**1. Instructions**

**1.1. Operating system installation on Raspberry Pi**

1) Operation system “Raspbian Stretch with desktop” was downloaded on laptop from <https://www.raspberrypi.org/downloads/raspbian/> in ZIP format and unzipped

2) MicroSD card was inserted in laptop through adapter

3) A graphical SD card writing tool was downloaded from <https://www.balena.io/etcher/> and installed

4) Using Etcher, image from ZIP file was copied to MicroSD

5) MicroSD was inserted in Raspberry Pi at the back side

6) Finally, Raspberry Pi was powered with 2.5A, 5V power supply and booted up

In case of any problems, more details for installation can be found here <https://www.techrepublic.com/article/how-to-set-up-your-raspberry-pi-3-model-b/>

**1.2. OpenCV installation on Raspberry Pi**

1) Filesystem was expanded to include all available space by “sudo raspi-config” command run in terminal

2) After reboot, existing packages were updated by “sudo apt-get update && sudo apt-get upgrade”

3) Then all necessary dependencies for images and video were installed by “sudo apt-get install”

4) Python was installed by “sudo apt-get install python3-dev”

5) OpenCV and OpenCV\_contrib were downloaded from official repositories

6) Numpy was installed by “sudo pip3 install numpy scipy”

7) OpenCV was compiled and built (careful with path to directories) with “cmake” and “make -j4”

8) Computer was rebooted and it was checked whether OpenCV works by “import cv2” and “cv2.\_\_version\_\_” run in Python

In case of any problems, more details for installation can be found here <https://www.alatortsev.com/2018/09/05/installing-opencv-3-4-3-on-raspberry-pi-3-b/>. For installation of OpenCV on laptop use <https://opencv.org/releases.html> and for Python https://www.python.org/downloads/.

**1.3. Training model**

1) Video with and without target was recorded on USB camera

2) 200 positive and 1000 negative sample frames were extracted from video by VLC player as described here <https://www.raymond.cc/blog/extract-video-frames-to-images-using-vlc-media-player/>.

3) Images were massively resized to 100x100 pixels with software downloaded from https://www.bricelam.net/ImageResizer/

4) Folders were created and labeled “p” and “n” respectively

5) Cascade Trainer GUI was downloaded from <http://amin-ahmadi.com/cascade-trainer-gui/> and installed

6) Path to folder with samples was given in Input section, all other parameters can be kept default, otherwise experiment with Number of stages for accuracy of model, Buffer size to make training process faster, Feature Type to choose suitable method for specific application

7) The training process was started and lasted for more than 10 hours, to give .xml output file

In case of any problems, more details on Cascade Trainer GUI can be found on <http://amin-ahmadi.com/cascade-trainer-gui/>. For more details about meaning of parameters used in this software please visit <https://docs.opencv.org/3.4/dc/d88/tutorial_traincascade.html>. For understanding theory behind the LBP and HAAR recognition methods, please refer to explanation prepared for in Section 3.2. and 3.3.

**1.4. Running the code**

1) Folder "dataset-test” was created to store pictures of target captured when it’s close to the center

2) “cascade-new.xml” file, which is HAAR trained model, was placed in the same folder where Python code is located

3) USB camera was connected to Raspberry Pi and put in “PC Camera” mode

4) Code was run in Python by clicking on logo of Raspberry, then Programming, then Python 3 (IDLE)

5) Target was detected and following commands given by programme, the center of the screen was reached

Explanation of code can be found in comments given after “#”. Tutorial on recognition can be found here <https://www.hackster.io/mjrobot/real-time-face-recognition-an-end-to-end-project-a10826> . Tutorial for threading can be found here <https://www.pyimagesearch.com/2015/12/28/increasing-raspberry-pi-fps-with-python-and-opencv/> .

**2. Materials, methods and results**

**2.1. Software**

Python is a programming language which among other things provides a powerful environment for learning and experimenting with Computer Vision and Machine Learning. It was chosen for this project because it is relatively easy to write and read the code making it friendly for beginners, has access to a huge number of libraries written for Python, is compatible with Raspberry PI, has big community and available tutorials with sample codes for OpenCV and lastly it is open source language which doesn’t require activation key or subscription. The disadvantage might be that Python is slower than C++ language and since OpenCV is written natively in C++, it would be more advantageous than Python.

