ICS3206
Fast Frontal Face and Eye Detection using Viola-Jones Object Detection

Ryan Camilleri (328400L)

Faculty of ICT

University of Malta

Bs.C IT A.I. Third Year

Contents

Statement of Completion

Item	Completed (Yes/No/Partial)
Viola-Jones technical discussion	Yes
Artifact 1	Yes
Artifact 2	Yes
Comparison to alternative approach(es)	Yes
Application of method to other scenarios	Yes
Experiments and their evaluation	Yes
Overall conclusions	Yes

List of Figures

List of Tables

1 Viola-Jones technical discussion

This section gives a brief overview on how Viola-Jones Object Detection works. Useful techniques proposed by P. Viola and M.J. Jones in 2001 [4] and later updated in 2004 [1] will be described. The discussion ends by showing how this technique is still relevant in today's state-of-the-art by showing its usefulness within real-time applications.

1.1 Haar-like Features

It is paramount to understand what Haar-like features are, prior to explaining the cascade classifier methodology. Owing their name to their similarity with Haar wavelets [5], a Haar-like feature is very similar to a kernel used in convolutional neural networks (CNNs). These features are used to extract line and edge information from images depending on the type of feature.

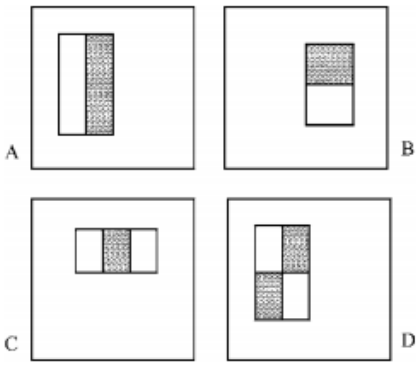


Figure 1: Different types of Haar features used for face detection. [1]

The above *Figure 1* shows the different types of features used within P. Viola's and M.J. Jones's solution [1]. By summing the pixels in the white and grey rectangles and finding the difference between the values, edge features (A and B), line features (C) and other hybrid features (D) are obtained from the input image. As claimed in their paper, instead of directly using pixels, Haar features offer the added benefit of being significantly faster, also having the ability to represent domain knowledge easier than pixel-based systems. However, since summation of pixels is required for each feature, the technique is

computationally expensive since it would require reference to all pixel values within the feature window. Taking the example of a small 24x24 pixel window, a total of 160 000 features could be obtained [1] which would require a lot of processing time to compute the whole set of features.

1.2 Haar Cascade Methodology

1.2.1 Integral Image

As a solution to the expensive feature computation, the authors proposed the integral image. Each point within the integral image is the sum of all the pixels above and to the left of the point in the original image. By making use of this technique, feature calculation time is drastically reduced since instead of having to find the sum of white and grey pixels for each feature, the sum of pixels would have already been found and represented within the integral image. The number of references made to the integral pixels array, corresponds to the type of feature being computed.

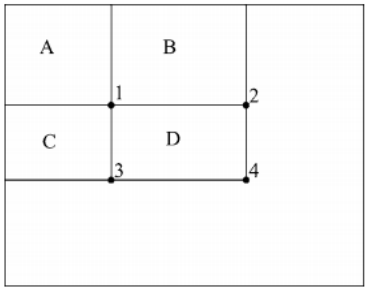


Figure 2: Integral images at four points. [1]

A total of 9 array references are required when computing the four-rectangle feature (D) shown in *Figure 1.* Assuming that the rectangles within the Haar feature are adjacent to each other, the feature D can be split into four parts as shown in the above *Figure 2.* To find the sum of rectangle D, all the four integral points are used for computation and calculated as 4+1-(2+3). The difference between point 3 and 1 is calculated for rectangle C. Similarly the difference between points 2 and 1 is found for rectangle B and lastly, the value at point 1 is used for rectangle A. From the above process, the usefulness of

the proposed integral image is showcased perfectly as any rectangular sum is shown to be computed from a maximum of only 4 pixel references.

Through using the integral image, the efficiency of feature calculation is greatly increased but now, a new problem arises. If the current bag of features were to be applied on every image for classification, the time needed to classify a single image would be substantial. The system must be designed to determine the most useful features from the set of all features so that the final set is a reduced version containing the least amount of irrelevant/weak features.

1.2.2 AdaBoost Classifier

The more features present in a classifier the more computationally expensive the classifier is when applied to an image. To account for this issue, several feature selection procedures have been used in the past for the reduction of irrelevant features. Papageorgiou et al. proposed a technique based on feature variance [6]. Yang et al. proposed another technique using the Winnow exponential perceptron learning rule [7]. Both solutions failed to meet the requirements by P. Viola and M.J. Jones as they claim that these still retain a large number of features. Instead, the authors chose the Adaboost boosting algorithm [8] as this was shown to decrease the feature set by a considerable amount, leaving only the perceived best features. The algorithm achieves this result by initially assigning an equal weight to each training image, applies features to the image, calculates error rates and new weights, and applies these new weights to images that have been misclassified. This process is repeated until the required accuracy and error rates are achieved, or the number of required features are found. The result from the algorithm would consist of a set of weak classifiers which together form a strong classifier for object detection.

Now that the resultant feature set is drastically reduced in size from its original count, the next step is to classify the objects themselves. However, given a test set of about 1000 images, when applying the strong classifier consisting of 5000 features, the quadratic time (O(nm)) needed for classification would still be significant, given that each feature would need to be applied to each image.

1.2.3 Cascade Classifiers

To resolve this issue, P. Viola and M.J. Jones propose their novel cascade of classifiers. Instead of applying all the weak features to every image, the features are now grouped into a different set of classifiers which are applied in stages to sub-windows of the image as shown in *Figure 3*. By further using the Adaboost algorithm on the reduced feature set, simpler classifiers are created as front liners for rejecting the majority of windows in early stages. More complex classifiers are created and applied at later stages of the cascade to ensure lower false positives. If a sub-window fails at any stage of the cascade, the window is immediately rejected. Otherwise, if it passes all stages, the window is classified as a face region.

