ISSN 1846-6168 (Print), ISSN 1848-5588 (Online)
https://doi.org/10.31803/tg-20220321232047

Preliminary communication
Received: 2022-03-21, Accepted: 2022-10-17

Face Detection and Recognition Using Raspberry PI Computer

Mario Dubovečak, Emil Dumić, Andrija Bernik*

Abstract: This paper presents a face detection and recognition system utilizing a Raspberry Pi computer that is built on a predefined framework. The theoretical section of this article shows several techniques that can be used for face detection, including Haar cascades, Histograms of Oriented Gradients, Support Vector Machine and Deep Learning Methods. The paper also provides examples of some commonly used face recognition techniques, including Fisherfaces, Eigenfaces, Histogram of Local Binary Patterns, SIFT and SURF descriptor-based methods and Deep Learning Methods. The practical aspect of this paper demonstrates use of a Raspberry Pi computer, along with supplementary tools and software, to detect and recognize faces using a pre-defined dataset.

Keywords: face detection; face recognition; haar cascades; LBPH; Raspberry PI

1 INTRODUCTION

As new technologies emerged, so did the desire and need to create artificial intelligence, which would improve and simplify our daily lives [1]. All fields of business employ computer technologies to improve processes while also advancing those activities through the introduction of unique, new solutions. Artificial intelligence, of which computer vision is an essential factor, seeks to automate specific operations as closely as possible to human nature [2]. It has been used for some time to identify qualities that are both unique to each individual and part of the group, such as papillary lines, speech patterns, and other physical features, as well as to collect additional information for data analysis, such age, sex, personal habits, and the like. It is a well-known fact that these data are already employed in virtualization for market research, machine learning, creating target products, and surveillance with the aim of enhancing safety and data protection. One of the most common examples is unlocking smart phones using biometric information (fingerprint or facial recognition unlocking) [3]. It all began with this specific goal in mind and then moved to other mobile applications in mobile banking, security and data storage, the application procedure for e-services, and other related areas.

A variety of options are provided with computer vision, with a continuing focus on new software advancements that make use of the hardware at hand. The majority of those are freely accessible on the Internet and have the potential to be upgraded, expanded, and implemented. One example that offers a variety of functions from the computer vision section and supports several programming languages such as Python, Java, and C++, is the open source computer vision library (OpenCV) [4]. Additionally, it is designed to be compatible with a variety of operating systems, including Windows, Linux, iOS, and Android. This paper demonstrates the interaction of software and hardware in a shared unit, as well as publicly accessible tools and services that do not fall under the definition of "personal computers" but rather are modular devices that enable and support all essential recognition functions.

2 ALGORITHMS FOR FACE DETECTION

The most important face detection techniques, including Haar cascades [5], histogram of oriented gradients (HOG) [6], support vector machine (SVM) [7] and deep learning methods [8] (which can be used for both object detection and recognition), will be discussed in the sections below.

2.1 Haar Cascades

Using the automatic object detection technology known as Haar cascades, objects can be instantly recognized in a video or image. The algorithm was developed by Michael Jones and Paul Viola in 2001. Although being an older algorithm, the Viola-Jones (or Haar cascades) technique proved to be an effective tool for real-time face detection (or generally object detection) [4]. The technique is based on four basic steps: calculating Haar features, creating integral image [9], AdaBoost training [10] and cascade classifiers [11].

2.1.1 Calculating Haar Features (Viola-Jones)

Human faces have inherent characteristics that distinguish one face from another. These include the cheeks, nose, lips, and eyes. The regions that can be utilized to train algorithms to find faces using rectangular regions differ significantly from one another. The calculation requires computing the differences between the sums after adding the pixel intensities in each location.

Three different features are used by Viola-Jones: The two-rectangle feature is a difference in the sum of the pixels contained in two rectangular sections (Figs. 1a, 1b). The three-rectangle feature (Fig. 1c) calculates the sum in a centre rectangle minus the sum in two outer rectangles. The diagonal difference between pairs of rectangles is computed using the four-rectangle feature (Fig. 1d).

