PERFECT SELFIE THAT CAN BE USED IN FACE RECOGNITION WITH A PASSPORT PHOTO

R.H. Rieksen
r.h.rieksen@student.utwente.nl

Abstract

The goal of this paper is to describe the development of an android application designed to make a perfect selfie that can be used in face recognition software with a passport photo. By using an illumination check and Viola-Jones landmark detection the idea is to give the user feedback how to meet the requirements a passport photo is held to. The closer the selfie looks like a passport photo, the better success rate face recognition software will have.

1. Introduction

Face recognition is used to identify individuals from facial images. Nowadays all passports have a NFC-chip in them with personal information and a passport photo of the user. For important issues such as opening a bank account it is needed for the individual to identify himself in person at the bank. The idea is that an individual can identify himself online with the help of an application. This application would exist out of four major parts.

- Reading data from passport
- Live face and facial features detection that will be used to give the user feedback how to make a live selfie that looks like a passport photo
- Anti-spoofing methods
- Face recognition that compares a passport photo to the live selfie

This paper focusses on the second part where live face and facial features detection will be used in making a passport photo selfie.

2. OpenCV

The application is designed in an android environment and depends heavily on the open-source library OpenCV 2.4.8.2. The installation of OpenCV is explained in [2] but is very hard for beginners. For beginners it is recommended to install the standard nvidea Tegra package [3] which includes among others Android SDK, NDK, Build tools and Platform tools.

3. Viola-Jones Detector

Haar-like feature classifiers are created using integral images. Each pixel in the integral image represent the change in brightness of the corresponding pixel in the image. This is done by making each pixel equal to the entire sum of all pixels above and to the left of the concerned pixel. This is demonstrated in Figure 1.

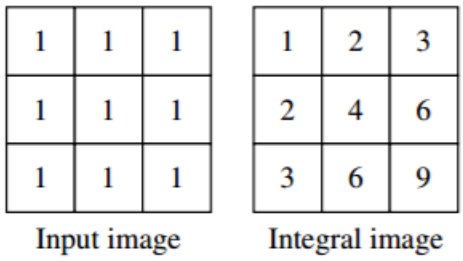

Figure 1. Integral image

Cascades are trained by running multiple classifiers one by one (stage) against a dataset of images. These images consist of positive

images with the desired feature and images without the desired feature. A process called AdaBoosting will give higher weight to the wrong sorted image data and will then run another classifier.

The job of each stage is to determine if a given sub window is definitely not a feature or maybe a feature. When it is definitely not a feature the sub window is immediately discarded. When the sub window is a maybe feature it is passed on to the next stage. The more stages a sub window passes the higher the chance it contains a feature illustrated in figure 2.

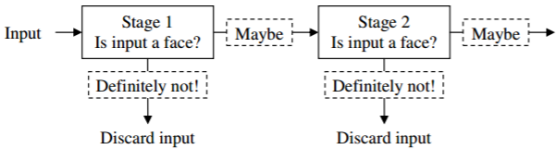

Figure 2. Cascasded classifier

This process makes sure less time is wasted on areas in an image which have little to no change to represent a desired feature. This framework was introduced by Paul Viola and Michael Jones in 2001 [4].

4. Designing the application

The application is required to have a good live performance (framerate) and has to be stable. The beginning of the project consists of an example code from OpenCV called face-detection. This application uses Viola-Jones detection to detect faces within a video frame given by the back camera and draws rectangles around them. The face detection uses a gray image from the camera output and only seemed to work when the phone was hold horizontally like in figure 3.

Figure 5. Horizontal phone

For a selfie the front camera has to be used. In our case when displaying the incoming camera frames onto the screen our face was shown upside down. This was fixed by flipping the frame 180 degrees.

Figure 3. Vertical phone

The starting position {0,0} of the screen is for the horizontal phone in the top left corner. When the phone is hold in selfie mod, seen in figure 4, the starting position {0,0} is in the top right corner. For the face detection to work the image has to be rotated ninety degrees.

Since the user has to be given feedback on the screen an area was made in which the user is requested to place his face in. This area is standard red outlined which means the user did not meet the requirements to make a perfect selfie. When these requirements are satisfied the color will change from red to green. In figure 5 there are three examples of where in the first two cases the requirements are not satisfied.

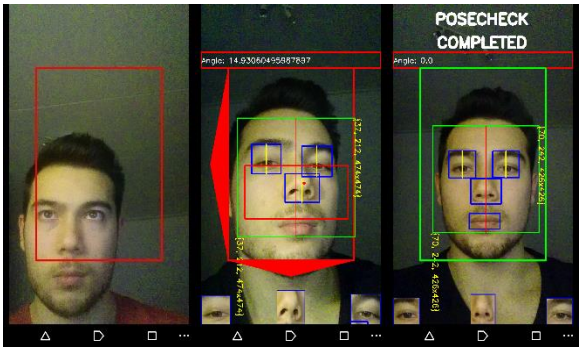

Figure 4.a and b the requirements are not satisfied, c the requirements are satisfied.

5. Illumination

Illumination variation is one of the challenging problems to be solved for robust face recognition systems. The changes induced by illumination, such as cast shadows or attached shadows, can be larger than the innate differences between individuals [4]. Not only does illumination variation

influence the performance for face recognition it also drops the performance for good landmark detection. One requirement of a passport photo that will also be a requirement of the selfie is that the illumination on the face is equalized.

A fast detection for an equalized illumination check is implemented and used as following. The input image is divided into four sections with an equal height and width. The sections are in green displayed in figure 6a. These sections will be resized for faster processing to meet the requirement of a fast application.

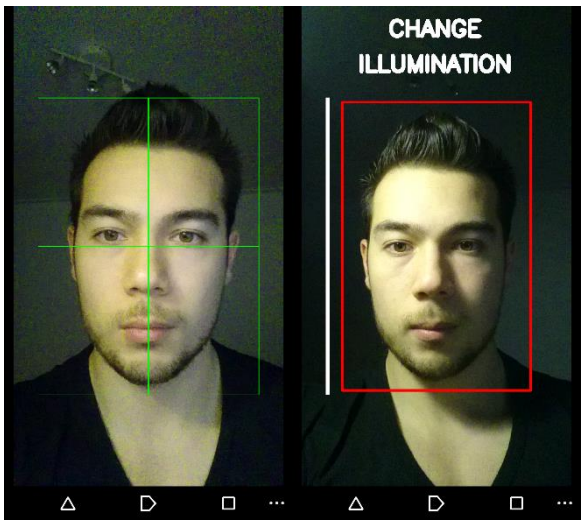

Figure 6.a Sections on which an illumination check will be done, b illumination check

As seen in figure 6a the four sections are within a window. Of each section the average gray value is calculated. The gray value will be higher the whiter the pixel is. This means the higher the illumination the higher the average gray value.

