Empirical comparison between the classic Viola-Jones face detection algorithm and a CNN-based one

Guo Qin Li, Kai Wang, Yingchao Shan (Group 34)
{gqli, k282wang, y28shan}@uwaterloo.ca
University of Waterloo
Waterloo, ON, Canada

Introduction

One of our group members, Brian, was trying to take a selfie with his African manager at the end of co-op and realized the face detection box would not appear for his manager unless they went outside. We had no idea what was happening so we watched some Computerphile videos about face detection and learned that the algorithm is actually very old, but good enough. There are now newer algorithms based on convolutional neural networks. This short story inspires us to investigate the principle of face detection technology in depth.

In 2017, MIT Technology Review ranked 'Paying with your face' as one of the ten breakthrough technologies of that year, the review shows that China has already allowed face-detection systems to make payments. Today, face detection has become a representative application of artificial intelligence. According to Enriquez (2018), "face detection is defined as a computer technology which can recognize a human's face in digital pictures". Because of this technology, appearance has become people's new identity. Apple released Face ID, a face detection system, together with its new devices in 2017. It is used as an authentication method for unlocking devices and making payments. Not only for security, face detection is also used for social media, Facebook can recognize people's faces in posted photos and automatically tag their friends; Snapchat users can apply fancy features onto their faces in both pictures and videos. Nowadays, face detection has become an important technology that is widely used in people's daily life.

For the scope of this project, we will focus on face detection, rather than more fancy facial recognition capabilities. The difference is that detection answers the question "Is there a face in this picture? Yes/No and how many?" rather than "Is this the same person? Is the person smiling?". This is important as detection of the presence of a face needs to happen first before any complex recognition algorithm is run.

Unfortunately, the two major algorithms that we used to implement face detection still have defects. Viola-Jones algorithm is a classic algorithm that people used to implement face detection. It is proven that Viola-Jones is competitively accurate and efficient (Ephraim, Himmelman, and Siddiqi 2009). However, it is only capable to the frontal face without covering. For instance, side or heavily angled profiles will both cause Viola-Jones to fail. Fortunately, the newer Convolutional Neural Networks (CNNs) can be a perfect complement. Farfade et al. (2015) suggested that CNNs is both fast and applicable for different angles of photos. However, CNN has its own drawback. In some situations, CNNs is not trusted and will find more false positive cases, and this is not tolerable for security purposes. Moreover, deep learning in general uses a lot more computing resources compared to using classifiers (Enriquez 2018). Some research has already been completed to find the capability of both algorithms, so our primary goal is to compare the performance between CNNs and the classic Viola-Jones algorithm on faces with some additional features such as glasses.

To make our comparison more accurate, we will also try to implement some basic preprocessing on images in an attempt to improve results where needed (discussed later). In the end of Enriquez's paper (2018), he suggested a new method that using Viola-Jones' Haar filter instead of Gabor Filter for CNNs algorithm, this will make the CNNs not only more efficient, but also as accurate as Viola-Jones. If time permits, we will also try to implement Enriquez's idea.

If our project is successful, we will have a solid understanding of the difference between Viola-Jones and CNNs in most real-life cases. If we manage to implement Enriques's idea, we will have a new face detection method that is superior to both existing algorithms.

Contributions

The research question is to compare the performance of the old Viola-Jones face detection algorithm to a newer deep learning based one called MTCNN. A comparison will be made over four datasets of various face placements and types, with the fourth one containing no human faces at all (test of false positives). Notes of detection accuracy and runtime are made, to help understand why Viola-Jones still seems to be used in places, including low-hardware devices. Next, a simple preprocessing step is examined to see if we can lower false positives by blurring when grid squares of the image do not have human skin tones. This simplified preprocessing should be usable on low-end hardware, as it is a simplified angle of various work such as one done by

(El Maghraby et al. 2013). As for results, we originally expected this to prevent Haar features from being detected in false positives dataset while only seldomly preventing true positives from being detected. We also expected a deep learning MTCNN to take much more resources than the simple Haar classifiers of VJ. After the implementation, we can say the following:

- Confirmed the current understandings: Viola-Jones is much less resource- intensive compared to MTCNN. For example, one dataset took 3.85 times longer to run on MTCNN than on VJ with the same hardware. It makes sense that VJ is still widely used for its performance and simplicity, especially on low-end hardware. Also confirmed that VJ is much less accurate when dealing with side and mixed profile images given the widely available classifiers. In general, detection accuracy rates are significantly lower with VJ, as much as around 53.6% given no preprocessing.
- A simplified skin tone definition range in YCrCb space, with simple linear constraints, used in preprocessing was able to filter skin reasonably well. Compared to some definitions proposed in other papers (Preprocess 2), we had a more narrow definition (Preprocess 1) which missed some skin, but sometimes allowed for non-skin items (ex. Artifacts of writing on a wall) to be not considered as skin in the preprocessing step. A more narrow skin tone definition generally led to slightly better true-positive detection rates compared to a less restrictive one
- Blurring or sharpening the image based on skin tone presence (preprocess) adversely affected true-positive detection rates. Fortunately, false positives did go down a bit as originally intended when using MTCNN with preprocessing. A post-process rejection of the frame would probably have been a better filter.
- MTCNN is much more robust to image distortion and various situations of faces. The accuracy was not affected as much by the preprocessing compared to VJ. In fact, MTCNN was able to keep detecting side profiles of mugshots really well despite any blurring or sharpening that might have happened in preprocessing.

Related Work

Face detection and recognition has a long history that goes back to the beginning of computer vision. It is a subset of the more general object detection problem and the algorithms fall into two broad categories: simpler machine learning with classifiers vs. modern deep learning ones. As early as 1964, a team of researchers: Woody Bledsoe, Charles Bisson, and Helen Chan Wolf, worked on the first computer assisted facial recognition technology on a given set of mugshots (Bichwe, Bichwe, and Satija 2016). However, much of this research was motivated and done for intelligence agencies during the cold war, so most of the details were never publicly released. More importantly, early research focused on the recognition part rather than detection since the datasets they examined are already assumed to have faces in them, as expected in mugshots. This did give information about the characteristics of faces and motivation for future researchers.