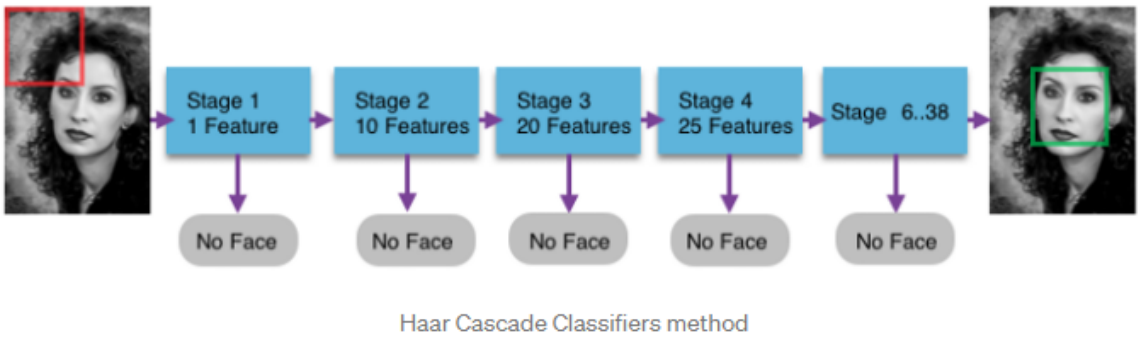


Figure 3: Cascade classifier process. [2]

The resulting Viola-Jones cascade provides a much faster alternative to applying the whole set of features while improving detection accuracy as claimed by the authors [1]. Thanks to this fast classification, this technique was and still is widely used in real-time applications.

1.3 Usefulness in real-time applications

A balance between real-time and robustness of the Viola-Jones algorithm needs to be maintained for achieving the best results. If a lot of features are present in the cascade, despite it being more accurate, the solution would take longer times to process image frames which is not ideal in real-time applications. The notion of a real-time system is

not tied to a specific frame rate, however more frames per second would result in a more continuous environment. As described by the authors, their final detector managed to process a 384 by 288 pixel image in just 0.067 seconds using a total of 6060 features spread throughout their cascade. Even with the primitive technology that was available at the time, the authors still managed to achieve 15 frames per second (FPS) which exceeds the recommended 10 FPS in [9]. Now that technology has improved substantially from the 700 Mhz Pentium processor that the authors used, the algorithm generates more frames per second and is still relevant in real-time domains especially those for face detection [10].

2 Artifact 1

In the previous section, the Viola-Jones detection method was discussed and explained in brief. This section shows the methodology adopted for developing artifact 1 of the project and the evaluation carried out to determine the best parameters for the trained classifiers.

2.1 Methodology

Firstly, since a cascade of classifiers needs to be trained on positive and negative images, data sets had to be found for both. In this case, since the classifier is required to detect faces, the positive data set had to consist of images with human faces and the negative without. The Flicker-Faces-HQ Human faces data set[1] was used for the positives, which includes 70 000 thumbnail images of size 128 by 128 pixels, each consisting of 1 human face. For the negative data set, this could have been any set of images that do not contain faces. However, in real applications these negative images should represent the general scene the classifier is to be applied on, such as empty rooms in security domains. Since this artifact is tested on images found online, a unique data set consisting of 3174 garages [2] was chosen as it contains a good mix of pictures showing garages, houses and empty rooms.

Following the OpenCV guide [3], the next step was to create file descriptors for each of the positive and negative images present. The negatives text file consisting only of local paths to the non-face images, was defined by creating a function which iterates over all images within the negatives directory, and append their path to the text file. A similar implementation was done for the positives text file, however, this file also requires the bounding rectangle coordinates which indicate where faces are within the thumbnail. OpenCV have an annotation tool which allows for manual determination of face windows by dragging the area on the image which represents a face. Since doing this manually is a very lengthy process for 70 000 images, an alternative to this had to be found. By exploiting the fact that the chosen data set consisted of 1 face per thumbnail which filled a good majority of the whole image, the bounding boxes were set to fit the whole thumbnail area.

[1] https://www.kaggle.com/greatgamedota/ffhq-face-data-set accessed on 23/11/2020
[2] https://www.kaggle.com/yeayates21/garage-detection-unofficial-ssl-challenge accessed on 23/11/2020
[3] https://docs.opencv.org/3.4/dc/d88/tutorial_traincascade.html accessed on 23/11/2020

The next step involved installing a set of pre-built libraries offered by OpenCV for providing tools to create positive samples and train the classifier. Using the OpenCV *createsamples* tool, the vector file containing positive samples (required for classifier training) was generated from the positives descriptor file. The command used to generate this file is listed below which contains parameters specifying the width and height of windows, and the number of bounding box rectangles to be considered. The chosen 24 by 24 window size was defined to resemble that used in the Viola-Jones implementation [1]. Larger windows correspond to more accuracy when detecting a face, but the training time required would also increase. Reason being, this size was chosen as a good compromise between the two factors. The number of bounding boxes was chosen to be 70 000 Since every image consisted of 1 rectangle. In situations where training images consist of more than one face, this number would be larger than the total number of positive images.

```
opencv_createsamples.exe -info positives.txt -w 24 -h 24 -num 70000 -vec
    positives.vec
```

Command to create the positives vector file

For the final step, a directory was created to store the trained classifier model. After this was done, the OpenCV *traincascade* tool was used to create and train the model. At this stage, parameters related to the number of positive and negative samples to be used in training had to be chosen. From the reviewed blogs, there is no specific ratio of positive and negative images which would always guarantee maximum results. Therefore, to evaluate what works best for this artifact, 3 classifiers were created and trained on a total of 1500 images for each. The first classifier is trained on 500 positives and 1000 negatives. The second on 1000 positives and 500 negatives and the third on 750 positives and 750 negatives. Despite the loss in accuracy and generalisation, the choice of using just 1500 images for training was done to reduce training time by a considerable amount. Apart from image parameters, the number of stages the cascade contains also had to be specified. The more stages the cascade contains, the more accurate the solution. However, a reasonable number had to be chosen to not over train the classifier. Initially, this number was chosen to be 15 stages, but since the classifier was being trained on few images, the model was observed to be over-fit. Additionally, OpenCV itself restricted further stages from being trained when a false alarm (FA) rate threshold was achieved as shown in *Figure 4*.