While making a selfie the assumption can be made that the user will have his face placed inside this window and the background image is neglectable. When we examine the sections the first thing we notice is that the upper left section mirrors the upper right section and the bottom left section mirrors the bottom right section. By adding the average gray value of the two left sections and comparing them to the added average gray value of the right sections an easy algorithm and fast algorithm is found for unequal illumination with a light source aimed from a side angle towards the face.

When the average gray value of one side of the sectors is higher than the other side there will be given feedback to the user in the form of a message and a white beam that appears onto the side of the screen where the light intensity is higher as can be seen in figure 6b.

When looking for the average gray values in these sections only a light source aimed from a side angle can be determined. There is no way to compare the average gray value of these section to determine when a light source is aimed from an upwards or downwards angle. In comparison to the mirrored right and left section the upper and bottom sections are total different. In the bottom sections a part of the neck and shoulder is displayed and the top sections also includes the hair. Even if the neck/shoulder and hair can be excluded from the average gray value, facial hair will still be a problem. In figure 7 an image is shown with an upwards angled illumination which obviously passes the illumination check.

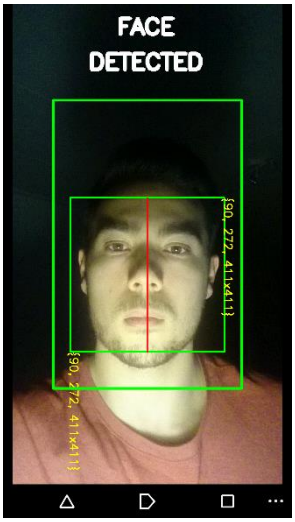

Figure 7. Upwards angled illumination

Only when the illumination on the left side of the face mirrors the right side the Viola-Jones face detection will find place.

6. Face detection

The first step in making a perfect selfie is the detection of the face. In OpenCV there are two different kind of cascades that can be used with Viola-Jones face detection. The lbp-cascade uses for its calculation integers while the haar-cascade uses floats. This makes the lbp-cascade a few times faster than the haar-cascade but also around 20% less accurate [6]. In OpenCV the Viola-Jones detection with the cascade gives back a starting position (x, y) and the width and height of the area in which the face is detected.
The used lbp-cascade and haar-cascade files are both include in OpenCV [7] and their name can be found in table 2.

Cascade	Name
LBP	Lbpcascade_frontalface.xml
Haar	Haarcascade_frontalface_alt2.xml

Table 1. Cascade names

The two cascades have been evaluated in two ways and are compared with each other on performance (frames per second) and stability. The performance is tested by looking how long it takes for the application to display hundred frames on the screen of the phone when a face is detected. The average frames per second are presented in table 2. The lbp-cascade is around four times faster than the haar-cascade.

Type cascade	Frames per second
LBP-cascade	8-12
Haar-cascade	2-3

Table 2. Performance haar and lbp cascade

The stability of the cascade is tested as following. The input image on which the Viola-Jones detection will be calculated is static, this means that the received data (x, y , width, height) should always be around the same value. For both cascades there will be looked at a frontal face image, an image of a face that is rotated to the left and is still well within the boundaries of the used cascade to be detected and an image of a face turned to the right that is just near the edge of the cascade to detect a face in it. The three faces are shown in figure 8.
In this experiment the x-position, y-position, width and height of the detected face area will be saved for twenty-five frames. To give an impression of how stable the cascade is, the values will added and divided by twenty-five to receive an average value. Then for each value the difference with the average value will be calculated, which is called the pixel distance from average. Only the x-position will be examined in this experiment.

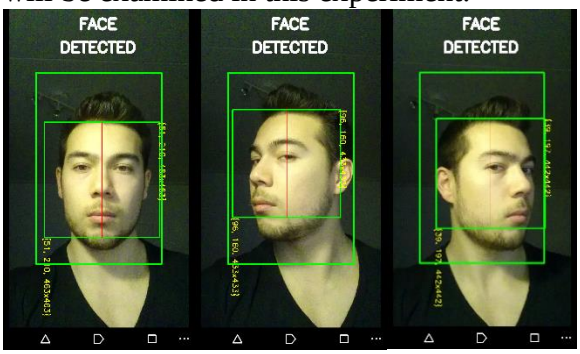

Figure 8. a frontal face image, b rotated to the left, c rotated to the right

6.1. LBP-cascade stability results

Figure 9 represents the pixel distance from the average with the use of a lbp-cascade.

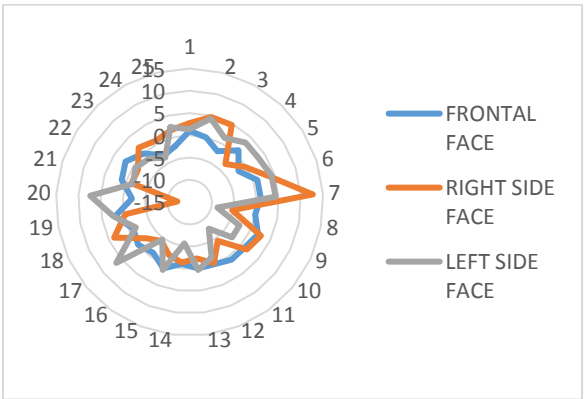

Figure 9. X-Pixel distance from average over 25 frames with the use of a lbp-cascade

As can be seen in the figure above is that the pixel distance from the average for a frontal face is pretty constant. When the face is rotated to the left in an angle that is still well within the boundaries of the cascade the pixel distance from the average gets more inconsistent with some peaks and valleys. When the angle of the rotated face gets closer towards the boundaries in which a face can be detected, in this case the right turned face in figure 8c, even higher peaks and valleys will arise.
The lbp-cascade is really fast with a framerate between eight to twelve frames per second and is pretty stable for a frontal face detection. But when it comes to the detection of face that is turned the stability will drop and can be pretty inconsistent.