Throughout the next decades, digital cameras came into use through the 1990s. One feature that manufacturers obviously wanted was to indicate faces on the LCD screen. This gave great motivation for facial detection implementations. One of the earlier works predating Viola-Jones is the work of Rowley, et al(1998) where they used a two step facial detection process: First, passing (trawling) small frames of various sizes over an image with a neural network to determine if there is a face. This is followed by a merge of the results. For its time, it was pretty successful at detecting faces in a reasonable amount of time for the given hardware and resolution (90.5% over 130 images of 320x240 pixel size on a 200 MHz cpu). To reduce the complexity of training the model, they did some preprocessing to use grayscale images. As for training, images were manually marked for facial features, centered, and sized appropriately for the window trawlers. The NN was also fed negative inputs of images that contained no faces to assist in training of negatives. Each window would return success or failure depending if there was a face there. A merge would occur afterwards where overlapping clusters of positive detections would result in a face if it exceeds their threshold measurement value. This is referred to as "thresholding" in their paper.

Within a few years, in 2001, a landmark paper by Viola and Jones for object detection (motivated by faces) was published. This paper described a very effective and efficient algorithm compared to what came before. Their method uses fast image processing techniques of subtraction on an integral (preprocessed) image to determine if there are any Haar-like facial features present. This measures the contrast difference between regions of the images in order to see if the pattern matches what one would expect to be on a face. It is easier to show what this means with an image, see Figure 1. Similar to the work of Rowley et al. (1998), a trawler or small boxes measuring 24x24 were used to scan the image for these Haar features. Also similar is that the algorithm operates on converted grayscale images rather than colour ones. It is interesting to note that these Haar features are not classified by the algorithm per say, so we are not directly asking "is there a face, nose, and mouth?" but indirectly so through contrast patterns.

Viola-Jones (2001) differs from previous work in three ways. One is the preprocessing. It is not just grayscale conversion, something called an "integral image" is calculated. This preprocessing allows for image subtraction and detection of Haar features in constant time. Another important difference, is the training they use which is based on Adaboost, which gives more efficient classifying systems for the clusters of Haar like features. Thirdly, they use a "fail-fast" approach with their classification. That is, the algorithm optimizes for rejection (surprisingly very successfully and accurately) of images that do not seem to contain faces as soon as unexpected Haar patterns. The DFA-like structure in Figure 2 shows the potential for quick early rejection of images.

Viola-Jones (2001) was so successful in that it allowed for real time detection at 15 frames/second on a 700 MHz Pen-

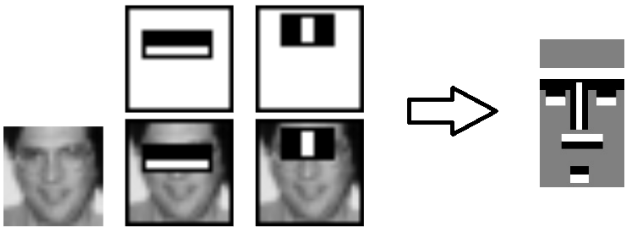


Figure 1: Haar features of contrasting regions. Left image was taken from page 4 of the original Viola-Jones' paper (2001) and the right image was taken from this blog post (Arubas, 2013): `http://eyalarubas.com/face-detection-and-recognition.html`

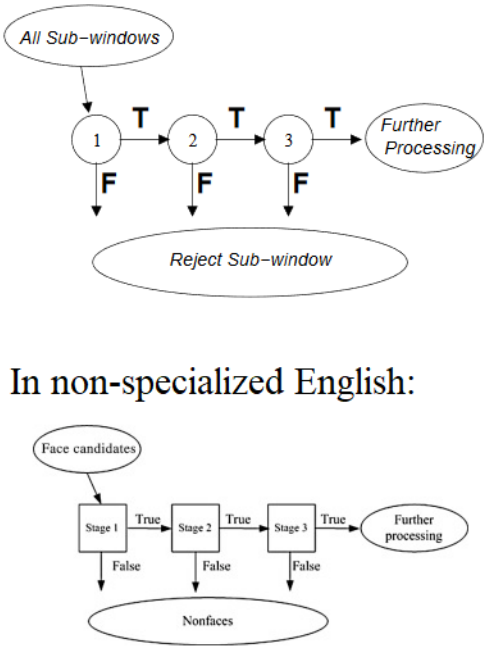


Figure 2: Early rejection cascade. Left image taken from original Viola-Jones paper (2001) from page 5. Second image was taken from (n.a., n.d.) `https://www.sciencedirect.com/topics/computer-science/face-detection`

tium processor, which was unheard of at its time and emphasized in the paper. Hence, it makes intuitive sense that many low-powered cameras began using this algorithm. As noted in the Introduction, vanilla Viola-Jones works for straight faces. Improvements were obviously needed so libraries such as OpenCV have implemented some improvements to support angled and tilted faces up to 45 degrees. This was made possible with the work of Lienhart and Maydt (2002), which gave a mathematical definition for rotated Haar like features.

Many improvements to Viola-Jones were explored throughout the years including simple image pre-processing techniques. For instance, Afifi et al. (2017) have empirically explored two blind pre-processing methods: "photometric normalization" (ex. single-scale retinex (SSR)) which is adjusting the image based on detected lighting sources, and using deblurring methods when the image is thought to be blurry. It was shown the deblurring was largely unsuccessful at improving the detection rate, but various lighting source adjustments were somewhat good. One of our approaches for this project will be from an even simpler angle - see if an overall contrast, brightness, and saturation preprocessing changes the detection rates for the better.