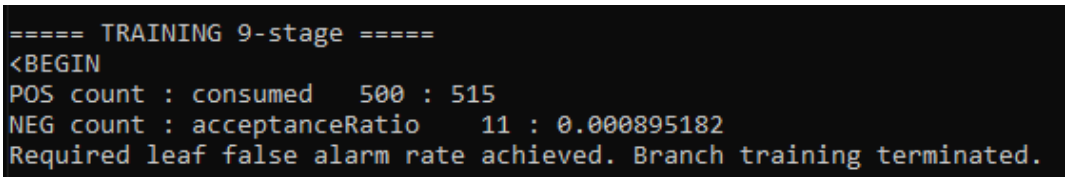


Figure 4: Terminal output showing forced cascade training termination

To address this issue 10 stages were chosen instead, although this still managed to reach the required FA rate for the classifier containing the least amount of negative images. To solve this, instead of increasing the number of images used for training, the *maxFalseAlarmRate* parameter was defined to be 0.4 instead of its default 0.5 value. By tweaking this parameter, individual stages are forced to be more complex thereby including more features per stage. This not only made the classifiers more accurate, but also allowed for the required amount of stages to be defined so that the classifiers may be fairly compared. The command format used for cascade training is shown in the below snippet.

```
opencv_traincascade.exe -data cascade_750p_750n_10s/ -vec positives.vec -bg
    negatives.txt -w 24 -h 24 -numPos 750 -numNeg 750 -numStages 10
    -maxFalseAlarmRate 0.4
```

Command to train the classifier consisting of 750 positive and negative images

As shown above, the command also consists of width and height parameters for sub-windows which is required to be equal to that used in the positive vector file. An additional *minHitRate* parameter could also be chosen which forces the model to over-fit to the training data. This rate is set to 0.995 by default and so, after determining the most efficient image ratio for the classifier, a further evaluation is carried out by creating two new classifiers using the best image ratio, with hit rates of 0.999 (higher) and 0.991 (lower). The results from each classifier can be observed in the sub section.

2.2 Evaluation

Evaluation was carried out to determine if varying the image ratio and the minimum hit rate is of any significance when detecting faces. Three test images were specifically chosen to test how capable the classifiers are in detecting faces with variation in light, rotation, environment and more. Detection results for each technique are displayed in confusion matrices following the same structure as that shown in Table 1. Using these values, the best model is determined in the end by finding the *precision*, *recall* and *F-Measure* using equations 1, 2 and 3 respectively. Precision measures the accuracy of a model in correctly detecting a face from all the detections made; Recall measures the model's ability to correctly identify faces from all the people in the test set; F-Measure is the harmonic mean between precision and recall which measures how good the model is on both and so, the best model will be determined from this value. An visual evaluation of the detections done will also be given for determining the strengths and weaknesses of each classifier.

		Predicted	
		Positive	Negative
Actual	Positive	T_p	F_p
	Negative	F_n	T_n

Table 1: Confusion Matrix structure

Since evaluation was to be carried on just 3 images, results were captured manually. An alternative approach would have been to split the initial positives file into a training set and a test set. Looping over the test set, each image with its face coordinates could then be compared with the detection results having an *Intersection Over Union* (IoU) of about 0.5, so that the metric values of each classifier could be found automatically. This wasn't done however, as manual annotation of bounding boxes was required and also, by using the visual method, unique observations may be made on the strengths and weaknesses of the different classifiers, which can't be done in an automatic evaluation scheme.

$$precision = \frac{T_p}{T_p + F_p} \tag{1}$$

$$recall = \frac{T_p}{T_p + F_n} \tag{2}$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{3}$$

2.2.1 Evaluation Methodology

Cascade Model	Positives	Negatives	Stages	Max False Alarm Rate	Min Hit Rate	Training Time
A	1000	500	10	0.4	0.995 (default)	4min 5s
B	500	1000	10	0.4	0.995 (default)	3min 50s
C	750	750	10	0.4	0.995 (default)	3min 42s
D	500	1000	10	0.4	0.999 (default)	4min 27s
E	500	1000	9	0.4	0.991 (default)	2min 33s

Table 2: Classifiers and the parameter values used

Each test image shown in Table 3 was selected purposely to assess the classifiers on different face variants. Family 1 with 4 faces, was chosen to test for face detection in bright environments. Family 2 with 5 faces, was chosen for detecting faces of different age groups ranging from toddlers to that of elders. Lastly, family 3 with 4 faces, was chosen to test if occluded faces could be detected by the cascades. 5 classifiers in total were trained as shown in Table 2. Classifiers A, B and C were all given the same parameters except for the number of positive and negative images. These are compared for determination of the best ratio of images which provided the most optimal results. Classifiers D and E use this image ratio with a varied minimum hit rate so that its effectiveness could be determined.