6.2. Haar-cascade stability results

Figure 10 represents the pixel distance from the average with the use of a haar-cascade.

The pixel distance from the average for a frontal face is near equal as that of the left turned face as can be seen in figure 9. Near the boundaries of the cascade when the face is turned to the right the pixel distance still got a lot of peaks and valleys. The values of these peaks declined compared to the ones of the lbp-cascade.

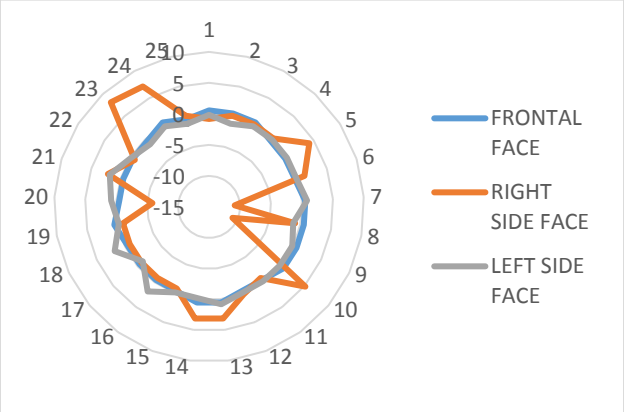

Figure 10. Pixel distance from average over 25 frames with the use of a haar-cascade

The haar-cascade is slow with a framerate between two and three frames per second but is more stable when it has to detect a rotated face. This can be seen when the largest peak and valley values are compared to the peak and valley values of the lbp-cascade.

6.3. Combined lbp and haar-cascade

The lbp-cascade is really fast but the stability of the cascade drops when the face that has to be detected is rotated. In contrary to the lbp-cascade, the haar-cascade is a lot slower but the stability for a rotated face is a lot better.

When both cascades are combined the expected result should be a cascade that has a performance between a lbp and haar-cascade and has the stability of a haar-cascade. Since the lbp-cascade is a lot faster than the haar-cascade it is first used to detect the area in which the face might be in. The values of this area are then given forward to the in series connected haar-cascade which can then run a more accurate face detection. This way the slower haar-cascade has a better pre-defined area to reduce the amount of calculations needed. This is illustrated in figure 11.

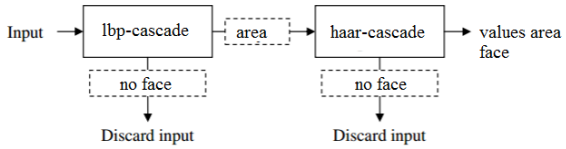

Figure 11. Lbp-cascasde in series with a haar-cascade.

The same way as before the framerate of combined cascade is determined. Table 3 now as well includes the framerate of the lbp-cascade in series with a haar-cascade.

Type cascade	Frames per second
LBP-cascade	8-12
Haar-cascade	2-3
LBP + haar-cascade	4-6

Table 3. Performance different cascades

Figure 12 represents the pixel distance from the average with the use of a lbp-cascade in series with a haar-cascade.

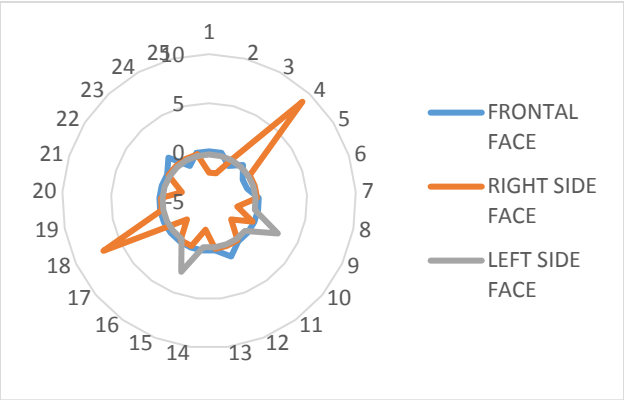

Figure 12. Pixel distance from average over 25 frames with the use of a lbp-cascade in series with a haar-cascade

As expected the results of the combined cascade matches the predictions. The combined cascade is twice as fast as a normal haar-cascade which is the result of a better pre-defined search area for the haar-cascade. The stability of the combined cascade also matches the predictions. The pixel distance from average from the combined cascade almost equals that of the haar-cascade for the frontal face and to the left rotated face. For the more to the right rotated face the peaks and valleys are still quite random but the values of these peaks and valleys also matches that of the haar-cascade. In this application the combined cascade will be used because it has the best tradeoff between performance and stability.

7. Landmark detection

After the face has been detected the next step will be to detect facial features like the eyes, nose and mouth, this is also called landmark detection. OpenCV has included multiple haar-cascades that can be used to detect these facial features. As seen with the face detection the use of haar-cascades in Viola-jones detection are several times slower than when lbp-cascades are used. To improve the performance of landmark detection a region of interest is defined for each facial feature. A region of interest is an area in which the landmark detection will look for the facial feature, similar as done for the face detection.

For the detection of both eyes, nose and mouth there are needed four regions of interest. On every region of interest another haar-cascade will be used with a Viola-jones landmark detection.

The area information of the detected face we call ***face*** and includes the x-position, y-position, width and height. ***face*** is defined to have its starting position in the top right corner of the green rectangle in figure 13a.

All facial features are positioned in ***face***. This means the region of interest of the facial features are all a subarea of ***face***. This makes it possible to define each region of interest based on the width and height of ***face***. As an example the region of interest of the right eye is specified as following. Starting from one fifth of the top of ***face*** as vertical position and one sixteenth of ***face*** as horizontal position. The width of the horizontal area is seven sixteenth of ***face*** and the height of the vertical area is one third of ***face***. Table 4 has all the specification of the used regions of interest based on ***face***.

Facial feature	X	Y	Width	Height
Right eye	1/16	1/5	7/16	1/3
Left eye	1/2	1/5	7/16	1/3
Nose	1/16	2/5	7/8	9/20
Mouth	1/4	7/10	1/2	3/10

*Table 4. Specification for their designated region of interest, values have to be multipled with **face.***

- X is defined as the horizontal starting position from the most right value of ***face*** to the left.
- Y is defined as the vertical starting position from the most upper value of ***face*** to the bottom.
- Width is defined as the horizontal width.
- Height is defined as the vertical height.