By the latter 2010s, processing power of modern hardware allowed for convolutional neural networks (CNN) and deep learning to be a hot topic. The hope was that it changes everything, including face detection. The improvement here is to support a huge range of poses and lighting conditions that the older algorithms struggled with. A form of NN called the "Multi-Task Cascaded Convolutional Neural Network" (MTCNN) was proposed in 2016 by Zhang et al. in their paper "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks". The difference with deep learning is that features are learned rather than more clearly defined, and the steps in the process begin to blur. This uses up a lot of computing resources. Unlike the trawler that slides a box across the image, a CNN is able to pick out boxes that it has learned to be interest areas. This is done with processing the image through layers of filters which are mathematical functions that are designed to pick up the interest areas. Convolution then figures out how much overlap is there between layers in order to pick up the pattern. Hence with more layers, the more complex patterns it can detect. In essence, there are three main steps for MTCNN, the first pass through a shallow CNN (Proposal network, P-Net) which will give lots of results that resemble faces. The second pass is through a Regression network (R-Net) which is more complex that is to immediately reject candidates that are not faces. The third pass is through output (O-Net) that does further refining to give a confident final box. Figure 3 shows these regression steps. A Medium post by (Dwiyantoro, 2018) gives a further summary. One final note is that MTCNN is capable of also aligning the face based on facial features, unlike the vanilla implementation of previous algorithms.

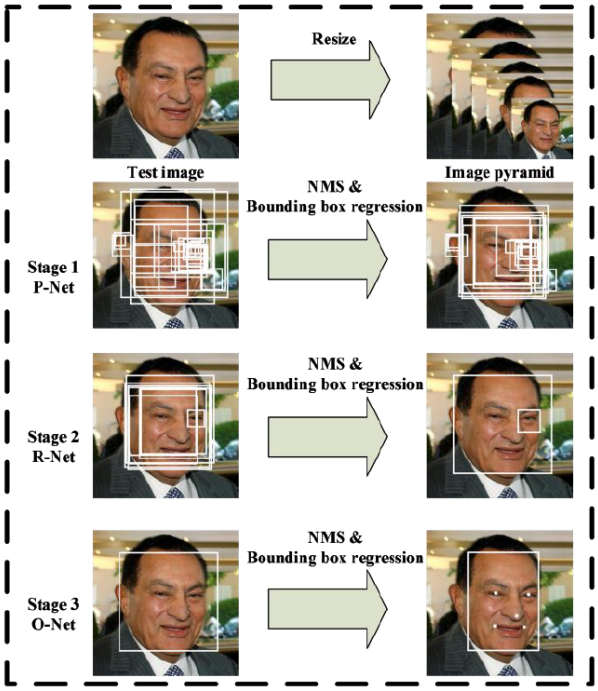


Figure 3: MTCNN Pipeline. We will compare the performance of this with the classical algorithms. Picture taken directly from page 2 of paper by Zhang et al. (2016)

Methodology

To Revised Milestone 1 Marker: the revised parts are underlined, most of them are to fix grammar and spelling mistakes. Blue text is the structure change.

Our priority is to compare the performance and accuracy of a pre-trained Viola-Jones implementation with that of a pre-trained MTCNN implementation on the same dataset. Training from scratch seems to use significant computing resources (especially for CNN), so it is unlikely that we will train large datasets ourselves. We will firstly write a framework/harness code to run both Viola-Jones and MTCNN face detectors for the dataset. OpenCV has libraries for Viola-Jones. Many existing implementations for MTCNN can be found on Github such as this one.

In order to compare the performance between the two algorithms in different situations, we decide to have four cases in total:

1. Compare the detection rates for covered faces of a single person, including hats, glasses, bangs and wavy hair.
2. Compare the detection rates for side faces of a single person.
3. Compare the detection rates for multiple faces for all angles
4. Compare the false-positive rates for disturbance pictures and random pictures

These four cases cover most real-life cases and will help us to determine their strengths and weakness. There are plenty of face datasets available, and we will use different data sets for the four cases(see Evaluation Method section). Our own code will feed these libraries the datasets while measuring the time it takes to complete. We will probably have to figure out how to automate the piping of data if we need to do checkpoints/comparisons.

Our second priority is to see if we can do preprocessing on images from a simplified angle a bit different than Afifi et al. ((Afifi et al. 2018)). That is, we will be making a simplified adjustment of images based on metrics such as overall brightness, contrast, and saturation and seeing empirical results. Then we re-run all the four cases above. By comparing the changes in detection rates and false- positive rates, we will find out how our preprocessing affect the performance of algorithms and which preprocessing technique can improve the performance.

Thirdly, and time-permitting (emphasis), we might see if we can combine CNNs with Viola-Jones as described in this quote from the Enriquez (2018) paper (much more reading is still needed) and make our own trained model:

> Like Sarwar et al., I propose using a filter to reduce the amount of computational memory needed for CNN training but, instead of using a Gabor Filter, I would use Viola-Jones' Haar like features. In doing so, the CNN would run for less time and would also have the trusted accuracy that comes with using the Viola-Jones algorithm (Enriquez 2018).

Evaluation Method

There are many available data sets online so that we will take advantage of this. To compare the performance between CNNs and the classic Viola-Jones algorithm on face detection, we are planning to use these separate datasets for the four test cases ,as we mentioned before:

1. Detection rate for covered faces of a single person, including hat, glasses, bangs and wavy hair. Dataset from Mmlab provides a capable data set for these situations.
2. Detection rate for side faces of a single person, *Nist.gov* is a national mugshot database that contains a mixture of both frontal and side face (known as "profile" shots) for 1333 subjects. The side shoots can be distinguished via the filename, and each file contains a corresponding text file of metadata (such as gender) for the subject. All images are of 8-bit greyscale, and they come from scans of police documents rather than the raw image itself. This does not affect the appropriateness of the dataset because Viola-Jones' classifiers were trained and do detection on greyscale images. Unfortunately, unlike some others, this dataset does not contain the expected detection box coordinates. However, this should not be a huge problem since we know that there is going to be one face per image (no exceptions - they are all mugshots of one person) and the size of the detection box should be close to the size of the image (i.e. face takes up most of the image). Hence, we feel this dataset is appropriate and good enough for our purposes in this category of features. Alternatives include the dataset from Areeweb and the datebase from MIT. They also have some side face photos, but we need

to select them manually from the dataset. Moreover, the set from MIT is very small containing 10 subjects.