**2.2. Hardware**



*Figure C –Hardware used in project (except headphones)*

*Source: own work*

Necessary hardware: Akaso Brave 4 camera, Raspberry Pi 3 Model B, Scutes Deluxe USB Power Bank, Raspberry Pi 7” Touchscreen Display, USB cable

For training the model, HP ProBook 6550b was deployed. It has Intel® Core™ i5-480M 2.67GHz processor and 4GB RAM.

Energy consumption calculations: 0.52A\*5V = 2.6W (Raspberry Pi power consumption when shooting 1080p video), 2.6Ah\*3.7V=9.62Wh (power bank energy capacity under nominal voltage), t=9.62Wh/2.6W=3.7h (time for use), finally 1h 51min considering safety factor of 2 is available.

**2.3. Measuring delay**



*Figure A – Measuring delay*

*Source: own work*

Connect camera and screen to Raspberry Pi. Start recording the screen with camera. Since until video is displayed on screen some time has passed, one should get a delay. To measure it, just put timer under the camera too. Thus, on final image, you can see real time captured by camera (without recognition process) and time when it appeared on the screen (with recognition). The difference between two is delay. With threading, around 0.5s delay is expected as shown on the picture.

**2.4. Video results of different methods**

<https://youtu.be/7XTZvwi2Tk4> delay measurements

<https://youtu.be/uVQTlMHmaKs> Haar-cascade classifier method (no threading)

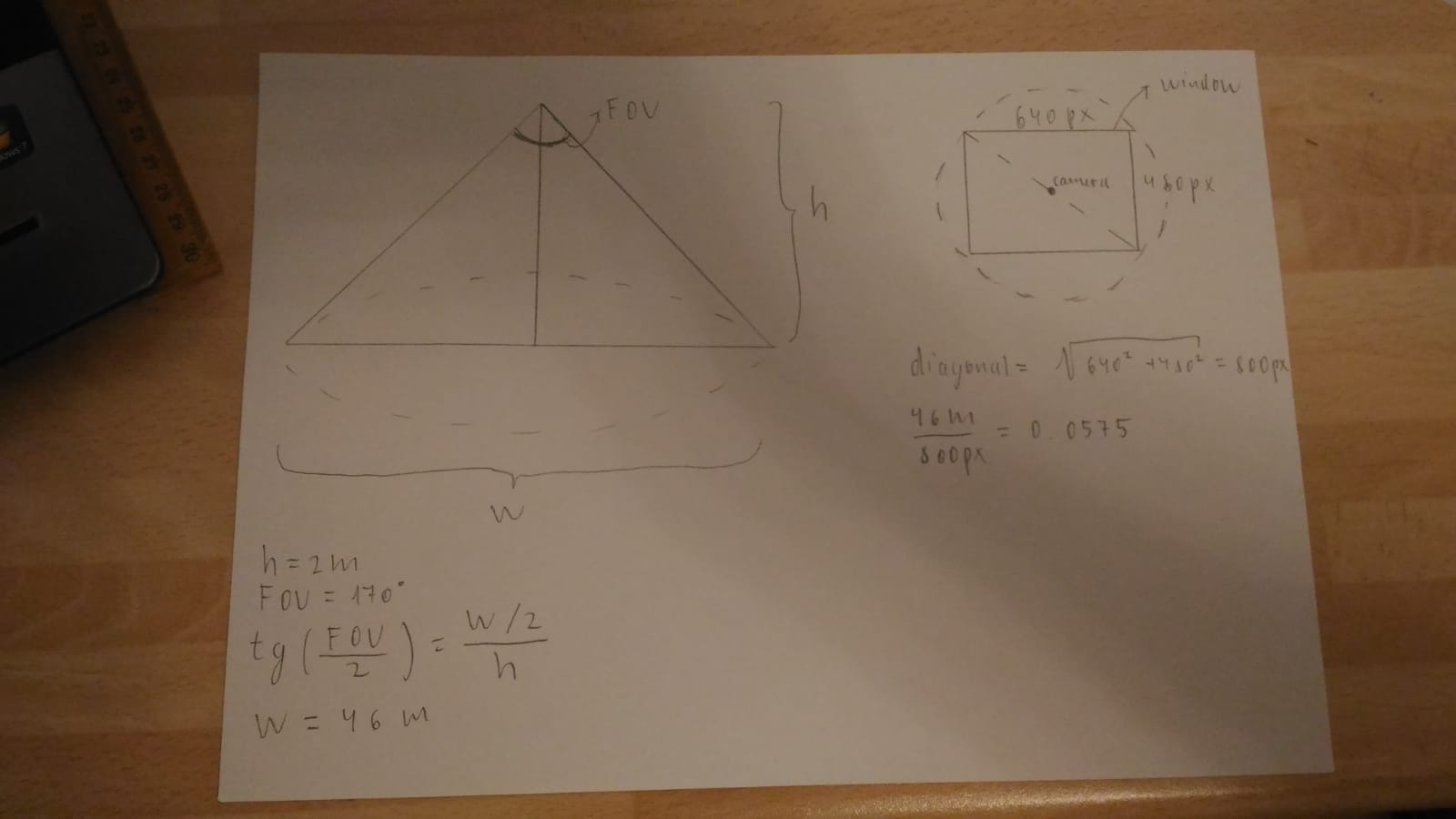
<https://youtu.be/bnXWr0IEIeM> Capture by PiCamera, processing on laptop