The used haar-cascade files are both include in OpenCV [6] and their names can be found in table 5.

Cascade	Name
Right eye	Haarcascade_righteye_2splits.xml
Left eye	Haarcascade_lefteye_2splits.xml
Nose	Haarcascade_mcs_nose.xml
Mouth	Haarcascade_mcs_smile.xml

Table 5. Used Haar-cascade names

In figure 13a all the region of interests are drawn. The left blue square is for the left eye, the right blue square for the right eye, the small red square for the nose and the yellow square for the mouth. In figure 13b the landmark detection of the eyes nose and mouth are drawn.

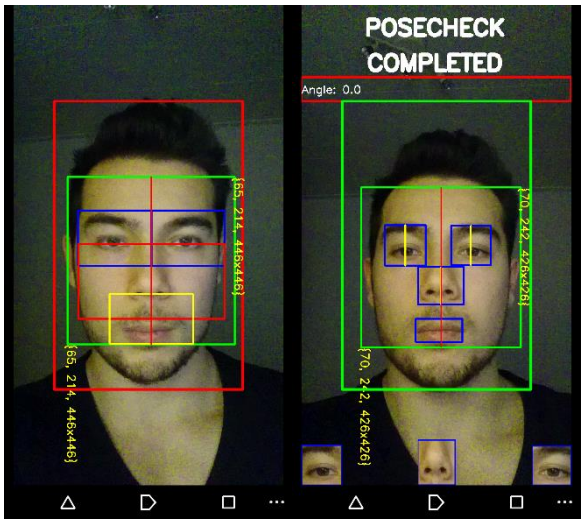

Figure 13.a region of interests, b landmark detection of the facial features

The landmark detection of the mouth is not used for a pose estimation and therefore will not be treated furthermore. The nose is the most important facial feature and is the basis for a good pose estimation and angle calculation in which the head is turned. When the face is turned the position of the nose will change dramatically while the position of the eyes will be around the same. This means to keep detecting the nose in a turned face it is required for the region of interest to be wide, as can be seen in figure 14.

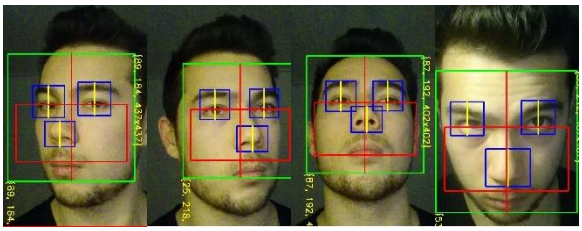

Figure 14. Nose placement inside its region of interest for different face rotations

8. Pose estimation

For a perfect selfie the only pose that is interesting is a frontal face. The nose alone can be used for a pretty good frontal pose estimation. As reference point for a pose estimation we split ***face*** in the middle with a red line**,** as can be seen in figure 13 and 14. A frontal face image means that the nose for an average person with a normal nose should be horizontally exactly in the middle of his face. The average vertical position of the nose should also be around an average value.

If the horizontal position of the nose compared to the middle axes of ***face*** is different it means the face is turned to the left or to the right. If the vertical position of the nose compared to the average vertical position is different it mean the face is turned upwards or downwards. To compensate for some inaccuracies in received nose area values and to compensate for noses that deviate from the average a margin is specified in which the nose position has to be in between to be a frontal face image.

The horizontal margin is taken as one percent of the width of ***face***. The vertical margin is a little more complicated. The assumption is made that the vertical position of the nose for a frontal face will be around the middle of the region of interest of the nose. We call the specifications of the region of interest of the nose ***Nroi***. The vertical margin is take as five percent of ***Nroi***.

$$hMarge = 1 * \frac{face.width}{100}$$

$$yMarge = 5 * \frac{Nroi.height}{100}$$

The x-position of the red line in the middle of ***face*** we call ***wd***.
The x-position and the y-position of the middle of the detected nose area are called ***nose.x*** and ***nose.y***.

For $nose.x < wd - hMarge$ the face is rotated to the right.

For $nose.x > wd + hMarge$ the face is rotated to the left.

For $nose.y < \frac{Nroi.height}{2} - yMarge$ the face is turned upwards.

For $nose.y > \frac{Nroi.height}{2} + yMarge$ the face is turned downwards.

For the selfie to be in a perfect frontal face position the only information needed is in which direction the face is rotated. Based on this information feedback can be given to make a perfect selfie. In the same way it is for example possible to give the user feedback to keep his head in a rotated angle to the left for a certain amount of time. An basic form of anti-spoof testing for liveness detection is giving the user a challenge response. First we need to take a better look at the frontal face pose.

8.1. Frontal face pose

The received information from the pose estimation is used to give the user feedback how to rotate his head to be in a frontal face position. When the user his face for example is turned to the right he will receive live feedback on his screen with the help of an arrow which way he has to move his head to get into a perfect frontal face position. The four standard movement up, down, right and left are shown in figure 15. The application can give multiple feedbacks at the same time, for example to turn your head upwards and to the right.

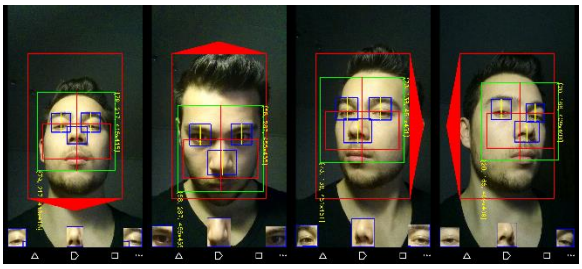

Figure 15. Feedback system towards a frontal face image

To test the stability of our frontal face pose we used the OUR database [7] to compute the pose estimation errors. The Our database contains 6660 images of 90 subjects. Each subject has 74 images, where 37 images were taken every 5 degree from right profile (defined as +90°) to left profile (defined as -90°) in the pan rotation. The remaining 37 images are generated (synthesized) by the existing 37 images using commercial image processing software in the way of flipping them horizontally. For two fast tests only used the images between -35° to 35° of twenty subjects are used to stay well within the boundaries of the face detection to work.

An important feature of the frontal face pose detection is its margin of error for non-frontal face images. The first information we are interested in is how many and what kind of errors our frontal face pose produces for purely 0° rotated frontal face images. The second thing we need to test is what the connection is between the horizontal rotation of the face and the produced errors that say if the face is rotated or a perfect frontal face pose.

