, 2004) proposes an automated object-detection framework that can be run in real time. Though the framework can be trained to detect any object, the paper specifically tackles the problem of face detection.

Face detection isn't a trivial task to accomplish computationally, as it raises many distinct problems. We need our detector to take into account differences in age, gender and race, faces hidden behind glasses or beards, and images of faces at different angles and at different light exposures.

The human eye can easily look at an image and distinguish a real person from, say, a painting depicting a person, but how does software make that distinction? Should it?

1.1. Uses for face detection

The world wide web can benefit from the advancement of face detection techniques, including the ability to search by image, or prompting users of social media sites to tag their friends in uploaded images that have had faces detected within them.

This being an *efficient* algorithm for face detection means that detection software can be run on low-power devices; particularly useful in digital cameras with little computational power, for instance, allowing them to automatically adjust the focus to try to provide the clearest image of the faces in view.

Face detection is a step towards facial recognition. By efficiently identifying the regions of images that contain faces, we're able to cut down the time taken to match a face to an identity, as this intensive processing does not need to be computed across every part of the image. Although face detection *alone* is useful, advances in face detection is most exciting in applications where face detection is the *bottleneck*. Fast, reliable face detection is worth investigating for its potential contribution to the economy, the well-being of the environment, and the spurring of hitherto unknown technological advances.

1.2. The Viola-Jones algorithm

Viola and Jones introduced three main contributions to the field of object detection:

1) *Integral image*: created by computing the sum of the grayscale values of all pixels above and to the left of every pixel in the image, allowing any rectangular sum to be computed in "four array references" [1]. This means that we can detect faces at different sizes and distances in constant time.

2) *Classifier*: a small set of critical features, ensuring fast classification in comparison to searching images for every possible feature.

3) *Cascade classifiers*: a method for combining successively more complex classifiers, ensuring a good balance of high detection rates and computational efficiency. These are pre-computed by running a variation of the AdaBoost learning algorithm against a labelled dataset of positive and negative images and extracting the common features. Viola and Jones were "the first to introduce the concept of boosting to the computer vision community, which involves training a series of increasingly discriminating simple classifiers and then blending their outputs" [2].

1.2.1 The Viola-Jones method at runtime

The input image is converted to grayscale so that each pixel in the image can be represented by an intensity value of 0-255, before being computed into its integral image. This image is split into sub-windows and scanned for features defined in the first classifier of the cascade. Sub-windows that pass the first classifier are then processed by the second classifier, and so on, until only sub-windows containing the detected objects remain.

Viola-Jones features differ to Haar basis functions in that they rely on 2-4 rectangular areas to represent a feature.

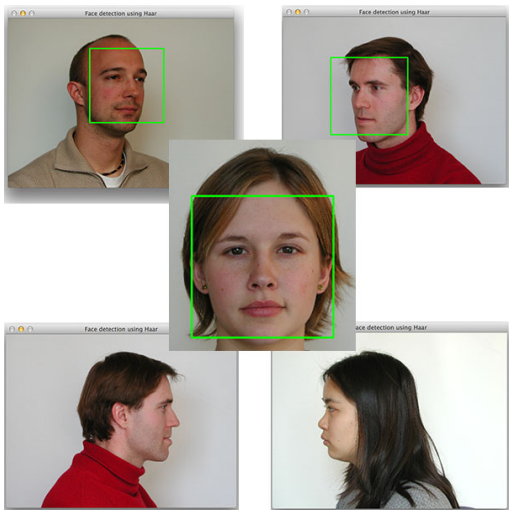

Figure 1. Running the Viola Jones algorithm with a full frontal face classifier leads to faces in profile not being detected.

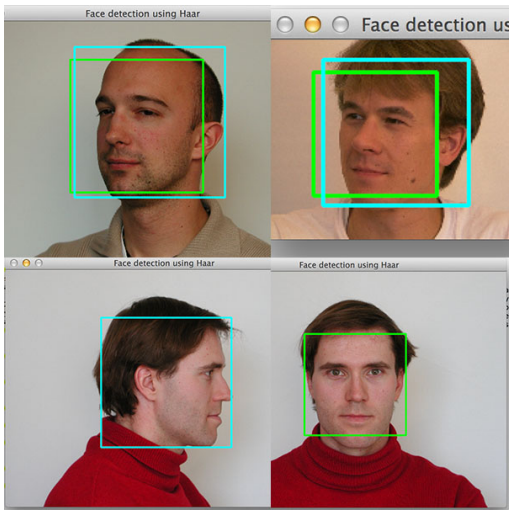

Figure 2. The use of multiple classifiers allowed all of the faces to be detected.