<https://youtu.be/NtAstoX5n4Q> Capture by PiCamera, processing on Raspberry Pi

<https://youtu.be/4vX8hrqULcE> Capture by USB camera, processing on laptop (VLC method)

**3. Some background and additional information**

**3.1. Coordinate system**



*Figure B – Pixel to meters conversion, geometrical model*

*Source: own work*

From angle and height, we can find width by using tangent function. Ratio of real width to width of screen in pixels gives us value of meters per pixel to convert between them. After target was detected, pixel coordinates are known and with this geometrical model, one can find values in real-world dimension.

**3.2. Haar feature-based cascade classifier**

To train model and recognize faces in video steam, machine learning based approach called “Haar feature-based cascade classifier” is being utilized. According to OpenCV (2018) it processes positive (picture containing object to be recognized) and negative (picture without the object of interest) sample images to extract “Haar features”, which are shown on Figure 1. To get features, detection window with specific shape from Figure 1 is moved over the image, being placed in all possible sizes and locations. Each feature is then represented by a value calculated by subtracting sum of pixels’ value (intensity) under the white area from sum of pixels’ value under the black area.

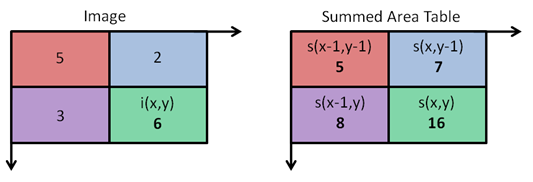


*Figure 1 - Haar-like features Figure 2 – Haar-like feature applied on human face*

*Source: OpenCV. 2018. Face Detection using Haar Cascades. OpenCV. Available at https://docs.opencv.org/3.4.1/d7/d8b/tutorial\_py\_face\_detection.html. Accessed on 25.01.2019.*

Figure 2 illustrates how features are extracted from the image. Basically, to recognize face, algorithm relies on the fact that area of cheeks is brighter than area of eyes. Likewise, area of eyes is darker than the bridge of the nose. These patterns are used in the process of object recognition afterwards. According to Viola and Jones (2001), to improve efficiency of Haar-feature extraction and application, thus ensure fast and accurate object recognition, the algorithm combines three important techniques: Integral image, Adaboost and Cascade of classifiers.

According to BADGERATI (2010), integral image is a calculated image representation of original image that allows very fast feature evaluation.



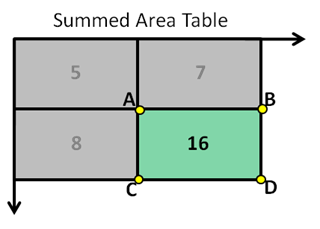
*Figure 3 - original matrix with pixel intensities Figure 4 – Summed Area Table*

*Source: BADGERATI. 2010. Computer Vision – The Integral Image. Computer Science Source. Available at https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/ . Accessed on 25.02.2019.*

Firstly, original image is converted to gray scale and intensities of each pixel (value from 0 to 255) are stored in the matrix as in Figure 3. Then Summed Area Table is calculated as shown in Figure 4 with the following formula:

sxyEq

For example, for lowest and rightest cell: S(x,y)=6+8+7-5=16. That’s the same result if we would use original image and summed pixel values above and to the left of original pixel value, which is relative simple calculation for this matrix size. After that, calculated Summed Area Table is used to effectively find area of a certain part of an image. For example, area of ABCD green region in Figure 5.