8.2. Results

In table 6 the results of test one are processed. When the frontal face pose of a frontal face image was tested the algorithm determined in seventy-five percent of the cases that it concerned a frontal face image. The distribution of the error that shows if the face was rotated up, down, left or right shows some consistencies. The most important explanation for these error can be blamed on how we designed our algorithm to detect a

correct pose. As explained earlier the algorithm is purely based on averaged positions of in what small area the nose has to be in to be in a frontal position. When a nose deviates from it, errors are produced.

Correct Pose	Up	Down	Right	Left
75%	5%	10%	5%	5%

Table 6. Frontal face images pose estimation.

In table 7 and its plotted figure 16 the results of test two are processed. It is seen that for input images that are rotated closer towards the 0° frontal face image the algorithm detects increasingly more correct frontal face pose while they should be detected as a wrong pose. One reason for this is explained in the results of the previous test and is produced by noses that deviate from the average. Another reason might refer back to how the face detection stability is directly influenced by the rotation of the face. The other reason is that the nose detection doesn't detect the nose tip.

Degrees turned head	Correct pose	Wrong pose
5°	20%	80%
10°	10%	90%
15°	5%	95%
20°	0%	100%
25°	0%	100%
30°	0%	100%
35°	0%	100%
-35°	0%	100%
-30°	0%	100%
-25°	5%	95%
-20°	10%	90%
-15°	15%	85%
-10°	25%	75%
-5°	40%	60%

Table 7. Turned face images pose estimation (Correct pose is the error frontal face)

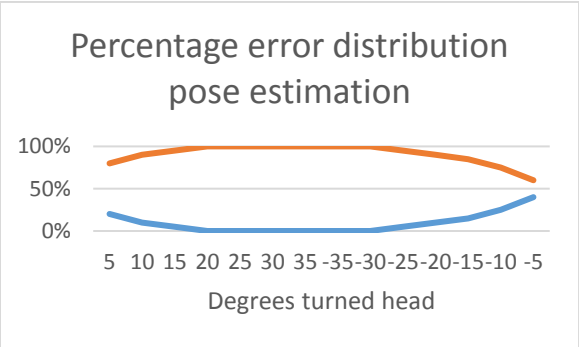

Figure 16. Error distribution pose estimation.

9. Nose tip detection

As stated previously the nose detection is not always as stable. For it to become stable an option is to look if the tip of the nose can be detected. An option for a possible nose tip detection can be found when we take a look at the image of a nose and its illumination.

In the first example we consider a frontal face image. When we start at the nose tip with a frontal illumination the light intensity should have the highest value because the nose is the closest to the light source and perpendicular to it. When we go further away from the nose tip we reach the side of the nose. Here the light source reaches the spot in an angle which means the light intensity will be lower. After that the cheeks are reached. The cheeks are also pretty close to perpendicular which means the light intensity will be closer to the light intensity of the nose tip.

Now the face is turned to the left. On the right cheek the light intensity should be high then when the side of the nose is reached the light intensity should drop. When the nose tip is getting closer the intensity should rise and when the nose tip is passed the intensity should drop again until the cheek is reached where the intensity should rise again. When the face is turned to the right exactly the same thing happens only reversed. In figure 17 the process of light intensity on the nose is easily seen and an example of the expected light intensity is drawn in figure 18.

Figure 17. a left rotated nose, b frontal nose, c right rotated nose.

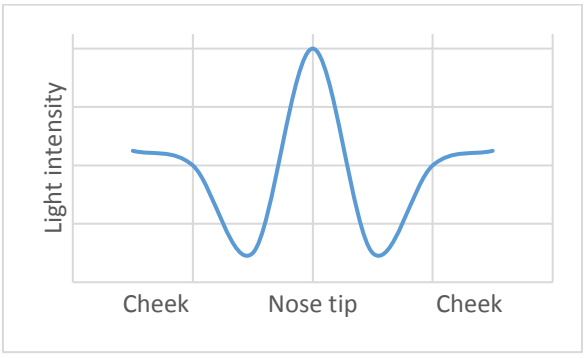

Figure 18. Expected light intensity of the nose.

In this test there has only be looked at the gray value of one line in the detected nose area. This is done to keep the performance high. The vertical position of the line is defined at the location of the nose for a frontal face image. The following data in figure 19,20 and 21 is received when plotting the gray value of the nose with a frontal face, rotated to the right and rotated to the left face.

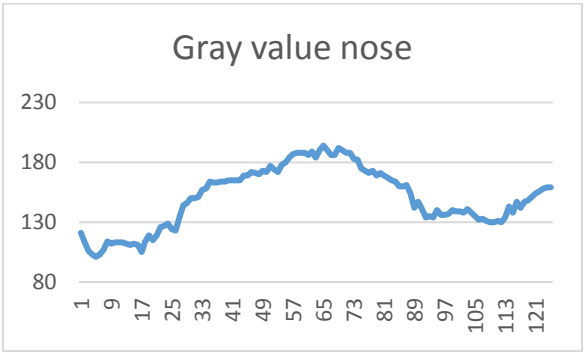

Figure 19. Gray value frontal face.

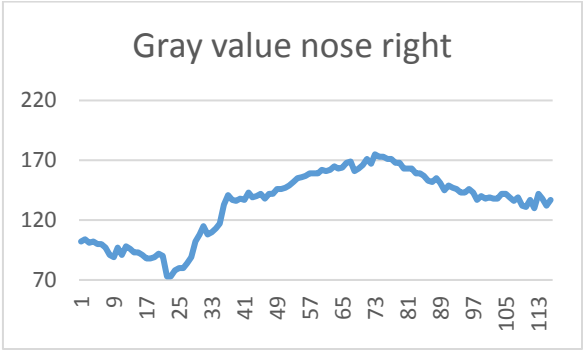

Figure 20. Gray value rotated to the right face

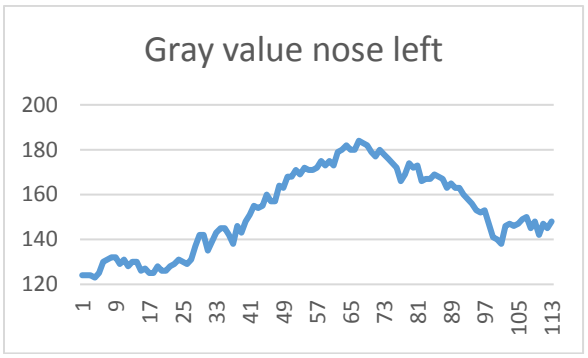

Figure 21. Gray value rotated to the left face

The data in the three figures above got a lot of noise in it. The most likely reason for this is the camera that has been used and the resolution of the camera input. To get rid of most of these spikes the data is averaged over three points. The following graphs are then received.