These features can be symbolised by black and white rectangles which map to the natural dark and light areas on a typical image of the object we're trying to detect. The algorithm takes the sum of the pixel values beneath the white regions minus those of the black regions and the area is said to have that feature if the sum is within a certain threshold. Each feature needs to "perform at a level only slightly better than chance" [3] to increase the overall detection accuracy and be worth incorporating into a classifier.

2. Critique of proposed method

2.1. How well the method solves the problem

I tried running the algorithm against a collection of faces downloaded from a university website [4]. The algorithm had a 100% success rate with the frontal face images, but did not pick out the faces that were taken from the side (see Figure 1).

Viola-Jones is not very robust to variations to the norm. According to Yang Cai, it "does not work well with yawn detection due to the variations and dynamics of the mouth." [5] Also, "the algorithm failed when lighting was poor, when there were multiple faces in the background of images, and when the person was too far from the camera".

Running the algorithm with two classifier files (*OpenCV's* 'haarcascade frontalface default' and 'haarcascade profileface') improved the accuracy but slowed down the processing time a little, due to the algorithm having to run twice (once for each cascading classifier). See Figure 2, where the portrait images were highlighted in blue. I also had a new issue, where a face was picked up by both the frontal and portrait classifiers, but it did detect all of the sample set.

Figure 3. The *OpenCV* classifier was reasonably accurate at detecting faces in Facebook photos.

These sample images were taken in controlled conditions. I decided to test the *OpenCV* [6] frontal face classifier and the Viola-Jones algorithm with a sample of photos from social media, whose lighting conditions and camera angles tend to be more random.

Figure 3 shows that the algorithm works well with the default parameters, provided faces are angled towards the camera and aren't too far away. I tried tweaking the parameters to see if I could improve the performance and decided to use the wedding photo with the largest number of people as my constant.

Reducing the minSize parameter from (30, 30) to (10, 10) led to ten detected faces and no false positives (Fig-

Figure 4. Tweaking the parameters doubled the face detection success of the wedding photo.

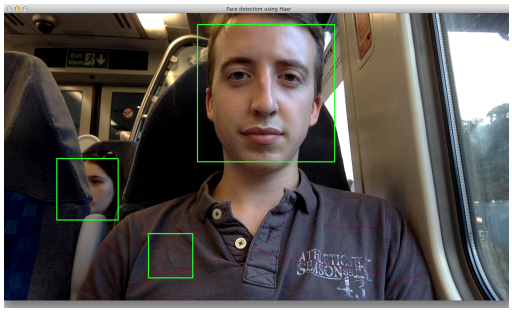

Figure 5. A screenshot showing the Viola-Jones algorithm detecting faces on a live webcam feed.

ure 4).

However, keeping minSize at (10, 10) and reducing minNeighbours to 1 led to the most number of faces being detected (37), but with a large number of false positives (15).

This varying level of accuracy shows the importance of setting detection parameters intelligently, even with a well-trained classifier. Overall, the Viola-Jones algorithm and classifier did a reasonable job of detecting faces, but it is very dependant on good parameter settings and the input images capturing people at the correct angle.

2.2. Successes of the method

One of the greatest strengths of the Viola-Jones algorithm is its efficiency at run time. It can be run in real time, even on low-power devices [1]. Figure 5 shows the Viola-Jones face detection algorithm running at frame rate on a Macbook Pro (2.8 GHz Intel Core i7 processor, 16 GB 1600 MHz DDR3 RAM).

Since the publication of the Viola-Jones paper, numerous improvements have been suggested, including the concept of the "rotated Haar-like feature" [7]. The original Viola-Jones algorithm was not robust to rotation, so any object that is regularly captured at different rotations (such as a human hand) might be difficult to detect using normal Haar-like features.

The advantage of the 45 degree angle twist is that "the diagonal coordinates of the pixel will always be on the same diagonal set of pixels", at any scale, meaning that "the number of different sized 45 degree twisted features available is significantly reduced as compared to the standard vertically and horizontally aligned features."