(a) Family 1

(b) Family 2

(c) Family 3

Table 3: Detection results for the eye and face classifiers

2.2.2 Results with varied training ratios

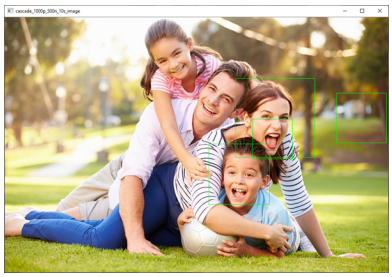
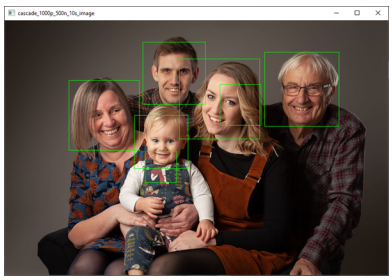
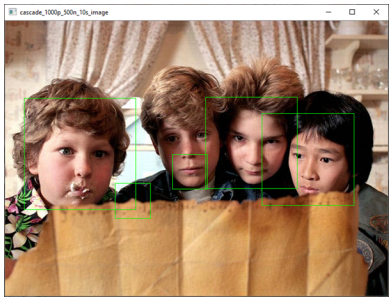
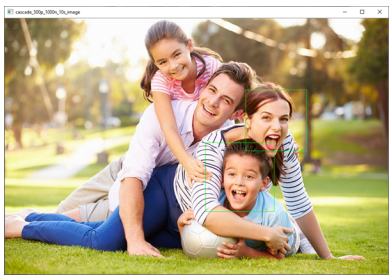
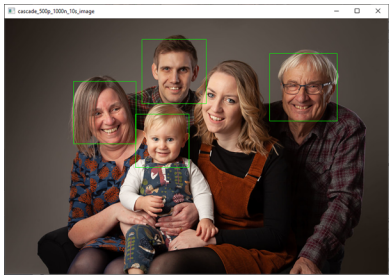
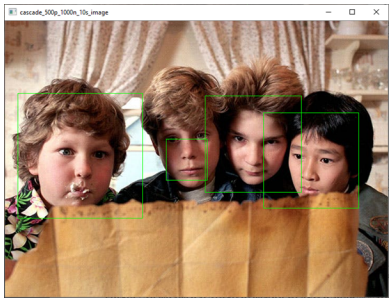
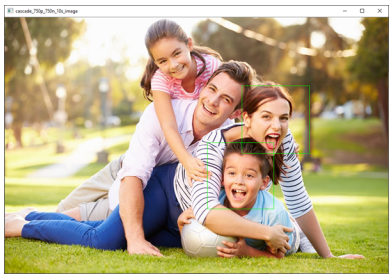
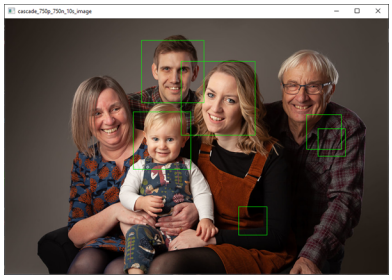
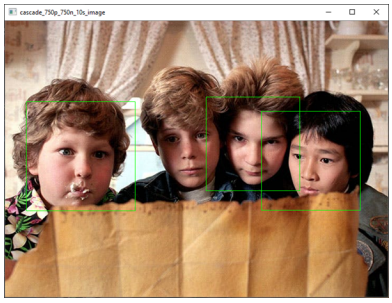
Model	Family 1	Family 2	Family 3
A			
B			
C			

Table 4: Detection results for classifiers A, B and C

Applying the models to the three test images, the results are as shown on Table 4. Observing the face detection results for the 1st family, it is noted that cascades B and C manage to detect the same faces. Model A is observed to detect a false positive (F_p) within the background environment and an additional face within a larger face bounding box. In practical applications, post-processing could be applied to bounding boxes for eliminating those which overlap others. Doing so would reduce the number of F_ps, thereby leading to greater accuracies. In each case however, the only 2 faces detected by each model were specifically upright. The male adult and female child both have varied head rotations com-

pared to those that were detected. By carrying out this test, it is confirmed that variations in head rotation do have an effect on the Viola-Jones method.

The results achieved on family 2 showed that the models were indeed capable of detecting faces of varied ages as well as individuals wearing glasses. Cascade C is shown to perform worst for this test having three F_ps. It is also confirmed that should post-processing be adopted, cascade A would have been superior to B. However, since this was not done, model B maximises precision achieving no F_ps whatsoever. The final test for family 3, shows that each classifier successfully detects all faces except for the boy in the middle. As expected, since half of the boy's face was occluded by the leftmost person's head, the small training sets used might not have had enough data to account for changes in pixel intensities like those observed. The only classifier which came close to detecting the boy however, is cascade B. Even though the defined ROI does not cover the full ground truth, this still covers most of the face than that observed for cascade A; By carrying out this test, it is confirmed that occlusion does have an effect on the Haar classifiers and that more training instances are required to account for darker regions.

	Positive	Negative
Positive	10	6
Negative	3	0

(a) Cascade A (1000p 500n)

	Positive	Negative
Positive	10	0
Negative	3	0

(b) Cascade B (500p 1000n)

	Positive	Negative
Positive	8	3
Negative	5	0

(c) Cascade C (750p 750n)

Table 5: Confusion matrices for cascades A, B and C

Model	Precision	Recall	F-Measure
A	0.625	0.769	0.690
B	1.0	0.769	0.869
C	0.727	0.615	0.666

Table 6: Metric results for cascades A, B and C

Accumulating the results of each classifier, the matrices in Table 5 were constructed. From these results, the metric values of each model were found which can be seen in Table 6. Should the precision of the model be of utmost importance to the application, model B is shown to be superior achieving the highest possible value of 1 with cascade A being the worst. In terms of recall, models A and B achieved the same result with C being the worst. When finding the harmonic mean, the superiority of cascade B is proven by achieving a value of 0.869 when the runner up achieved 0.690. From these results, it may be concluded that the ideal image ratio for training is a 1:2 ratio, whereby the number

of negative images is double that of positive images. Since cascade A was trained mostly on positive images rather than negatives, results showed that a lot of non-face areas were being detected due to the lack of negative representations. This model however, could be made comparable or even better to cascade B, should post-processing be adopted within the solution. Cascade B showed the best results all round, identifying most of the faces present with no false positives. The increased number of negative images used in training may be a cause for this as with more non-face references, the classifier could determine irrelevant features more accurately. The final classifier C showed a balance of the other two. However, this would still be the worst performing model given that post-processing is performed.

2.2.3 Results with varied hit rates

Building on these findings, cascades D and E were also defined having the same image ratio as B. Classifier D was given a higher minimum hit rate value which was 0.004 more than the default value used in the previous tests. It was observed that by modifying the hit rate, the training time is impacted considerably, the larger the rate is. This makes sense as by having a larger rate, the model is trying to over-fit to the training data more. To achieve this, more features were observed to be added in each stage. Since classifier E had a lower hit rate value which was 0.004 less than the default value, the model was observed to include less features per stage. Consequently, the training time was the lowest for all the trained models and the max number of stages that could be trained was 9 instead of 10. This was due to the model reaching the required false alarm rate as specified by the OpenCV tool.

Applying these new classifiers to the test images (Table 7), results show that the default hit rate is superior to the higher and lower rates, which managed to achieve the best precision and F-measure as shown in Table 9. Interestingly, cascade E provided the best recall, which proves that when using a less over-fit model, generalisability is increased. However, this increase severely impacts precision due to the larger number of F_ps that may be achieved. It is also interesting to note that despite being more over-fit, cascade D detected more regions than B. By carrying out this test, it is found that using a balanced minimum hit rate is likely to achieve more optimal results.