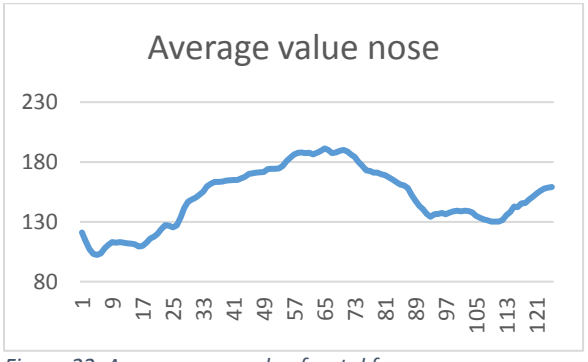

Figure 22. Average gray value frontal face

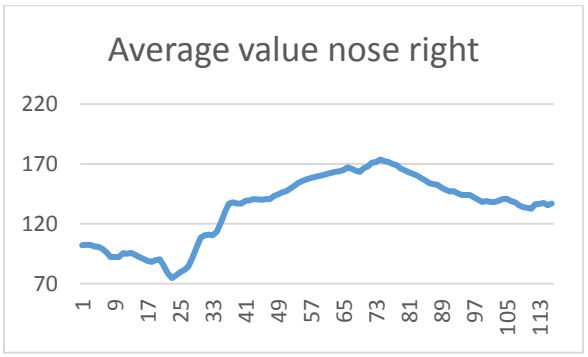

Figure 23. Average gray value face rotated to the right

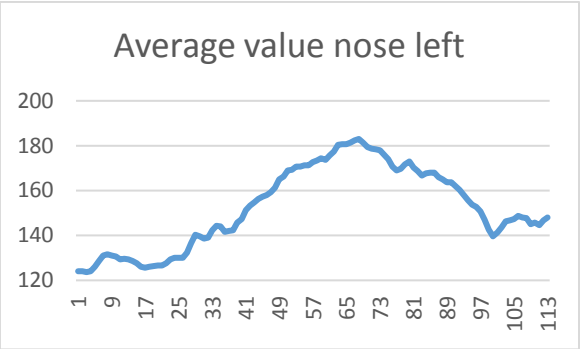

Figure 24. Averaged gray value face rotated to the left

To get usefull information out of the three graphs above the peaks have to be located. The average gray value is calculated by taking the sum of all gray values and divide it by the total amount pixels on the line. Since only the peaks are relevant for the detection of the nose tip all gray values below a certain value can be put on zero.

The highest gray value is ***hgv*** and average gray value is ***avr***.

If $gray\ value < \frac{hgv+avr}{2}$f then gray value will be set to zero.

The following graphs can then be drawn.

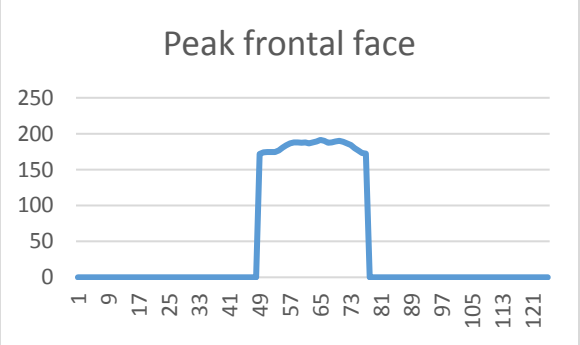

Figure 25. Edited gray value for frontal face

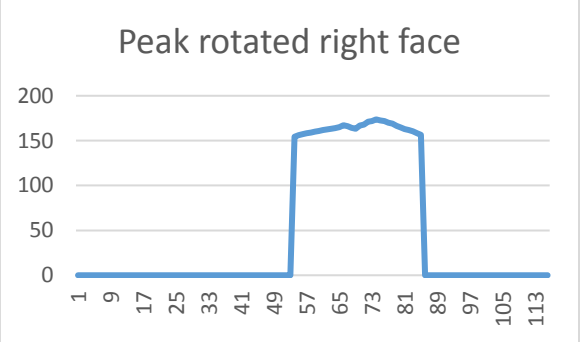

Figure 26. Edited gray value face rotated to the right

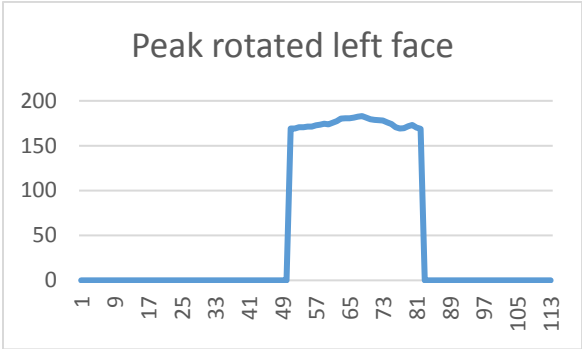

Figure 27. Edited gray value face rotated to the left

All three graphs pretty much look the same. In some conditions it is possible for the edges on the graphs to contain their gray value, for example when a little piece of a cheek is displayed in the detected nose image. This is easily resolvable by just looking at the peak closest to the middle point of the line. In this case around sixty pixels.

When figure 25 is compared to figure 17b it is easily noticed that the nose tip got the highest light intensity. The left side and the right side of the nose tip should have around the same intensity and that matches with the data from the test. The nose tip is located in the middle of the peak.

However when the face is turned the highest light intensity is not on the nose tip as can be seen in figure 17a and 17c. The nose tip still got a high light intensity but compared to a frontal face the nose tip is relocated from the middle to the side of the peak. To determine the nose tip when the face is turned to the right the most left value of the peak is taken and when the face is turned to the right the most right value of the peak is taken. This value is then averaged with the value from the detected nose into a position for the nose tip. To determine which side the face is turned the previous pose estimation is used. In figure 28 the calculated nose tip is displayed as a red dot.