There is a further advantage in that the integral image calculations can be offloaded to the GPU by writing a custom shader for it. This is typically faster than calculating the integral image on the CPU, especially for large image sizes, "allowing more complex classifiers to be implemented in real time", and therefore more accurate detectors.

2.3. Failure modes of the method

The importance of having a diverse dataset cannot be understated. I refer to the Pentagon's failed multi-million dollar mainframe whose aim was to detect camouflaged tanks hiding in the trees. [8].

The problem was that the training dataset contained photos taken in *consistent conditions*, and the classifier had trained itself to look for the biggest contrast between the positives and the negatives: the brightness of the sky. It would return true or false based on the weather conditions, rather than whether or not there was a tank in the image.

In addition to a diverse dataset, some heuristics are required at the classifier-training stage. AdaBoost must be provided with parameters including a minimum detection rate, maximum false positive rate, height and width ratio of the object we want to detect, and the number of stages required of the cascade. Requesting too many stages is likely to lead to an over-fit for the data, *i.e.* it will accurately detect the object in the images used for training but will be poor at detecting objects "in the wild". However, too few classifiers will detect many false positives.

The probabilistic nature of evaluating features means that the training algorithm often takes days or weeks to generate a cascade of the required accuracy. However, the benefit of this long training stage is that, once computed, the XML representation of an object's features is transferable and can be computed against new input images very quickly.

3. Application of the proposed method to the scenario

The Viola-Jones algorithm alone is not suited to scenarios where we have to *maintain state*. To illustrate this, let us apply the algorithm to the "Safety" scenario, which requires us to keep a tally of everyone who has entered or exited a venue.

To prevent the same person being counted from one frame to the next in a live video stream, we'd need to add face recognition technology. Assuming that the venue's

entrance is also its exit, the person's direction of motion will determine whether the tally ought to be incremented or decremented. We now need to not only detect and recognise a person, but track the direction in which they're moving!

My hypothesis is that the Viola-Jones algorithm is best suited to scenarios where no state needs to be maintained. For instance, let's examine the "Money" scenario which counts the number of people in a group so that the organisation can charge for the right number of attendees. In this case, a photographer could take a single image of the group and use face detection to count the number of people on a *group-by-group, image-by-image basis*.

For the purposes of this paper, I'll examine the third example given in the scenario: that of a supermarket wanting to automate the opening and closing of checkout lanes based on the number of customers queuing. A frame could be sampled once every few seconds, the number of faces or bodies counted and the number of required checkouts calculated. Again, this is stateless: we only care about making a quantitative decision based upon the number of customers in a queue at that *specific point in time*.

3.1. Camera installation

If we were to use the Viola-Jones algorithm for body detection, the camera would need to be installed at a suitable height and angle so as to maximise the chance of detecting peoples' bodies frontwards on. In the case of the supermarket scenario, this would mean positioning the camera near the exit and facing the checkout.

Depending on the logic of the program, I'm confident that one camera should be able to accommodate two checkouts, thus the number of cameras required in the system would be $n/2$, where n is the number of checkouts in the store.

3.2. Using the classifiers on typical CCTV images

I sourced 48 random supermarket CCTV images, all of which contained people, through scouring the web. I then ran my Python script [9] once using the *OpenCV* face cascade classifier and once using the *OpenCV* full body cascade classifier.

The face classifier detected faces in 28 of the images whereas the body classifier detected bodies in 16. These also contained 23 and 11 false positives respectively.

The top right portion of Figure 6 shows a false positive in the wine shelving but the real human face was missed entirely. The bottom left portion shows the challenging conditions of detecting faces when customers are naturally inclined to be facing away from the camera. Finally, the bottom right portion shows that even small inconsistencies in the angle of the face can lead to false negatives.

Bodies angled towards the camera were sometimes detected, as seen in the top left portion of Figure 7. However,

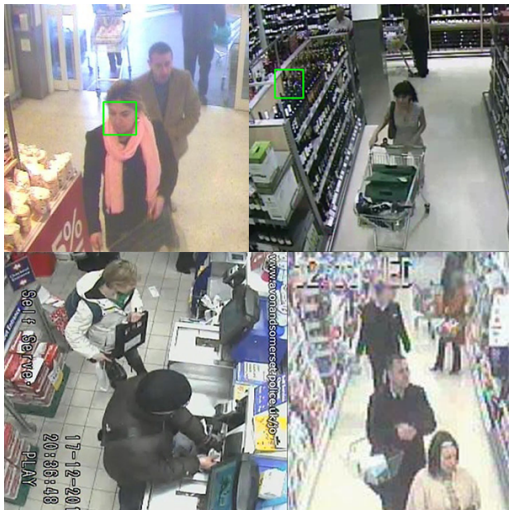

Figure 6. The default face classifier has a limited detection rate of faces over a range of CCTV systems.