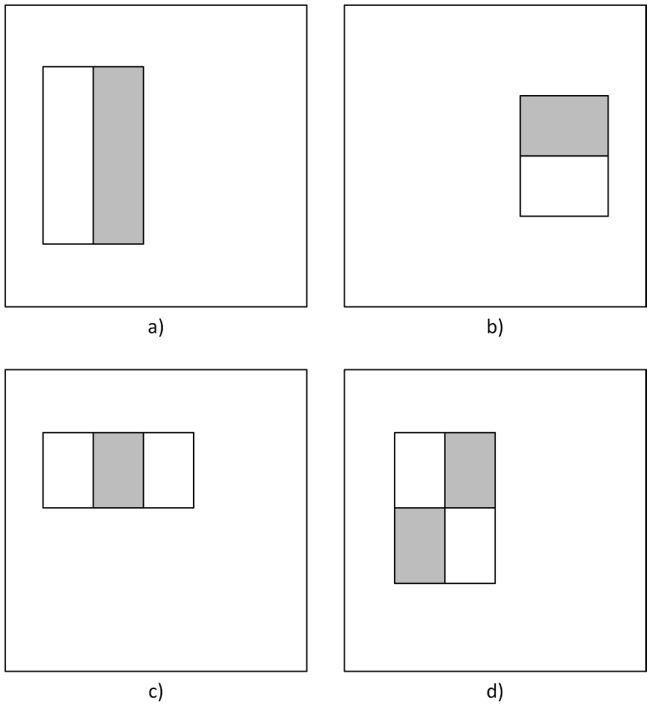

Figure 1 Four different types of features used in the frame

2.1.2 Creating Integral Image

Using an indirect image representation known as an integral image [9] or a summed-area table [12], rectangle features can be quickly determined. In fact, the integral image is a double integral of the image (first along the rows and then along the columns). Fig. 2a shows an integral picture at position x, y, where $ii(x, y)$ is an integral image and $i(x, y)$ is the original image. Eq. (1) states that the integral image at position x, y is the sum of the pixel values above and to the left of x, y.

$$ii(x, y) = \sum_{x' \le x,\, y' \le y} i(x', y') \tag{1}$$

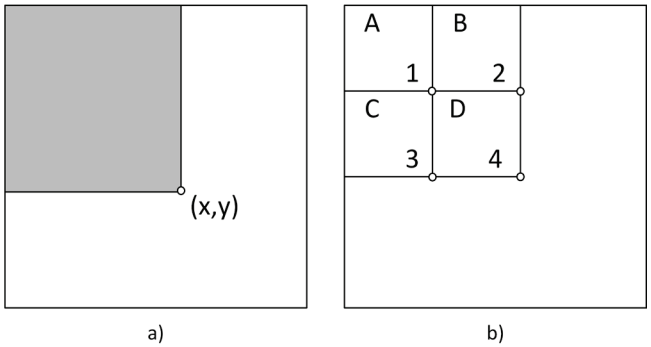

Figure 2 a) Integral image value at point (x, y); b) Example of four rectangle features

As an example, integral image calculation at position x, y is shown in Fig. 2a, as a sum of all pixels in the gray rectangle. The total value of rectangle D can be calculated based on the locations shown in Fig. 2b. Area A can be calculated by summing the pixels in image up to point 1 ($Sum_A = 1$ in integral image). Area B can be calculated by subtracting all the pixels in image within the space of point 2 and point 1 ($Sum_B = 2 - 1$ in integral image). Area C can be calculated by subtracting all the pixels in image within the space of point 3 and point 1 ($Sum_C = 3 - 1$ in integral image). Area D can be calculated by subtracting all pixels in image within the space of points 4 and 1, 2 and 3 ($Sum_D = 4 - Sum_A - Sum_B - Sum_C = 4 - 1 - (2 - 1) - (3 - 1) = 4 - 2 - 3 + 1$ in integral image).