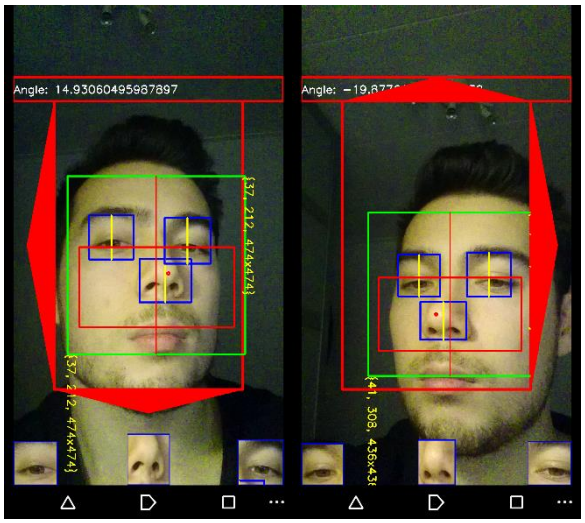

Figure 28. Nose tip detection.

To test the stability of our nose tip detection we used the MUCT database [9] to compute the nose tip detection error. The MUCT database consists of 3755 faces with 76 manual landmarks. These landmarks are the 68 points defined by the popular FGnet markup of the XM2VTS database [10], plus four extra points for each eye as can be seen in figure 29.

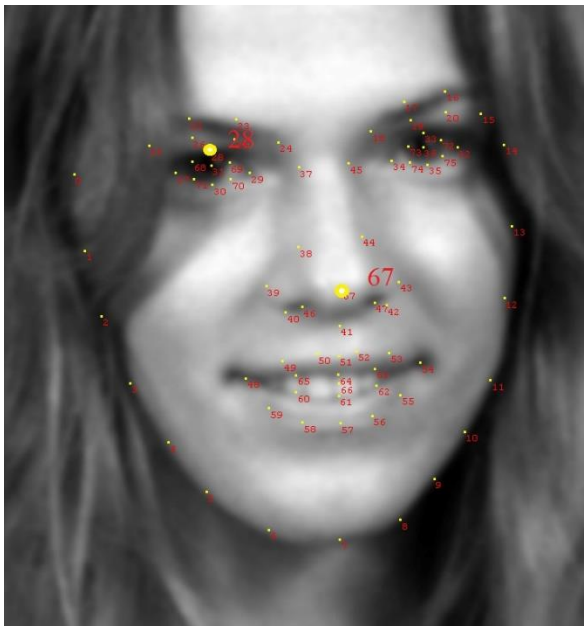

Figure 29. Landmarks placement.

Because of some issues the images could not be loaded directly into our application. For this reason the setup consisted of a phone aimed towards a screen on which the images were shown. Because of our setup it is not known how much influence it has on the values. In figure 30 the screen of the phone is shown when looking at the screen of my laptop.

To compare the data of the landmarks provided with the MUCT database with our received data it is not possible to look directly to the received landmark positions. Our solution was to determine the horizontal and vertical difference of landmark 28 and landmark 67 in the MUCT database. The middle of the left eye area received of the left eye detection matches landmark 28 and the nose tip position matches landmark 67.

The average horizontal and vertical difference between the detected left eye and nose tip has been calculated for landmark 28 and 67 and for our own received landmark positions.

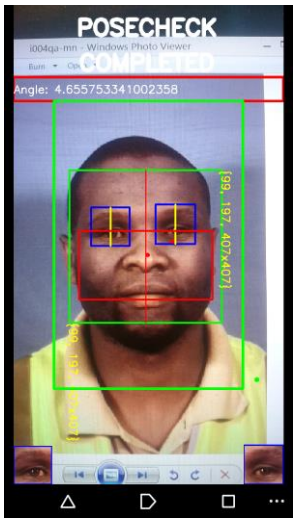

Figure 30. Setup of how the application receives the images

9.1. Results stability test nose tip

For the first 20 images in MUCT the data of the landmark positions in MUCT are shown in table 8 and our found data of the landmark positions are shown in table 9.

Left eye		Nose tip		X/Y difference		Difference in % of the resolution	
X 1	Y1	X2	Y2	X=X1-X2	Y=Y1-Y2	X/480*100%	Y/640*100%
240	337	295	395	-55	-58	-11.46	-9.063
211	349	270	411	-59	-62	-12.29	-9.688
265	345	327	405	-62	-60	-12.92	-9.375
198	305	251	358	-53	-53	-11.04	-8.281
241	368	291	430	-50	-62	-10.42	-9.688
238.1	320.2	290	382	-51.9	-61.8	-10.81	-9.656
240	332	300	397	-60	-65	-12.5	-10.16
344	328	408	391	-64	-63	-13.33	-9.844
195.6	238.8	246	291	-50.4	-52.2	-10.5	-8.156
237	397	287	461	-50	-64	-10.42	-10
232	333	290	387	-58	-54	-12.08	-8.438
236	345	301	402	-65	-57	-13.54	-8.906
342	341	414	396	-72	-55	-15	-8.594
189	246	246	290	-57	-44	-11.88	-6.875
232	410	286	474	-54	-64	-11.25	-10
206	356	254	401	-48	-45	-10	-7.031
167	366	222	414	-55	-48	-11.46	-7.5
211	364	275	410	-64	-46	-13.33	-7.188
162	333	209	369	-47	-36	-9.792	-5.625
207	372	251	428	-44	-56	-9.167	-8.75

Table 8. Landmark positions +calculations for the first 20 images in the MUCT database

Left eye		Nose tip		X/Y difference		Difference in % of the resolution	
X 1	Y1	X2	Y2	X=X1-X2	Y=Y1-Y2	X/720*100%	Y/1080*100%
319	639	417.07	724	-98.07	-85	-13.6208	-7.87
276	611	373	705	-97	-94	-13.4722	-8.704
255	633	369.36	715	-114.4	-82	-15.8833	-7.593
219	669	309	736	-90	-67	-12.5	-6.204
236	680	326	751	-90	-71	-12.5	-6.574
231	679	312.67	750	-81.67	-71	-11.3431	-6.574
356	546	454.04	642	-98.04	-96	-13.6167	-8.889
302	577	398	675	-96	-98	-13.3333	-9.074
282	579	371	650	-89	-71	-12.3611	-6.574

294	720	381	791	-87	-71	-12.0833	-6.574
284	719	375	788	-91	-69	-12.6389	-6.389
288	720	381	793	-93	-73	-12.9167	-6.759
307	421	407	495	-100	-74	-13.8889	-6.852
302	425	403	505	-101	-80	-14.0278	-7.407
291	423	384	499	-93	-76	-12.9167	-7.037
312	582	418	656	-106	-74	-14.7222	-6.852
318	612	430	688	-112	-76	-15.5556	-7.037
316	617	424.01	695	-108	-78	-15.0014	-7.222
251	495	356	594	-105	-99	-14.5833	-9.167
265	498	365	589	-100	-91	-13.8889	-8.426

Table 9. Our detected landmark positions + calculations for the first 20 images in the MUCT database

The average difference of horizontal and vertical position of the nose and left eye are calculated by taking the sum of all difference % value of the landmark detection and subtract the sum of our detected difference % value. The result is displayed in table 10.