3. Compare the detection rates for multiple faces for all angles. *kaggle.com* has many pictures with real-life scenes, such as meetings, concerts and sports games.
4. Compare the false-positive rates for disturbance pictures and random pictures People can imagine cloud as animals and people's faces, machines also have this kind of perception. *ssbkyn.com* provides a set of cloud pictures that looks like people' s faces. By adding these 'cloud faces' as disturbance, the accuracy we calculate from the program will be more convincing.

Figure 4: Cloud that is recognized as a face by computers, picture from `http://ssbkyh.com/works/cloud_face/`

For this project, we focus on more "Are there any faces in this picture?", "how many faces are there in the images", so our evaluation criteria will be whether the algorithm can find faces in the dataset, how close the number of faces found compared to the correct answer given by the dataset. In addition to accuracy, efficiency is another aspect we are interested in. Given a data set, how much time and memory will the algorithm require to complete detection? Generally speaking, we are evaluating the two algorithms by their accuracy and efficiency. The following is our brief timeline before Milestone 2.

- By June 30, we should have a running implementation for Viola-Jones and CNN, also our own program that can feed pictures to both algorithms with the features we need.
- By July 7, we should complete Test Case 1-4 without a preprocessing method.
- By July 14, we should create our own preprocessing method, and complete Test Case 1-4 again with our preprocessing method.
- After July 14, we will investigate the feasibility of implementing Enriquez's idea, no guarantee of progress can be made here, but we will try to at least include our discovery in our final report.

Algorithm

As per our research question, we will examine two face detection algorithms: Viola-Jones and modern CNN-based ones. As previously explained, OpenCV has an open-source implementation of Viola-Jones, and an open source implementation of the MTCNN specification is available on this Github repository. Descriptions of the algorithms (theory-wise) were discussed in the Related Works section. Implementation-specific details are discussed as follows.

Viola-Jones Implemented in OpenCV We plan on using this existing implementation within the well-known library. They provide many pre-trained weighted cascade trees to choose from in the form of XML files. Here is an example. Most uses for face-detection tend to use the default "haarcascade_frontalface_default.xml", but there are others of interest to us such as "haarcascade_frontalface_alt.xml". We plan on trying them all and take the best result (the one closest to the expected result). This is so that we can see the hypothetical "best case" scenario - that is, given a training set and the test set happened to have similar features that match very well, we want to know if there are still incorrect or missed detections with Viola-Jones. This implementation also returns the box size and location of the detected faces. Moreover, there are convenience methods that make it easy to change parameters to the detector, such as limiting the maximum size of the detected face. Finally, there is significant documentation for this well-supported library in case we get stuck getting it to run. Hence, we feel that this implementation is robust and versatile enough to use for this research project.

MTCNN (Deep Learning CNN-based Face Detection) There are a few implementations of this found on Github. One popular one implemented for TensorFlow as mentioned is this Github repository. Unfortunately, this implementation only gives one pre-trained model option, as found in the "mtcnn_weights.npy" file (there is only one set of weights given). This is not a huge problem for us because we do not plan on running multiple models on the same datasets with MTCNN. This is because it is quite resource intensive to even run one CNN-based detector on a dataset containing many large images. As expected with any implementation, there are enough helper methods to retrieve the results, including the coordinate location of the detected face.

Obviously, we need to do our own additional implementation of getting a test harness which will run both algorithms. When implementing this test harness, we need to carefully go over each of the readme of the dataset to determine the serialization structure of the metadata. For example, the mugshot dataset luckily has filenames with "_R" for side images with side view. A simple regex filter should pull out all the side shots. Repeat for the structure of the other datasets. We will have to do some environment scripting to determine some other basic metrics such as the time to completion for each dataset. This can

be done with mostly BASH shell scripts.

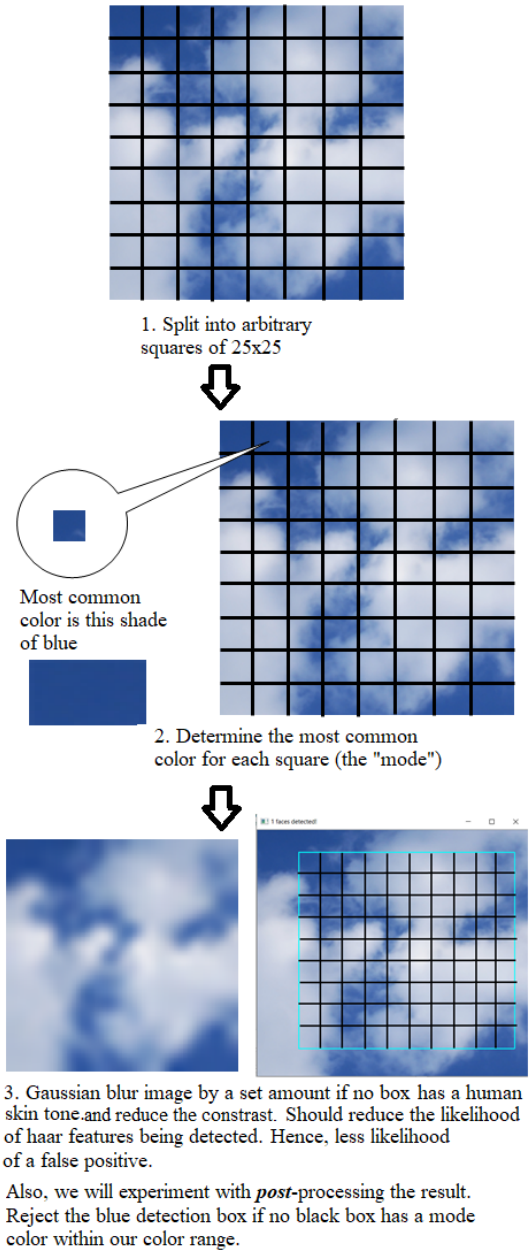


Figure 5: Preprocessing, picture from `http://ssbkyh.com/works/cloud_face/`

More importantly, we need to implement our simple preprocessing strategies of adjusting the image sharpness and brightness before feeding it into one of the algorithms. One of these strategies is to reduce the sharpness of the picture before the images are fed into one of the detection algorithms to encourage the fast-rejection of the image when skin tones are not detected. Refer to figure 5.