Concluding, each rectangular sum may be computed using the integral images with four numbers, and as a result, the difference between two rectangular sums can be computed using eight numbers. It requires six numbers to calculate the difference between two adjacent rectangle features (Figs. 1a, 1b), eight numbers to compute three rectangle feature example (Fig. 1c) and nine numbers to compute the four-rectangle feature example (Fig. 1d).

2.1.3 Adaptive Boosting (AdaBoost)

Yoav Freund and Robert Schapiro [10] created the machine learning meta-algorithm AdaBoost, which may be used with a variety of other learning algorithms to increase performance. The results of the other learning algorithms are merged to create a weighted sum, which represents the boosted classifier's final results.

The algorithm becomes more accurate by learning from the images it receives as input, which might define false positives and true negatives. A very accurate model might be created by looking at every possible position and combination. Due to the numerous possibilities and combinations that must be tested for each scene or image, such training can be time-consuming.

2.1.4 Cascade Classifiers

Cascade classifiers are made of stages, each of which is an ensemble of weak learners [11]. Each stage is trained with the help of the boosting method. By using a weighted average of the decisions made by the weak learners, boosting makes it possible to train a classifier that is highly accurate.

Each classifier stage assigns a positive or negative label to the area that is specified by the sliding window's current position. Positive indicates an object was detected in a window, whereas negative means the object was not detected in the window. If the label is negative, the classification of that region is complete, and the detector then moves on to the first location afterwards. If not, the classifier with a positive label passes the scanned area for additional processing. The detector analyzes the selected area and provides information about the discovered object. It takes a lot of time to find genuine positive results because the great majority of windows do not contain the object of interest.

The more samples that help the recognition process successful, the better the overall result for indicating and rejecting false positives will be, which will be used as the basis for the judgment. For this, a tool for image labeling that identifies an object of interest and generates a table of positive samples can be used. To obtain acceptable detector accuracy, it is also necessary to offer a set of negative images

from which the function can automatically generate negative samples.

When calculating performance metrics for object detection, usually three sample patterns are used: A true positive, TP – when a positive sample is correctly classified; A false positive, FP – when a negative sample is mistakenly classified as positive; A alse negative, FN – when a positive sample is mistakenly classified as negative. A true negative, TN – when a negative sample is correctly classified as negative, is usually discarded in performance metrics for object detection due to the higher number of TN samples.

2.2 Histogram of Oriented Gradients (HOG)

In computer vision and image processing, a feature descriptor HOG can be used for object detection. The inventor of the HOG concept was Robert K. McConnell. Navneet Dalal and Bill Triggs presented in 2005 their supplementary work on HOG descriptors concentrated on pedestrian detection in static images as well as person detection in videos [6].

Given the enormous variety of appearances and poses that humans might take, it is necessary to develop a collection of features that would make it possible to distinguish the human figure from the background. The HOG approach is comparable to Edge Orientation Histograms and SIFT descriptors, but it varies in that it is computed on a dense grid of evenly spaced cells and uses overlapping local contrast normalization for increased accuracy.

The basic classifier uses a linear SVM which results in simplicity and speed. The essential idea behind the HOG descriptor is that the object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The image is divided into small connected regions called cells, and for the pixels within each cell a HOG is compiled. This type of histogram concatenation is called a descriptor. For improved accuracy, the local histograms can be contrast-normalized by computing a measure of the intensity across a larger region of the image (a block) which is then used to normalize all cells within the block. HOG operates on local cells, it is invariant to geometric and photometric transformations, with the exception of object orientation.