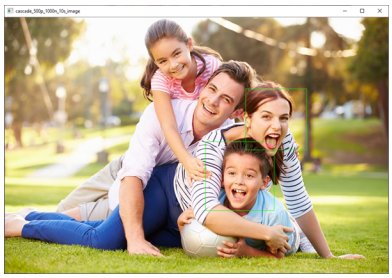
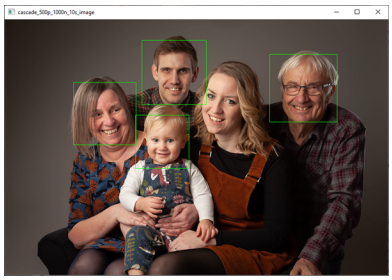
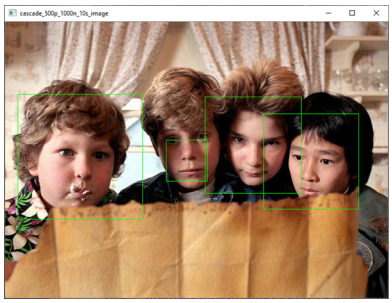
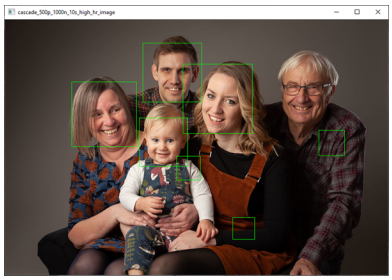
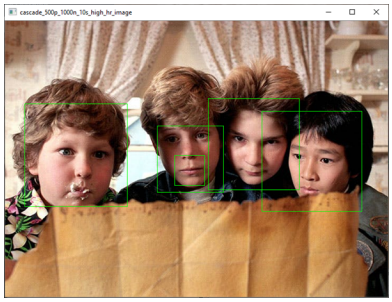
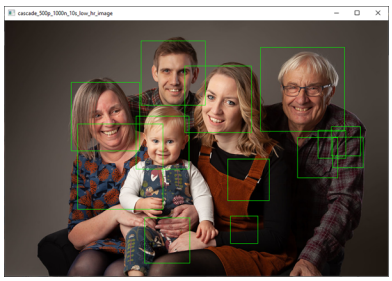
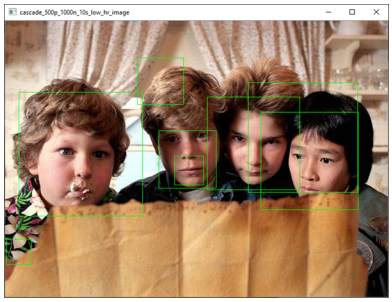
Model	Family 1	Family 2	Family 3
B			
D			
E			

Table 7: Detection results for classifiers B, D and E

	Positive	Negative
Positive	10	0
Negative	3	0

(a) Cascade B (0.995hr)

	Positive	Negative
Positive	10	5
Negative	3	0

(b) Cascade D (0.999hr)

	Positive	Negative
Positive	11	12
Negative	2	0

(c) Cascade E (0.991hr)

Table 8: Confusion matrices for cascades B, D and E

Model	Precision	Recall	F-Measure
B	1.0	0.769	0.869
D	0.667	0.769	0.714
E	0.478	0.846	0.611

Table 9: Metric results for cascades B, D and E

3 Artifact 2

In the previous section, the methodology and evaluation carried out for artifact 1 were discussed. This section illustrates the methodology adopted for developing artifact 2 of the project and the evaluation carried out to determine the efficiency of the chosen pre-trained classifier on several different variants such as different lighting conditions and head rotations.

3.1 Methodology

Pre-trained Haar cascade models were obtained from the suggested GitHub directory [4]. Using the video capture code found on the OpenCV detection tutorial [5] as reference, code for capturing screenshots from webcam was implemented for real-time video tracking. The detection function was then defined to accept the current frame as input to be analyzed by the face and eye classifiers. Since eyes have to be identified for every face, the region of interest (ROI) had to primarily be found before detecting the eyes themselves. If this weren't done, random eye bounding boxes would be identified outside of face regions which is not ideal. The above was achieved by first detecting all the faces within the frame, and for each face, the bounding box coordinates are used to draw the face rectangle in green and define the face ROI. The eye cascade is then given this ROI and outputs the eyes detected within it. A nested for loop then applies the same logic as that done for faces to draw the eye regions, only this time the colour was set to red. The modified frame is then outputted with the outlined face and eye regions.

3.2 Evaluation

Since pre-built classifiers are used, these would most probably be fine-tuned to output the best results when applied to real-time scenarios. To test this claim, the classifiers were tested on numerous different tests to identify the strengths and weaknesses of the 2 models. The following tests were carried out:

[4]https://github.com/opencv/opencv/tree/master/data/haarcascades accessed on 24/11/2020
[5]https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html accessed on 24/11/2020

- **Test 1 (T1)** : Face detection when head is rotated slightly rightward.
- **Test 2 (T2)** : Face detection when head is rotated slightly leftward.
- **Test 3 (T3)** : Face detection when head is rotated slightly upward.
- **Test 4 (T4)**: Face detection when head is rotated slightly downward.
- **Test 5 (T5)**: Face detection whilst wearing glasses.
- **Test 6 (T6)**: Upright face detection at a normal distance away from the webcam.
- **Test 7 (T7)**: Upright face detection at a larger distance away from the webcam.
- **Test 8 (T8)**: Upright face detection at a smaller distance away from the webcam.
- **Test 9 (T9)**: Face detection in bright environments.
- **Test 10 (T10)**: Face detection in dark environments.
- **Test 11 (T11)**: Face detection when eyes are closed.
- **Test 12 (T12)**: Random Detection.

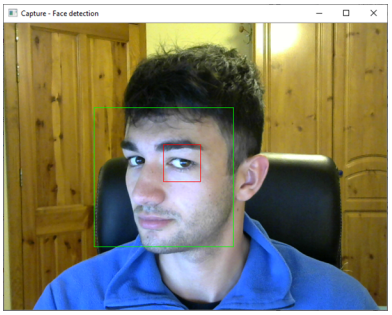
3.2.1 Test Results

To showcase the detected regions, screenshots were taken at random intervals of the video capture. It was noticed that any slight movement changed detection results considerably and so, before every test, I made sure to stand still and wait for the classifier to adjust to the new frame before taking the screenshots. The captures are all displayed in Table 10 on page 18. As a further note, whenever reference to left or right directions are being made in this discussion (such as a left eye), the reader should take this direction from his point of view.