Step 1. Divide the image into 25x25 tiles (edges can be smaller tiles). This size was chosen by intuition as it seemed enough information in a block from the images in our dataset.

Step 2. Determine if the dominant colour in that block is a skin tone. Dominant colour was envisioned as the centroid of a 3-clustered YCrCb histogram of the image in M1. However, during implementation, we further simplified to a simple count of most frequent YCrCb pixels due to speed limitations.

Our proposed definition of Skin Tone In YCrCb spectrum, all three inequalities hold:

$$60 \leq Y \leq 255 \tag{1}$$

$$145 \leq Cr \leq 180 \tag{2}$$

$$65 \leq Cb \leq 115 \tag{3}$$

This is actually a deeply explored area so we will draw inspiration from given skin tone range equations from (Kukharev and Nowosielski 2004). These equations are done in YCbCr color space (instead of easily visualized RGB or HSV) because "they allow users to intuitively specify the boundary of skin color by hue and saturation" (El Maghraby et al. 2013). These simple constraints are based on intuition and are simplified from other papers because they form a simple range box of colors that do not involve complex systems of equations which makes it easier to implement and they are of different values. Refer to Figure 6 for a visualization of the YCrCb space and constraints.

Step 3. Blur the image if none of the tiles contain a dominant skin tone colour. This strategy is trying to prevent the false positives that are known about the cloud face dataset. Otherwise, sharpen with dominant colours to highlight Haar features and hopefully promote detection. We will use library functions and convolution in OpenCV to achieve this. This is a bit different compared to other papers that were mentioned as it is simplified to the integral picture as whole squares rather than with Bezier curves (predictably inferior, but worth seeing the results). We choose these starting base kernels to expand (depending on image dimensions) by experimentation on how we perceived the given cloud image to not have Haar features visible anymore.

$$SHARPEN\_KERNEL = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{4}$$

$$BLUR\_KERNEL = \begin{pmatrix} 85 & 85 \end{pmatrix} \tag{5}$$

Step 4. Feed into the detection methods of VJ or MTCNN. Throughout this process, we will need to be careful of BGR conventions that OpenCV uses (i.e. not RGB) and will need to implement methods to convert tuples from BGR to YCrCb to have some finer control over pixels. Note that we are using the full spectrum, not a constrained one used in hardware engineering. One such formula is as follows, and we will use the appropriate inverse

matrix as needed: [Y, Cr, Cb] =

$$\begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix} + \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad (6)$$

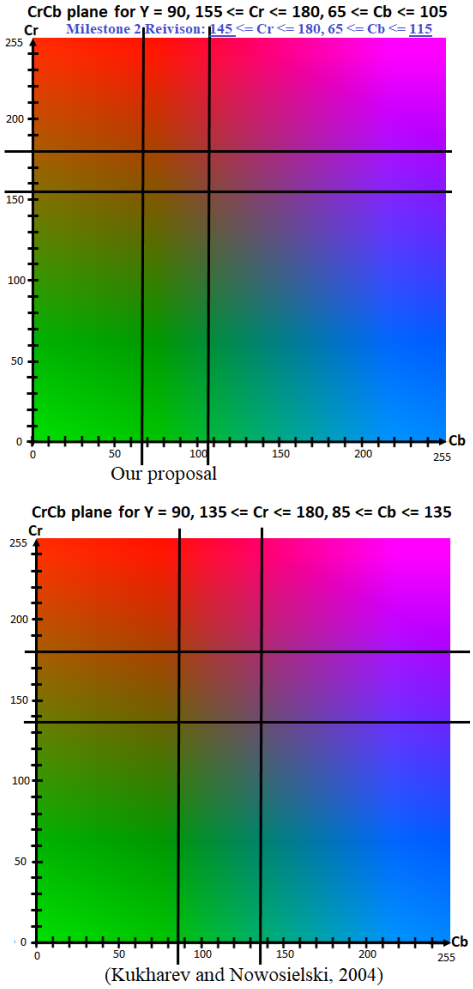


Figure 6: Our proposal for skin tone color range in YCrCb, compared with one suggested by (Kukharev and Nowosielski 2004). Both are simplified representations with set $Y = 90$. Color space generated by Kdenlive software and image edited with MS Paint.

We can also perform post-processing to determine if the face detected is indeed a face, by checking if the color of the face is present in the area detected. Similarly to the preprocessing step, we cut the area detected into 25 small regions, and check if the dominant color of any box is sensible human skin color. Considering that certain photos might have high variance in lightings or other factors that may affect the color of the faces, we can set the post-processing process to be a final activation function, with input including confidence of the face detected, and the difference between the majority color of the face and its closest sensible skin color.

Timeline of Algorithm Implementation

- June 30 - All of us will clone the MTCNN repository, install Tensorflow and OpenCV libraries on our development environment.
- By July 1 - Read all the readmes of the datasets that detail the serialization structure of how they represent the metadata, such as the number of faces to be expected. Each of us will pick one of the four datasets to examine, one of us will do two. For instance, Kai will deal with datasets 1 and 2, Tom with 3, and Brian with 4.
- July 3 - Implement basic parsing methods to the test harness to determine the expected results for a given data point within the set. As a simple example, Kai will add code to parse the filenames of the mugshots. We will deal with our respective datasets.
- July 7 - We should now be able to put together our testing harness in terms of accuracy validation by being able to run it with Viola-Jones and MTCNN. A simple shell script can be written by Brian to measure some environment metrics such as the runtime on a given dataset. Also, Tom will investigate if we will need to use an AWS instance to run MTCNN on. He will do some DevOps work on that end. Kai will do the actual hooking up of the test harness to the algorithms by calling the needed methods within the test harness. Again, we are doing a trial run of plain Viola-Jones and MTCNN at this point, no preprocessing algorithms.
- July 10 - Make progress on the preprocessing implementation. The first step is to divide the image into arbitrary equal-sized squares. Kai can add code to do that; Tom will then implement the color conversions from RGB to YCbCr and the skin tone range logic; Brian will implement gaussian blur with various experimental values.
- July 14 - We should be done implementing the preprocessing algorithm. We will re-do the items from July 7, this time with the preprocessing.
- After July 14, we will continue to examine Enriquez's idea to see if there is an easy way to generate weights using Haar features and train a working model. This will be an exercise in supervised learning, in that maybe we can just simply use Viola-Jones to detect faces on a huge, and varied dataset and use that as a training model on MTCNN. However, we already know this will be of lower quality than the pre-trained model as the original implementations used quite accurate human-examined data as the training set.