Figure 7. The body detector had even poorer detection rates.

the top right and bottom left portions show that the nature of people shopping in the supermarket (*e.g.* holding baskets and wheeling trolleys) can be problematic for body detection. Finally, the bottom right portion shows that even relatively unobstructed images can still have false negatives.

3.3. Training a classifier for an individual installation

The position and angles of the installed CCTV cameras varied considerably in my dataset, as did the captured image quality. Perhaps it is unfair to expect a general body/face detector to be effective in such volatile environments.

I wondered if a cascade classifier could perform better if it could be calibrated according to the image quality and

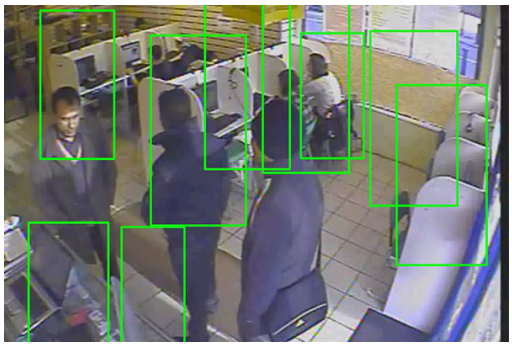

Figure 8. My custom classifier was poor at detecting people.

viewing angle of an *individual camera*. Sample images could be collected from the camera after its initial installation, any people in the sampled images could be labelled, and that dataset could be used to train a camera-specific person detector.

Freely available supermarket CCTV footage proved difficult to find, so I downloaded 23 minutes worth of CCTV footage [10] of what appeared to be an internet cafe.

I extracted a frame from the video every five seconds: 282 images in all. I then manually cropped every person from every frame, ending up with 279 cropped positives to train the classifier with. Finally, I cropped 253 negative images from the same sample of 282 images.

I followed a tutorial [11] to train my own classifier. This required compiling a vector from my list of cropped positive images, creating 1500 samples from my original 279 positive images.

Training a classifier often takes days or weeks - mine took around 23 minutes. I believe that this was down to the algorithm over-fitting the positive samples. Many of my samples were very similar (*e.g.* many were of the same person sitting at a computer), and the background scene throughout the feed obviously remained identical. It didn't take long for the classifier to be trained to the specific environment of the cafe.

Figure 8 is the result of running my classifier over one of the frames in the video.

It managed to detect the two people sat at the back of the cafe with their computers. These two people didn't move much in the 23 minutes of video, and made up quite a proportion of my positive samples, so they were correctly identified.

The rectangles surrounding them are elongated because of the parameters passed to the "createsamples" and "traincascade" programs. The *OpenCV* classifier trainers are somewhat limited in that they require all positive samples to have the same aspect ratio. However, as can be seen in the screen shot, the aspect ratio of a person when they're sitting, standing, or partially obscured behind a counter varies

Figure 9. One of the positive samples used to train my classifier. The uncropped laptop corrupted the feature learning process.

Figure 10. The same footage was passed through the *OpenCV* full frontal face cascade classifier.

considerably. My parameter choices (a width of 40 and a height of 80) were somewhat arbitrary.

In every frame, my custom classifier consistently detected a person in the bottom left corner. I believe this was because this was the most feature-rich corner: the sharp edge of the laptop makes for an easily detectable feature. A number of my sample positive images were of a man leaning against the counter, with the laptop in the crop (see Figure 9): AdaBoost simply picked up the most detectable feature.

I passed the same CCTV footage through the Haar cascade "fullfrontal" face classifier (Figure 10). It had fewer false positives, but again we had numerous false negatives. Neither classifier worked very well for the scenario of counting people in this particular cafe, where people form multiple body positions and face multiple different angles.

4. Conclusion

In general, the Viola-Jones framework is a good object detector, provided that the cascade classifier is trained with

good data and parameters and that the input images are of similar orientations and poses as the training images. However, the framework is not very robust to noise or variations to the norm, limiting its usefulness in the "real world".