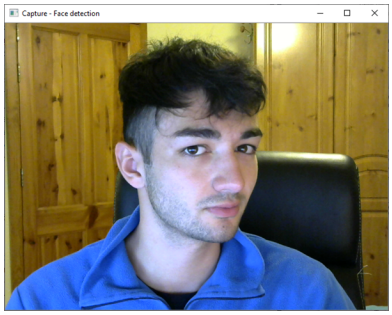
Starting with **T1** it seems as though the classifiers correctly determined the face and eye regions although only one eye was detected. If the eye classifier was trained on just open eyed front facing individuals, the model would have a harder time detecting stretched eyes like my own. Otherwise, since the head rotation in T1 makes the left eye smaller, the Haar features may not have had enough pixels to detect the left eye. Continuing with

T2, no face was primarily detected by the face classifier. This suggests that the training images used consisted of mostly faces looking in the opposite or frontal directions. As a further improvement, if such classifier were to be trained on the same data set again, the positive images could be rotated horizontally (thereby facing the right direction) to have a balance of all head rotations. Despite **T3** having less number of pixels representing the face and eyes, the classifiers still manage to detect all the required regions of interest. The same cannot by said for **T4** which just detects a single eye. In the way T4 was captured, it is surprising at how the eye classifier managed to detect an almost closed left eye but not detect that in T1. Since the eye lashes were more defined in T4 this may suggest that the model finds it easier to detect eyes with visible eye lashes. The classifiers are shown to be able to detect faces and eyes when wearing glasses as shown in **T5**. A reason for being unable to detect the right eye may most probably be related to the reflection on the glass lenses which affect the detection process. Despite this reflection being present in the left eye also, the model still manages to detect the eye as it is slightly more visible than the right.

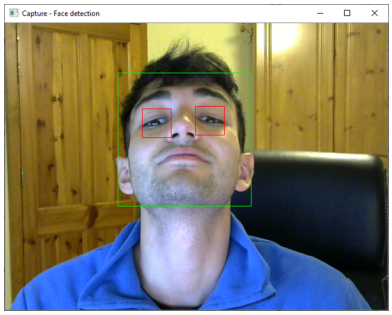
Moving on to the distance tests, **T6** and **T8** provided the same successful results despite T8 being zoomed in considerably. Surprisingly, despite **T7** being based on larger distances, an eye was still detected accurately which was not as expected. A possible reason for this is that the eye model contains line features which detects the eyebrows and eyes which would be the most visible features at larger distances. Varied illumination is also shown to have no affect on detections as observed in **T9** and **T10**. As expected in **T11** only the face is detected due to the closing of eyes. This however falsifies the statement that eyebrows have an effect on eye detection. If this were so, eyes would have probably been detected and so it can be concluded that eyebrows are just a facial feature used in face detection. Lastly, for any Star Wars fans out there, the final test **T12** carried out, sought to determine if any other objects would falsely be determined as faces and/or eyes when no faces where in the scene. Since no detections were made, this suggests that the classifiers are trained on a good amount of negative images.