Results

Original Prediction from Proposal We expect MTCNN to be able to detect oblique faces within the dataset better than Viola-Jones. That is, more correct true positives will be found. However, depending on the training model that we end up using, more false-positives with MTCNN will also be found. Also, the amount of RAM and processing power running a MTCNN will be much greater than running Viola-Jones on the dataset. It is hard to estimate the exact magni-

tude without playing around, but we estimate that it will not be more than one order (10x). It would hence make sense that CNNs and complex models are likely not widely used in portable and weak cameras yet. As for the results of attempting to preprocess poorly taken images and feeding them into Viola-Jones, we expect something as simple as a blind sharpness adjustment should have a tiny improvement on a subset of images where darker-skinned individuals are in the same place as a lighter-skinned person. Maybe a few percentage points increase, a bit less to how the more advanced preprocessing adjustments have resulted in. This is not surprising as the paper evaluating much more complex preprocessing adjustments depending on the lighting source (Afifi et al. 2018) had only small improvements too. We expect the reduced and simplified subset that we approach this will have similar, but small results.

Preliminary results for Milestone 2

Results of Preprocessing One of the main factors that influence our preprocessing algorithm is the skin tone range definition. Figure 7 gives a visualization of this range using two examples.

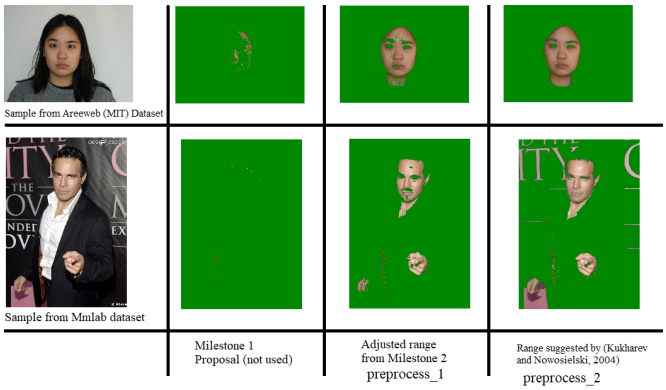


Figure 7: Dataset examples to help visualize the skin range we have implemented to use. The green background is because I did the mask while still in YCrCb mode and then converted it to BGR without another mask for the background in BGR (i.e. realized I did a bitwise when black is not (0,0,0) in YCbCr). This problem is isolated to the creation of this visualization diagram. Will provide a black background on final report appendices.

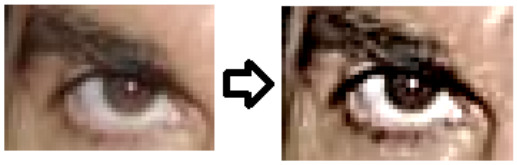
Note how "Preprocess 2" range definition can more often capture other artefacts from the environment such as the writing on the wall. However, our proposed range in "Preprocessing 1" leads to some tearing or distortion along Haar features such as the defined edges of eyeballs. Either way, it means that preprocess 2 was more likely to trip our logic to sharpen if the skin is detected, even if there is just writing on the wall in the false-positive dataset. One final point is that the division of tiles is not shown in the visualization in Figure 7.

The convolution blurring and sharpening was apparent on images that get sent to the detectors. Figure 8 provides the image that is provided to the detector (extracted after some modification at a checkpoint in the code). Note that it is quite challenging to see the Haar features in this as expected, which we were hoping would reduce the false positive rate.

BLUR_KERNEL = np.array([85, 85]) # Scaled in method.

Figure 8: Blurring to reduce the visibility of Haar features. Cloud picture from dataset 4 *ssbkyn.com*

Similarly, sharpening leads to Haar features being more visible, especially when the face is small or of lower resolution image. The contours of a face were slightly more apparent when we did a visual inspection. See the closeup of the eye below. Lastly, post-processing to reject a detection box (to encourage rejection of false positive) leads to negligible difference if added onto the preprocessing result since the logic for preprocessing already accounted for skin colour detection.



```
SHARPEN_KERNEL = np.array([[-1, -1, -1],
                           [-1, 9, -1],
                           [-1, -1, -1]])
# Base kernel. Will be scaled. See method.
```

Figure 9: Sharpening gives a clearer image of the contour of the eye, especially in a smaller sized image. Processed image from the Mmlab dataset.

Details of the Experiment We used OpenCV for our Viola-Jones implementations and MTCNN for our CNNs implementation. The datasets are selected based on what we proposed in the Methodology section. The details of each dataset are given below:

- Dataset 1 contains 2999 images from Mmlab, which are covered faces of a single person, including hat, glasses,

bangs and wavy hair.

- Dataset 2 contains 256 images from Areeweb and the datebase from MIT, which are side faces of a single person,
- Dataset 3 contains 252 images from *kaggle.com*, which contains multiple faces from different angles with real-life scenes, such as meetings, concerts, and sports games.
- Dataset 4 contains 7 'cloud-face' images from *ssbkyn.com* plus 90 random images generated from *randomwordgenerator.com*. None of them contains an actual human face.

The execution environment is Windows10 with the CPU Intel(R) i7-8700, a GeForce GTX 1060 is used as GPU acceleration for CNN algorithm. Only the execution time on dataset 1 is recorded for simplicity.