*Figure 5 – Summed Area Table with ABCD rectangle highlighted*

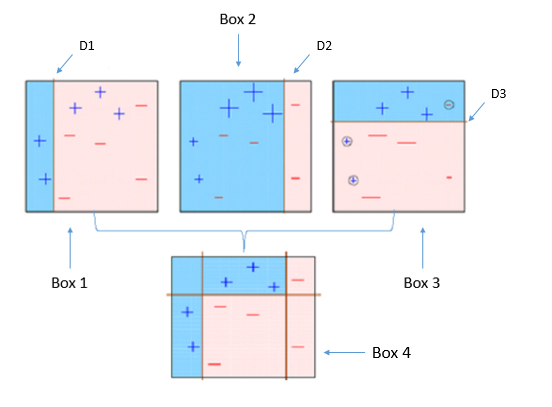
*Source: BADGERATI. 2010. Computer Vision – The Integral Image. Computer Science Source. Available at https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/ . Accessed on 25.02.2019*

To find the area under green region, the following formula is applied:

regionadding_2

For this example, S(x,y)=5+16-7-8=6. That’s the value we had in the same cell in original matrix. This was obvious in this example, but, as size of matrix increases, integral image allows to reduce number of calculation operations needed to calculate area for Haar-feature if Summed Area Table is calculated first.

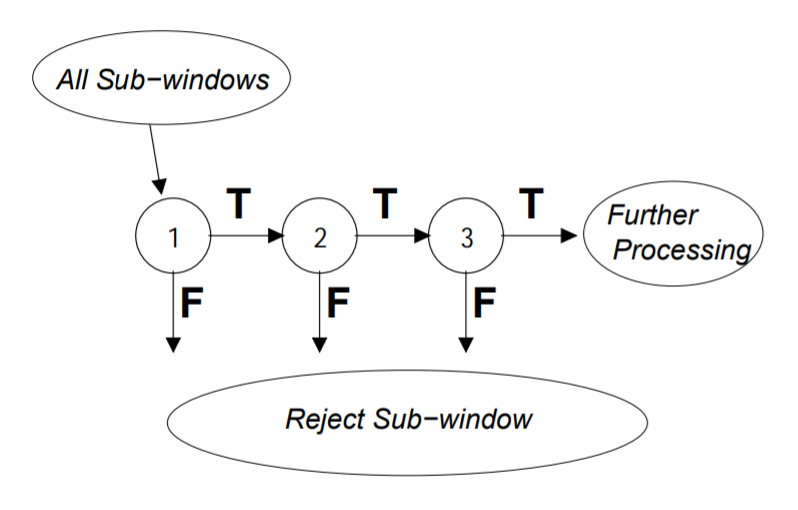
According to OpenCV (2018) Image with object contains a lot of irrelevant Haar-features which are not useful to classify the object thus have to be discarded. To select appropriate ones, all the features are tested on positive and negative samples in order to find the ones which describe the object best. This is done by looking at error rate and selecting features with minimum value for this parameter or in other words we keep the feature which classifies the face and non-face regions most accurately (and give it higher weight). The technique called Adaboost performs this function. Adaboost is a machine learning approach, based on the idea of creating a highly accurate prediction rule by combining many relatively weak and inaccurate rules.



*Figure 6 – Adaboost learning process*

*Source: S. Ray. 2015. Quick Introduction to Boosting Algorithms in Machine Learning. Analytics Vidhya. Available at* [*https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/*](https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/)*. Accessed on 25.01.2019.*

Figure 6 illustrates working principle of Boosting. According to Ray (2015), the algorithm generates some rule to classify – and + in Box 1 by drawing a vertical line. It is noticed that three + found in pink area are misclassified as –. In the next step, higher weight is given to these three + and a new line is generated as shown in Box 2. This time three – are found in blue area, indicating misclassification. So, in Box 3, horizontal line is drawn to classify them correctly, but generating new mistakes this time. The process is iterative. None of the rules alone can classify + and – on the image correctly. However, when these rules (week classifiers) are combined, a new rule is generated (strong classifier) that comprises successful attempts of all previous steps. Similar to example above, no single Haar-feature alone (weak classifier) can identify the object correctly. However, when several characteristic features (with corresponding weights) are combined they form strong classifier with significantly higher accuracy.