Average difference X %	Average difference Y %
1.88	1.25

Table 10. Difference percentage x and y

Although this is a really abstract examination of the stability of our nose tip detection it seem that with an average of below 2% our nose tip detection is pretty accurate.

9.2. Angle calculation of the face

To calculate the horizontal angle of the face the nose tip is very useful. The length of the nose (seen from the cheek to the nose tip) is taken as an constant. The visible nose length is determined by how far the nose tip is located from the horizontal middle of the ***face***. In a frontal face image the nose tip is very close to the middle of the ***face*** which means the angle of the rotated face is near zero. When the face is turned from zero to thirty five degrees the visible nose length x will increase sinusoidal. This is visualized in figure 30 where the start of the nose is located as the bottom left red dot.

$$nose\ length = constant$$

$$x = visible\ noselength$$

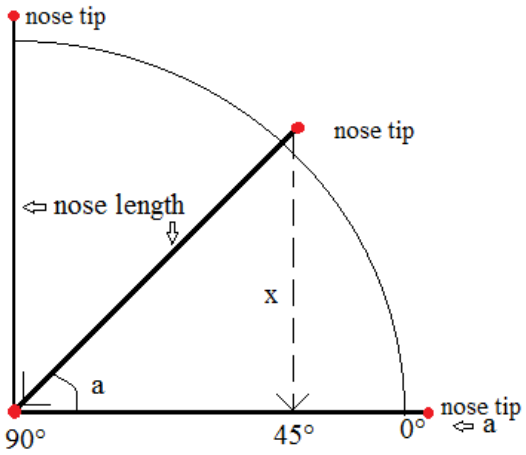

Figure 31. Visualisation of the visible nose and how to calculate its angle

$$\sin(a) = \frac{x}{nose\ length}$$

This means the angle of the rotated face is determined by $a = \arcsin\left(\frac{x}{noselength}\right)$.

Using OUR database again the angles of the test subjects will be determined using our

own angle calculation. Again our setup consisted of a phone aimed at a screen where images are displayed. Because of our setup it is not known how much influence it has on the values. For five subject their calculated angles are put in table 11.

	Subjects					
Angle	**1**	**2**	**3**	**4**	**5**	**Average**
35	28	66	34	33	40	40.2
30	29	60	35	28	35	37.4
25	23	53	24	21	32	30.6
20	18	25	22	19	24	21.6
15	14	17	17	16	20	16.8
10	10	9	9	10	9	9.4
5	4	4	8	0	6	4.4
0	2	1	8	0	2	2.6
-5	0	0	7	-2	-8	-0.6
-10	-1	-3	3	-7	-15	-4.6
-15	-5	-4	0	-10	-11	-6
-20	-11	-16	-3	-23	-14	-13.4
-25	-20	-24	-20	-22	-17	-20.6
-30	-28	-27	-21	0	-32	-21.6
-35	-33	-30	-22	-8	-40	-26.6

Table 11. Angle calculation of five subjects

The average angle of the five subjects are also calculated. When we look at each subject separately the calculated values of the angle can be miles of compared to given angle. This is explained by the fact that in every calculation average values are used. The different width of the subjects noses and the nose length influences the angle calculation heavily. However when we look at the average angle for the five subjects it matches the given angle way better. It is still nowhere near perfect and therefore the angle calculation will not be used to try improve the pose estimation

10.Conclusion

In this paper the development of a perfect selfie application is explained. Three different cascades for use in Viola-Jones detector have been introduced. The performance and stability of all three cascades have been investigated. It was found that the combined cascade gives the best tradeoff between performance stability.

When looking for a good pose estimation it stood out that the nose detector only shows an area in which the nose can be found. The nose tip could be in any position inside this area and therefore the nose detector alone is not stable enough. The theory was made that with the help of light intensity the nose tip could be detected. In experiments with the MUCT database it showed that detection of the nose tip is pretty solid.

When looking for a way to improve the pose estimation the idea was to use the nose tip to calculate the angle of which the face is rotated. Unfortunately this could not be implemented because the individual calculated angles deviated too much.

In the end the basics of a perfect selfie application is developed for an android environment which meets the established criteria but can still use some improvements.

11.References

[1] https://www.rijksoverheid.nl/onderwerpen/paspoort-en-identiteitskaart/inhoud/eisen-pasfoto-paspoort-id-kaart

[2] http://docs.opencv.org/2.4/doc/tutorials/introduction/android_binary_package/O4A_SDK.html

[3] https://developer.nvidia.com/codeworks-android

[4] S.-I Choi. Face Recognition Under Illumination Variation Using Shadow Compensation and Pixel Selection. International Journal of Advanced Robotic Systems. Volume 13. 2016. http://cdn.intechopen.com/pdfs-wm/40176.pdf

[5] P. Viola and M. Jones, Rapid object detection using a boosted cascade of simple features. CVPR, 2001. https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

[6] http://stackoverflow.com/questions/8791178/haar-cascades-vs-lbp-cascades-in-face-detection
[7] https://github.com/opencv/opencv/tree/master/data/haarcascades
[8] http://robotics.csie.ncku.edu.tw/Databases/FaceDetect_PoseEstimate.htm#Our_Database_
[9] S. Milborrow and J. Morkel and F. Nicolls. The MUCT Landmarked Face Database. Pattern Recognition Association of South Africa. 2010 . http://www.milbo.org/muct/
[10] http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/xm2vts/xm2vts_markup.html