Detection rate on Dataset 1-3 with no preprocessing

	Dataset 1	Dataset 2	Dataset 3
Viola-Jones	0.832	0.612	0.385
CNNs	0.997	1.000	0.921

False-positive rate on Dataset 4 with no preprocessing

	Dataset 4
Viola-Jones	0.010
CNNs	0.134

Detection rate on Dataset 1-3 with Preprocess 1

	Dataset 1	Dataset 2	Dataset 3
Viola-Jones	0.656	0.387	0.293
CNNs	0.994	1.000	0.941

False-positive rate on Dataset 4 with Preprocess 1

	Dataset 4
Viola-Jones	0.021
CNNs	0.103

Detection rate on Dataset 1-3 with Preprocess 2

	Dataset 1	Dataset 2	Dataset 3
Viola-Jones	0.535	0.408	0.243
CNNs	0.803	1.000	0.65

False-positive rate on Dataset 4 with Preprocess 2

	Dataset 4
Viola-Jones	0.021
CNNs	0.093

Execution time (s) on Dataset 1

	Without preprocessing	With Preprocessing 1	With Preprocessing 2
Viola-Jones	67.338	298.491	257.62
CNNs	258.165	455.257	505.271

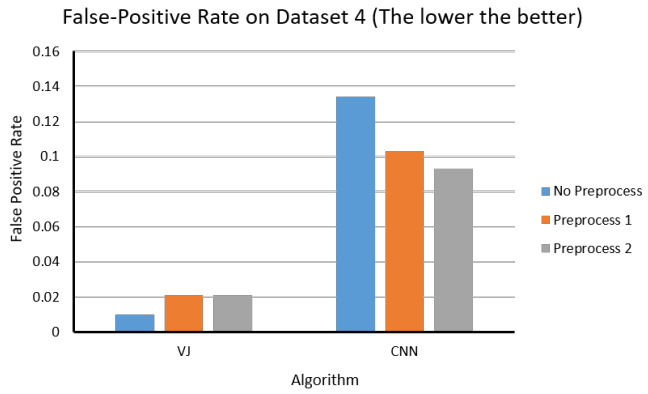


Figure 10

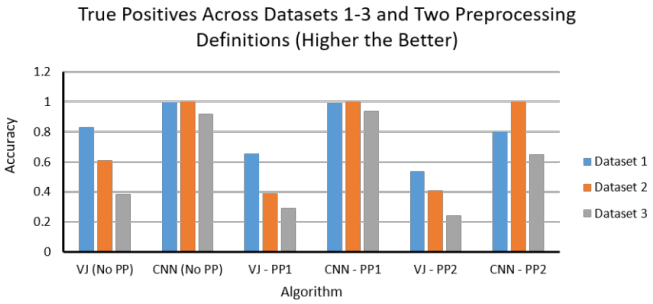


Figure 11

Discussion and Analysis of Results

Without any preprocessing, the first table shows that MTCNN gave a much better true-positive detection accuracy rate compared to VJ. This is especially true for dataset 2 (side profiles), and dataset 3 (diverse mixed set), where the rate was substantially lower on VJ, by about 38% and 54% respectively. The false-positive rate is reasonably low for both algorithm. However, we are surprised that MTCNN had a higher false-positive rate of 13.4% compared to VJ of just 1%. This dataset (dataset 4) should have 0 detections as there were no actual human faces. Also, wearing glasses or other obscurities on the face in dataset 1 did not impede MTCNN as much compared to VJ. Additionally, the runtime taken with CNNs is much higher than that of VJ in all cases. Without any preprocessing, CNNs took about 3.85 times longer to complete compared to VJ. This is a bit better than what we originally predicted (an order of magnitude

or 10x longer). Keep in mind this is just running the detection, not including running cross-validation training. Thus, it would make sense that VJ could still be used on the low end or cheap hardware products such as cameras.

The preprocessing strategy of blurring or sharpening based on simplified skin tone detection did not yield as much improvement to the false positive detection rate as we hoped. We were hoping to get the false positives down to near zero as the whole image is blurred when no skin tones are detected. When doing some manual analysis of our preprocessing technique, the Haar features seemed to be distorted significantly, yet VJ and CNN were still able to pick out the features in some cases (see figure 8 for an example) and determine it is a face. Also, the preprocessing has negatively affected true-positive accuracy rates much more than we had anticipated. A post-processing approach on the detection box might have been the better approach. Finally, there was a slight increase in false positives for VJ on dataset 4 when preprocessing was introduced, from 1% to 2.1%, but MTCNN improved in false positives, going down from 13.4% to 10.3% and 9.3% for preprocess 1 and 2 respectively. The reasons for the increase in false positives in VJ is not immediately clear, but we can speculate that a few of the sample images had big animal faces which despite the blurring, still allowed Haar features to slip through with the contour or boundary of the face ill-defined.

The skin tone definition difference between Preprocess 1 to Preprocess 2 resulted in better true-positive accuracy for Preprocess 1 in most cases. Perhaps the more narrow and restrictive skin tone definition resulted in less noisy areas (areas where skin boundaries are not well defined, as seen in figure 7) when compared to the broader definition in Preprocess 2.

Overall, the best (or rather most interesting) data point was that MTCNN was really accurate and robust for the side profile mugshots, even without the preprocessing. With the simple preprocessing that can be run on low-end hardware (simple skin tone detection), we were able to reduce the false-positive rate markedly for MTCNN. The worst data point was the scale in which preprocessing affected the true positive detection accuracy, with a reduction of about 19% on average across the three datasets. The trend of the markedly decreasing false-positive rate with either Preprocessing definition was promising. Unfortunately, the results were not as expected. We did not anticipate such a small improvement to the false positive rate compared to the higher reduction in the true-positive rate. Hence the disadvantages to our preprocessing outweigh the benefits when running them through the detection algorithms. Compared to the work of (El Maghraby et al. 2013), our implementation is simplified in both the skin tone range definition (linear vs complicated parametric equations) and grid logic used to trigger (dominant colour in grid tile vs per pixel to isolate face). So, it is not surprising that our results are possibly due to an oversimplification of constraints.