Given the efficiency of the algorithm, Viola-Jones is suited to real time applications where object detection is desirable, but where accuracy (or lack thereof) is not a major financial or health and safety risk.

In terms of the problem of counting people, the algorithm doesn't work well "out of the box". However, it could be a good basis for a solution.

In Figure 1, some faces were not detected because they were not captured at the expected angle. In Figure 2, those faces were captured because I used multiple cascade classifiers to detect both frontal and profile views of faces. Similarly, I believe that multiple cascade classifiers could be used to detect people in supermarkets with typical supermarket noise. For example, full-frontal body cascade classifier could be used in tandem with a classifier that detects a person partially obscured by a trolley, plus a classifier detecting a person bending over to pick up groceries. The combined efforts of these classifiers could lead to a fairly accurate detector.

Although my first attempt at training a classifier finely tuned to a specific environment was not very successful, I believe that given enough time, the correct training parameters, and better positive and negative samples, a proprietary cascade classifier for a specific supermarket CCTV installation could also be a reasonably accurate person detector. This, perhaps in combination with other cascade classifiers, could provide an accurate enough detector to become a viable solution to some of the problems outlined in the scenario.

The key consideration is that the Viola-Jones algorithm alone is not well suited to problems which require the program to maintain an internal state. Instead, Viola-Jones is a reasonable solution to self-contained snapshots of problems which can be tackled at arbitrary intervals.

5. Self-evaluation

I have a strong understanding of the Viola-Jones algorithm, shown by my in-depth discussion of the algorithm in section *1. Introduction*.

In order to appreciate the computational speed of the Viola-Jones algorithm, I downloaded *OpenCV* and ran the algorithm against a live input feed from my webcam. Code examples provided in *Programming Computer Vision with Python* [12] were useful for testing the algorithm. I examined the robustness of the algorithm and its classifiers by passing in images from social media which were captured in a variety of conditions, and I adjusted the function parameters to assess how it affected the accuracy of the detector.

Finally, to fully understand the algorithm learning process, I followed a tutorial on creating my own cascade classifier to try to create a person detector custom to a specific environment. This required aggregating a large dataset of images, identifying positive and negative samples, and modifying the learning algorithm parameters to try to attain reasonable degrees of accuracy.

Looking at my work constructively, I could have invested more time in training my classifier and using a final percentage of accuracy in helping to form my conclusion. However, this was beyond the scope of the assignment.

Overall, I believe my work is deserving of an A grade.

References

[1] P. Viola and M. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004. 1, 3

[2] R. Szeliski, "Computer vision: Algorithms and applications," pp. 1–789, 2010. 1

[3] M. J. F. Daniel Westreicha, Justin Lesslerc, "Propensity score estimation: neural networks, support vector machines, decision trees (cart), and meta-classifiers as alternatives to logistic regression," *Journal of Clinical Epidemiology*, vol. 63, no. 8, pp. 826–833, 2010. 2

[4] M. I. of Technology, to the Center for Biological, and C. Learning, "Face recognition database." `http://cbcl.mit.edu/software-datasets/heisele/facerecognition-database.html`, 2004. [Online; accessed 20-October-2014]. 2

[5] Y. Cai, "Ambient diagnostics," 2014. 2

[6] itseez, "Opencv." `http://opencv.com/`, 2014. [Online; accessed 10-October-2014]. 2

[7] C. Messom and A. Barczak, "Stream processing for fast and efficient rotated haar-like features using rotated integral images." 3

[8] N. Fraser, "Neural network follies." `https://neil.fraser.name/writing/tank/`, 1998. [Online; accessed 09-November-2014]. 3

[9] C. Ashton, "Viola-jones experiments." `https://github.com/ChrisBAshton/viola-jones-experiments`, 2014. [Online; accessed 14-November-2014]. 4

[10] M. R. Islam, "Cctv footage of london." `https://www.youtube.com/watch?v=MlMerr9H9xw`, 2013. [Online; accessed 09-November-2014]. 5

[11] T. Ball, "Train your own opencv haar classifier." `http://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html`, 2013. [Online; accessed 09-November-2014]. 5

[12] J. E. Solem, "Programming computer vision with python," pp. 257–277, 2012. 6