*Figure 7 – Working principle of Cascade of classifiers*

*Source: P. Viola, M. Jones. 2001. Rapid Object Detection using a Boosted Cascade of Simple Features. Computer Vision and Pattern Recognition. pp1-9.*

Furthermore, in an image captured by camera, face occupies a specific area and the rest is non-facial region or, in other words, background. Therefore, simple method should be used to distinguish regions containing face and background, so the latter could be immediately discarded and not being processed any further. Instead, the focus should be on regions potentially containing face. According to OpenCV (2018), to implement this idea, concept of Cascade of Classifiers is being utilized, which is a stepwise filtering approach, consisting of several simpler classifiers (strong classifiers from AdaBoost are used here) as shown on Figure 7. Features are grouped in so called “stages” of classifier and applied one by one. First stage analyses window very roughly (contains small number of features) and if found the features of the object in an image, it goes on with the next stage with higher number of features. If a window fails at any stage, it’s being discarded. If a window passes all the stages, then a face is being detected. Application of Cascade of Classifiers makes the process of recognition very efficient and fast, since it reduces computational workload by rejecting regions not containing face already at the early stages and just processing the rest.

**3.3. Local Binary Patterns Histograms**

According to Prado (2017), local Binary Patterns Histograms (LBPH) is a type of visual descriptors and one of the oldest and most popular face recognition algorithms.



*Figure 8 – Working principle of LBPH in steps*

*Source: K. Prado. 2017. Face Recognition: Understanding LBPH Algorithm. Towards Data Science. Available at* [*https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b*](https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b)*. Accessed on 25.01.2019.*

It uses “sliding window” approach to analyze the image. The idea is to create an intermediate image that describes the original object in a better way by highlighting its unique characteristics. As shown in Figure 8, a part of original image is converted to 3x3 grid with a value corresponding to intensity of each pixel (from 0 to 255). Then, central value is taken as threshold. New matrix is formed as a result of comparison between central and neighbor pixel values in the previous matrix. Binary numbers are combined line by line to form a new binary value. This will then be converted to a decimal value. The process is repeated in the same way for each pixel to get its decimal value. These values are then used to form histogram for the region and then combined in one complete histogram for whole image, which represents the characteristics of the original image. To compare two objects, algorithms extracts histograms first and then compares them. For example, using Euclidean distance, which can then be used as “confidence” measurement. The lower the value, the closer (similar) the two histograms are with 0 indicating exact match.

**4. Ideas and suggestions:**

-To stream video to ground station and process it there. With PiCamera it’s possible to achieve around 0.5s delay. Steps are described on <https://randomnerdtutorials.com/video-streaming-with-raspberry-pi-camera/>. For USB camera, streaming via VLC, delay was more than 10s. Steps are shown on <https://www.youtube.com/watch?v=JjPsW-7FUng> .

-To use camera with built-in Wi-Fi video transmission capability and process video on ground station. Thus, Raspberry Pi is not required at all.

- To try RPi 3 Model B+, which has more powerful processor than RPi 3 Model B (1.4 GHz instead of 1.2 GHz).

- To use Local Binary Patterns training model, which is faster to train, but mostly less reliable. Moreover, it’s possible to use combination of both LBP and HAAR. For example, one for detection (LBP) and one for recognition (HAAR).

- To address problem of detection at changing angles, two or more separate models can be trained effectively for specific angles and then used all together in combination.

- To train model with pictures of higher resolution. For this project, 100x100 size was used, because otherwise more time or more powerful machine is needed.

--Contour analysis recognition, Template matching recognition, Edge detection recognition work slow on Raspberry Pi, result in 10+ seconds delay and are unreliable at 2+ meters. Object tracking algorithms are not reliable once the target was lost from the screen and then reappeared. Thus, it’s better to focus on machine-learning based algorithms.