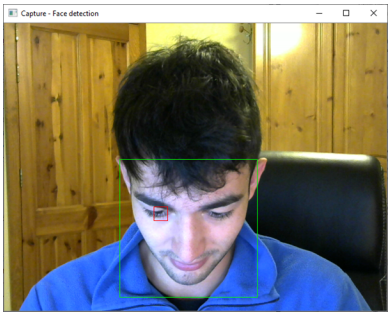
(a) T1



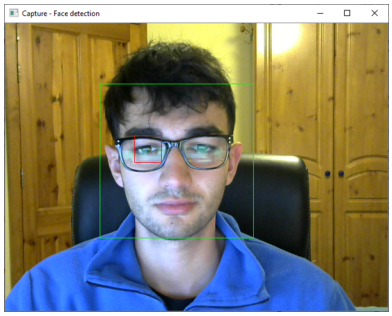
(b) T2



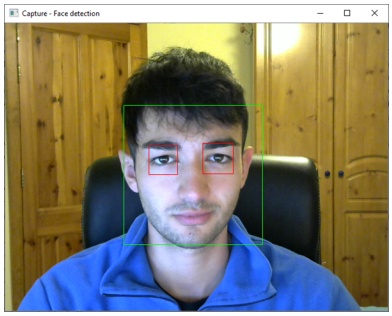
(c) T3



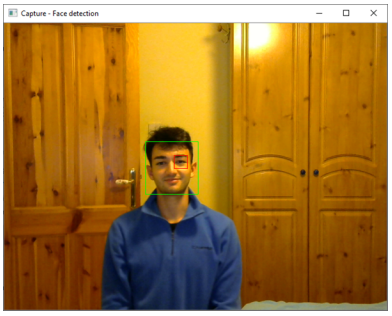
(d) T4



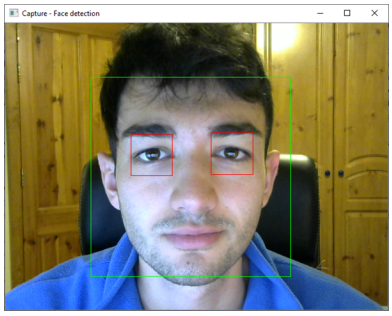
(e) T5



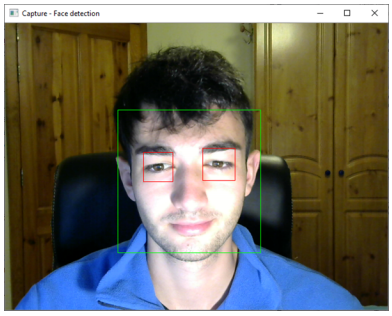
(f) T6



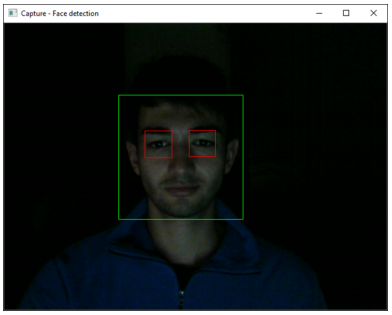
(g) T7



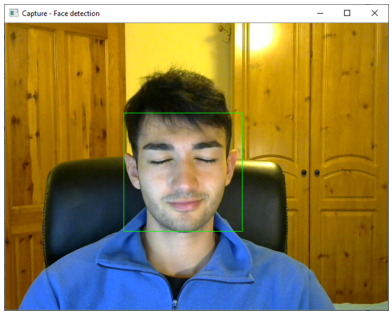
(h) T8



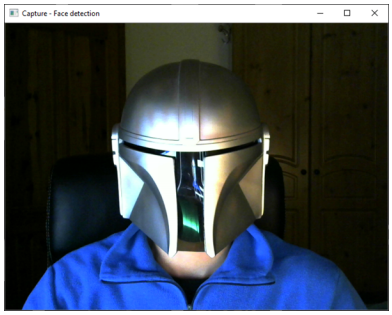
(i) T9



(j) T10



(k) T11



(l) T12

Table 10: Detection results for the eye and face classifiers

4 Comparison with alternative approaches

The previous sections illustrated the methodologies and evaluations carried out for each artifact in the project. This section provides alternative approaches to the Viola-Jones solution and compares these approaches with it.

4.1 Alternative Feature approaches

Even though Haar features provide an efficient and flexible form of object detection, these also have their drawbacks. As stated by P. Viola and M.J. Jones [1], their detector is mostly suited for frontal and upright face detection. The reason for this is because the only orientations available within the rectangular features are vertical, horizontal and diagonal features. The authors also notice how their detector usually fails on occluded faces especially when the eyes are occluded.

4.1.1 Local Binary Patterns

An alternative to Haar features, is the Local Binary Pattern (LBP) which yields integer precision of pixel values instead of the floating point precision offered by Haar features [11]. By extracting LBP histograms from the image sub-windows, these would then be concatenated together to form the whole feature histogram representing the image. By using this method of integer precision, LBP is shown to be faster than Haar which provides similar detection quality within a percentage of the training time. To top it off, S.Liao et al. suggest an improvement to standard LBPs through their proposed Multi-scale Block Local Binary Pattern (MB-LBP) [12]. MB-LBP is another alternative to Haar which is not only more robust than its LBP predecessor, but also makes use of the integral image technique for efficient computation.

4.1.2 Histograms of Oriented Gradients

Another alternative proposed by N. Dalal and B. Triggs [13] uses Histograms of Oriented Gradients (HOGs) which is evaluated to achieve better results for pedestrian detection,

than the best Haar wavelet based detector at that time. This is achieved through the use of a Support Vector Machine (SVM) based window classifier with a dense grid of HOG descriptors. Instead of considering pixel intensities like the Viola-Jones method does, the technique counts the occurrences of gradient vectors representing light directions for localising image segments. The benefits to HOGs over Haar features is mostly related to their robustness in face detection even when the human is wearing glasses or has a bit of head rotation as shown in *Figure 5*. Although, to achieve this increase in precision, the whole detection process is slowed down when compared to the more practical Viola-Jones detector.

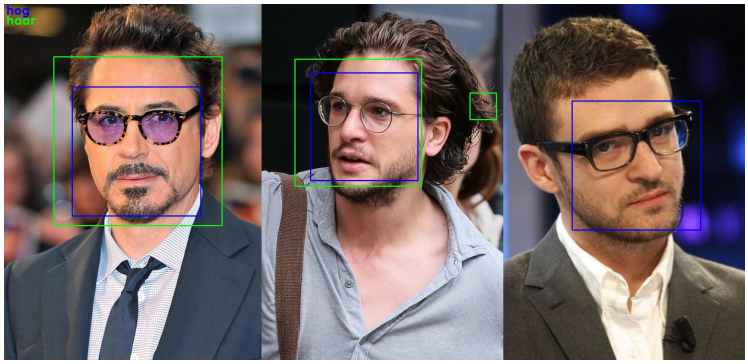


Figure 5: Output of HOG and Haar detectors. [3]

4.1.3 Comparison of features

It is difficult to determine which feature descriptors to use from those mentioned above, as each has their own pros and cons. In a recent study however [14], HOG, LBP and Haar-Like features are compared for on-road vehicle detection. Results showed that the LBP feature descriptors offer the best detection rates when compared to the other methodologies, with HOGs being the worst and Haar being a good medium between LBP and HOG. K. Kadir et al. [15] confirm that LBP features are more efficient and reliable than Haar-like features for face detection systems. In another study however [16], R.A. Asmara et al. evaluate that Haar features are better than both HOG and LBP for vehicle detection, also having the fastest execution time. It seems as though the choice of which features to choose is between Haar and LBP, with some studies showing that LBP is the better descriptor whilst others show otherwise.

4.2 Deep Learning approaches

As described in Section 1, Haar features resemble kernels used within deep learning algorithms, more specifically, Convolutional Neural Networks (CNNs). Therefore, an obvious alternative to the Violja-Jones solution would be one making use of CNNs.

4.2.1 Cascaded Convolutional Networks

Since there is a certain limit to the types of features Haar descriptors could detect, a number of visual variations existing in real-world domains such as those of lighting and orientation, affect detection accuracies. As stated by Viola and Jones, *"harsh backlighting in which the faces are very dark while the background is relatively light sometimes causes failures"* [1]. CNN models resolve this issue by using trained kernels as opposed to manually determining features from pixel intensities. To maintain a computationally efficient environment, CNNs are also shown to adopt a cascade methodology for quickly rejecting background regions of an image [17]. In their solution, Li et al. propose a CNN cascade architecture which uniquely uses a CNN-based calibration stage to adjust detection windows after every stage within the cascade. K. Zhang et al claim that the solution by Li et al. requires unnecessary computational expense and ignores correlations between facial landmarks and bounding boxes [18]. As a solution to this, the authors propose their Multi-task Cascaded Convolutional Network (MTCNN) which achieves superior results over all CNN approaches at that time. The novelty in their solution lies with the fact that it consists of 3 CNNs (P-Net, R-Net, and O-Net) connected in a cascade. P-Net is a fast network used to produce candidate windows, which are later refined using their R-Net refinement network. Lastly, their output network O-Net produces the final bounding box with facial landmarks such as eye and nose positions, from the refined image. In a more recent study, Yang et al. [19] continue improving the cascade CNN architecture by taking inspiration from solutions using cascade CNN and MTCNN. Results show that the newly presented cascade CNN model is superior to many other methods of face detection, including that of Viola-Jones.