Conclusion and Future Work

Conclusion

Face detection, as an achievement in the development of artificial intelligence, has become an essential technology that is used in people's daily life. Since face detection is mostly used for security purposes, this has raised our concern of how our existing algorithms perform. By reading related work, we learned that there are primarily two algorithms that are used to implement face detection, Viola-Jones algorithm and an Convolutional Neural Networks. We would like to investigate their performance in different real-life cases and discover their strength and weakness. Also, we wanted to implement our own preprocessing method that would hopefully improve the accuracy of both algorithms.

Our first goal is to compare the performance between Viola-Jones and CNN. For the implementation of the two algorithms, we took advantage of existing libraries. We used Viola-Jones implementation from OpenCV and CNN from MTCNN. We also created four datasets using images from the Internet. As our second goal, we tried to improve the accuracy (especially with false positives), by detecting skin tones in a 25x25 tile as a preprocessing step. If no image tile contained a human skin tone as the dominant colour, the entire image would be blurred in an attempt to make Haar features less visible and thus prevent a false-positive detection. Otherwise, the image would be sharpened. Our approach differs from others such as (El Maghraby et al. 2013) as it uses a simplified skin tone definition and convolutional adjustments are made from one base kernel adjusted for image size.

Overall, the results from our experiment proved our guess in the first goal. CNN found more faces among all datasets compared to Viola-Jones (dataset 1-3), which means that CNN is robust and capable of more complicated situations. However, CNN finds more false-positive cases in our test of dataset 4. From these results, we can draw a conclusion that if false-positive cases can be tolerated, we should use CNN because it can generally find more faces. However, when security is the primary purpose, we should use Viola-Jones instead. For our second goal, the results from our experiment do not agree with our expectations. Our preprocessing method does not improve the false-detection rate significantly. In contrast, the true-positive rates for both algorithms decreased to an unacceptable level. We conclude that the problem is actually much more complicated than we thought, and we need to put more effort into investigating the principle of algorithms and our preprocessing technology.

Future Work

Since we did not totally achieve the results of what we hoped for with the preprocessing step, there will be much research that can be done to improve our preprocessing methods. For example, how does the choice of colour range affect the accuracy of the algorithm? We have seen improvement from Preprocess 2 to Preprocess 1 where Preprocess 2 includes more colours than Preprocess 1. However, when we tried to cover even more colours, the accuracy did not increase. People who are interested in this can perform more experiments

to investigate how the choice of colours can affect the accuracy and possibly receive the best colour range. Also, there are several variables in our experiments, including colour selecting, sharpening and blurring. We did not control the variables rigorously during the experiment, so all our tests have the assumptions that those variables are independent. However, there is a possibility that those variables are not independent. Are those variables really independent? What are the relations between them if they are dependent? How do they affect the accuracy compositely? These are all valuable topics that are worth investigating to improve the performance of our preprocessing methods. Since our preprocess methods do not agree with our expectations, we might think that post-processing can be another technology to improve performance. How does post-processing work, and how can we implement post-processing? This will be a totally new story.

Unfortunately, we did not have time to research what Enriquezs proposed in 2018, that is train a new Convolution Neural Network that uses Viola-Jones Haar as the filter. Our experiments have proven that both of the two existing algorithms have their disadvantages. Viola-Jones is not capable of different situations, while the existing CNN requires more computational resources and finds more false-positive cases. For the people who want a new generation of face detection, this is a promising direction. According to Enriquez, this new CNN will overcome the defect that the existing CNN has and will combine the advantages of Viola-Jones and the existing CNN: it requires fewer resources and increases the accuracy significantly. Since face detection has become an important technology that is used in people's daily life, it is definitely worth it for future researchers training this new CNN.

References

[Afifi et al. 2018] Afifi, M.; Nasser, M.; Korashy, M.; Rohde, K.; and Mohamed, A. A. 2018. Can we boost the power of the viola–jones face detector using preprocessing? an empirical study. *Journal of Electronic Imaging* 27(4):043020.

[Bichwe, Bichwe, and Satija 2016] Bichwe, M. R.; Bichwe, S.; and Satija, S. 2016. Design and implementation of resampling techniques for face recognition using classical lda algorithm in matlab. *International Journal of Computer Applications* 152(6).

[Dwiyantoro] Dwiyantoro, A. P. J. The evolution of computer vision techniques on face detection, part 2.

[El Maghraby et al. 2013] El Maghraby, A.; Abdalla, M.; Enany, O.; and El Nahas, M. Y. 2013. Hybrid face detection system using combination of viola-jones method and skin detection. *International Journal of Computer Applications* 71(6).

[Enriquez 2018] Enriquez, K. 2018. Faster face detection using convolutional neural networks & the viola-jones algorithm.

[Ephraim, Himmelman, and Siddiqi 2009] Ephraim, T.; Himmelman, T.; and Siddiqi, K. 2009. Real-time viola-jones face detection in a web browser. In *2009 Canadian Conference on Computer and Robot Vision*, 321–328. IEEE.

[Farfade, Saberian, and Li 2015] Farfade, S. S.; Saberian, M. J.; and Li, L.-J. 2015. Multi-view face detection using deep convolutional neural networks. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, 643–650.

[Kukharev and Nowosielski 2004] Kukharev, G., and Nowosielski, A. 2004. Visitor identification-elaborating real time face recognition system.

[Lienhart and Maydt 2002] Lienhart, R., and Maydt, J. 2002. An extended set of haar-like features for rapid object detection. In *Proceedings. international conference on image processing*, volume 1, I–I. IEEE.

[Rowley, Baluja, and Kanade 1998] Rowley, H. A.; Baluja, S.; and Kanade, T. 1998. Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence* 20(1):23–38.

[Viola and Jones 2001] Viola, P., and Jones, M. 2001. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, I–I. IEEE.

[Zhang et al. 2016] Zhang, K.; Zhang, Z.; Li, Z.; and Qiao, Y. 2016. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters* 23(10):1499–1503.