**5. References**

A. Rosebrock. 2015. Increasing webcam FPS with Python and OpenCV. pyimagesearch. Available at <https://www.pyimagesearch.com/2015/12/21/increasing-webcam-fps-with-python-and-opencv/>. Accessed on 09.02.2019.

Alex. n.d.. How Much Power Does Raspberry Pi3B Use? How Fast Is It Compared To Pi2B?. RasPi.TV. Available at https://raspi.tv/2016/how-much-power-does-raspberry-pi3b-use-how-fast-is-it-compared-to-pi2b. Accessed on 09.02.2019.

BADGERATI. 2010. Computer Vision – The Integral Image. Computer Science Source. Available at https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/ . Accessed on 09.02.2019.

BalenaEtcher. 2018. Flash. Flawless. balenaEtcher. Available at <https://www.balena.io/etcher/>. Accessed 09.02.2019.

ComStern. n.d.. Scutes Deluxe Powerbank 2600mAh mit Taschenlampe – schwarz. ComStern.de. Available at <https://www.comstern.de/Scutes-Deluxe-Powerbank-2600mAh-mit-Taschenlampe-schwarz-3a20372030.htm>. Accessed on 09.02.2019.

K. Prado. 2017. Face Recognition: Understanding LBPH Algorithm. Towards Data Science. Available at <https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b>. Accessed on 09.02.2019.

Mindfire Solutions. 2017. Advantages and Disadvantages of Python Programming Language. Medium. Available at <https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121>. Accessed on 09.02.2019.

MJRoBot. 2018. Real-Time Face Recognition: An End-to-End Project. hackster.io. Available at <https://www.hackster.io/mjrobot/real-time-face-recognition-an-end-to-end-project-a10826>. Accessed on 09.02.2019.

Nick Heath. 2018. How to set up your Raspberry Pi 3 Model B+. TechRepublic..Available at https://www.techrepublic.com/article/how-to-set-up-your-raspberry-pi-3-model-b/. Accessed on 09.02.2019.

OpenCV. 2018. Face Detection using Haar Cascades. OpenCV. Available at <https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html>. Accessed on 09.02.2019..

OpenCV. 2018. Open Source Computer Vision Library . GitHub. Available at <https://github.com/Itseez/opencv/archive/3.3.3.zip>. Accessed on 09.02.2019.

OpenCV. 2018. Open Source Computer Vision Library . GitHub. Available at <https://github.com/opencv/opencv_contrib/archive/3.4.3.zip>. Accessed on 09.02.2019.

OpenCV. 2018. Open Source Computer Vision Library . GitHub. Available at <https://github.com/opencv/opencv/tree/master/data/haarcascades>. Accessed on 09.02.2019.

OpenCV. 2019. About. OpenCV. Available at https://opencv.org/about.html. Accessed on 09.02.2019.

P. Viola, M. Jones. 2001. Rapid Object Detection using a Boosted Cascade of Simple Features. Computer Vision and Pattern Recognition. pp1-9.

R. Raja. 2017. FACE DETECTION USING OPENCV AND PYTHON: A BEGINNER’S GUIDE. SUPER DATASCIENCE. Available at <https://www.superdatascience.com/opencv-face-detection/>. Accessed on 09.02.2019.

RASPBERRY PI FOUNDATION. n.d.. Raspberry Pi 3 Model B. Raspberrypi. Available at <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Accessed on 09.02.2019.

RASPBERRY PI FOUNDATION. n.d.. Raspberry Pi 3 Model B. Raspberrypi. Available at <https://www.raspberrypi.org/downloads/>. Accessed on 09.02.2019.

S. Puttemans. n.d.. Cascade Classifier Training. OpenCV. Available at <https://docs.opencv.org/3.4.1/dc/d88/tutorial_traincascade.html>. Accessed on 09.02.2019.

S. Ray. 2015. Quick Introduction to Boosting Algorithms in Machine Learning. Analytics Vidhya. Available at <https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/>. Accessed on 09.02.2019.

Techopedia. n.d.. Computer Vision. techopedia. Available at https://www.techopedia.com/definition/32309/computer-vision. Accessed on 09.02.2019.

**6. Annex**