5 Alternative Application

Even though Viola-Jones object detection was mostly intended for face detection, studies have also tested its use in many other domains. D.C. Lee applies the method to a car detection domain [20] as he believes that rear views of cars have distinctive features similar to those on faces. His belief is shown to be true as results show 96% accuracy on the 80 test images used. In another study [21], Moghimi et al. apply the Viola-Jones technique to detect frontal car regions in different lighting environments. The study uniquely determines how different lit scenes show comparable results with an average of 94% accuracy in all lighting conditions. To account for the loss of accuracy from varied car rotations [22], Xu et al. propose a novel solution for their top-down car detection method using imagery from unmanned aerial vehicles (UAV). In their solution, a road orientation adjustment method is proposed, which rotates the UAV images so that roads and vehicles are aligned horizontally. By restricting the rotation which cars could have, loss in detection is reduced when applying the standard Viola-Jones detector. Although this method is shown to achieve promising performance, the authors still highlight difficulties that their method has when trying to detect turning and illuminated cars.

6 Conclusion

In this report, the Viola-Jones Object Detection Framework proposed by Paul Viola and Michael Jones back in 2001, was discussed and tested. Despite its age, by combining concepts of Haar-like Features, Integral Images, the AdaBoost Algorithm, and the Cascade Classifier, the method still provides sufficient results and accurate detections suitable for real-time applications. The evaluation carried out within the second artifact, confirms this, as results show valid detections even in dark environments. The flaw in using Haar features was also observed in this evaluation, as the classifiers struggled in detecting all features when the head was rotated off-center. In the first artifact, an evaluation to determine the best positive and negative image ratio for training was carried out. The variation of minimum hit rate parameters was also evaluated to determine if these were of any significance. Results show that the classifier having double the number of negative images than positives, performed the best out of all the classifiers. Additionally, the best results were achieved when using the default hit rate of 0.995 instead of higher or lower amounts. As a future improvement, the trained classifiers should have been given a larger set of training images for richer feature generation. Additionally, the bounding boxes specifying the face locations in the positive images should be manually annotated for enhanced precision. The evaluation step could also be extended by including more test images consisting of coloured individuals and other variations. Following these, a review was also carried on alternative approaches which make use of different features instead of Haar-like. The state-of-the-art in deep learning object detection techniques relating to cascaded CNNs, were also mentioned and reviewed. Lastly, studies using the Viola-Jones method for car detection were also described, showing the technique's ability to be generalised to other domains.

References

[1] P. Viola and M. J. Jones, "Robust Real-Time Face Detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004. Place: Boston Publisher: Springer Science and Business Media LLC, Kluwer Academic Publishers.

[2] D. Adakane, "What are Haar Features used in Face Detection ?," Nov. 2019.

[3] l. phy, "Methods for face detection and face recognition - A review.," Oct. 2018.

[4] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," vol. 1, pp. I–I, IEEE, 2001. ISSN: 1063-6919.

[5] Lepik and H. Hein, *Haar Wavelets: With Applications*. Mathematical Engineering, Springer International Publishing, 2014.

[6] C. Papageorgiou, M. Oren, and T. Poggio, "General framework for object detection," vol. 6:, pp. 555–562, Feb. 1998.

[7] M.-H. Yang, D. Roth, and N. Ahuja, "A SNoW-based face detector," pp. 862–868, Jan. 1999.

[8] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, pp. 119–139, Aug. 1997.

[9] "Real-Time Image Processing," in *Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems* (T. Bräunl, ed.), pp. 297–315, Berlin, Heidelberg: Springer, 2008.

[10] R. R. Damanik, D. Sitanggang, H. Pasaribu, H. Siagian, and F. Gulo, "An application of viola jones method for face recognition for absence process efficiency," *Journal of Physics: Conference Series*, vol. 1007, p. 012013, Apr. 2018.

[11] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face Recognition with Local Binary Patterns," in *Computer Vision - ECCV 2004* (T. Pajdla and J. Matas, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 469–481, Springer, 2004.

[12] S. Liao, X. Zhu, Z. Lei, L. Zhang, and S. Z. Li, "Learning Multi-scale Block Local Binary Patterns for Face Recognition," in *Advances in Biometrics* (S.-W. Lee and

S. Z. Li, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 828–837, Springer, 2007.

[13] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, June 2005. ISSN: 1063-6919.

[14] A. Arunmozhi and J. Park, "Comparison of HOG, LBP and Haar-Like Features for On-Road Vehicle Detection," in *2018 IEEE International Conference on Electro/Information Technology (EIT)*, pp. 0362–0367, May 2018. ISSN: 2154-0373.

[15] K. Kadir, M. Kamaruddin, H. Nasir, S. Safie, and Z. Bakti, "A comparative study between LBP and Haar-like features for Face Detection using OpenCV," pp. 335–339, Aug. 2014.

[16] R. A. A. e. Al, "Comparison of Vehicle Detection Using Haar-like Feature, LBP and HOG Technique for Feature Extraction in Cascade Classifier," *International Journal of Advanced Science and Technology*, pp. 834–838, Oct. 2019.

[17] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A convolutional neural network cascade for face detection," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5325–5334, June 2015. ISSN: 1063-6919.

[18] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," *IEEE Signal Processing Letters*, vol. 23, pp. 1499–1503, Oct. 2016. Conference Name: IEEE Signal Processing Letters.

[19] W. Yang, L. Zhou, T. Li, and H. Wang, "A Face Detection Method Based on Cascade Convolutional Neural Network," *Multimedia tools and applications*, vol. 78, no. 17, pp. 24373–24390, 2019. Place: New York Publisher: Springer Science and Business Media LLC, Springer US.

[20] D. Lee, "Boosted Classifier for Car Detection," Jan. 2007.

[21] M. Moghimi, M. Nayeri, M. Pourahmadi, and M. K. Moghimi, "Moving Vehicle Detection Using AdaBoost and Haar-Like Feature in Surveillance Videos," *International Journal of Imaging and Robotics*, Jan. 2018.

[22] Y. Xu, G. Yu, X. Wu, Y. Wang, and Y. Ma, "An Enhanced Viola-Jones Vehicle Detection Method From Unmanned Aerial Vehicles Imagery," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, pp. 1845–1856, July 2017. Conference Name: IEEE Transactions on Intelligent Transportation Systems.