A linear SVM is used in the basic classifier, which results in speed and simplicity. The central part of the HOG descriptor is that the distribution of intensity gradients or edge directions can be used to describe the appearance and shape of objects inside an image. A HOG is compiled for each pixel contained in each of the image's small, interconnected cells. A descriptor is a histogram concatenation from all cells. By obtaining a measure of the intensity across a larger area of the image (a block), which is then used to normalize all cells inside the block, the local histograms can be contrast-normalized for increased accuracy. With the exception of object orientation, HOG operates on local cells and is invariant to geometric and photometric transformations.

2.3 Support Vector Machines (SVM)

SVM is used as a classification method to create the class border [7]. One-class SVM is used to discover anomalies in the normal class, and all points outside of margins are considered anomalous. SVM belongs to the class of linked algorithms for supervised machine learning that combine data analysis and pattern recognition to analyze data for classification or regression. The classifier's structure and properties determine the methods.

Liner SVM is the best-known classifier which predicts each input's member class between two possible classifications. SMV builds a hyperplane or set of hyperplanes to classify all inputs in a high-dimensional or infinite space. The closest values to the classification margin are known as support vectors whose goal is to maximize the margin between the hyperplane and the support vectors. Optimal separation can be achieved when the hyperplane is at the maximum distance from any class data point (functional margin). The following rule applies: the larger the distance the smaller is the classifier's generalization error. Support vectors are in fact data points that lie at the edge of the plane closest to the hyperplane that separates them.

The most well-known classifier, liner SVM, predicts the member class of each input between two alternative classes. For the purpose of categorizing all inputs in an infinite or high-dimensional space, SVM constructs a hyperplane or group of hyperplanes. Support vectors are the values that are closest to the classification margin and whose objective is to maximize the margin between the hyperplane and the support vectors. The hyperplane can be at its farthest position from any class data point to ensure optimal separation (functional margin). The following rule applies: the generalization error of the classifier decreases with increasing distance. In fact, the data points at the edge of the plane closest to the hyperplane separating them are the support vectors.

A training dataset of n points in the form $(\vec{x}_i, Y_1), \ldots, (\vec{x}_n, Y_n)$ is given, where the Y_i are either 1 or –1 depending on the class to which $\vec{x}_i$ belongs. Each $\vec{x}_i$ is a p-dimensional vector. The objective of a linear SVM is to find the maximum-margin hyperplane that separates the group of points $\vec{x}_i$ for $Y_i = 1$ from the group of points for $Y_i = -1$, where the distance between the hyperplane and the closest point $\vec{x}_i$ is maximal.

3 DEEP LEARNING MODELS

The detection of objects, faces, human activity, human poses, and data sets can all be done using deep learning algorithms [8]. Deep learning techniques could be divided into: Convolutional neural networks, Deep Belief Networks (DBN), Deep Boltzmann Machines (DBM), as well as stacked autoencoders (SAE).

3.1 Convolutional Neural Network (CNN)

The convolutional neural network (CNN) typically has three types of layers: the convolutional layer, the pooling layer, and the fully connected layer. Each layer has its own role in detecting input data and converting it into neural activation. In the end it leads to fully connected layers which

results in copying the input data into the vector feature (output). CNN is widely used in the areas of face detection and recognition, machine navigation and pattern recognition, the automotive industry, and the like.

3.2 Deep Belief Network (DBN) and Deep Boltzmann Machines (DBM)

DBN as well as DBM are both deep neural networks. They use the restricted Boltzmann machine (RBM), a generative stochastic neural network, as a learning module. Restricted Boltzmann Machine is s variant of Boltzmann machine, with the restriction that the neurons must form a bipartite graph. DBNs have undirected connections at the top two layers made of restricted Boltzmann machines and directed connections to the lower layers (Fig. 3a). DBMs connect all network layers using undirected connections (Fig. 3b).

By stacking the RBMs, DBNs are models that guarantee joint probability distribution on the observed data and labels. At the beginning, DBNs use training layer-by-layer to calculate initial weights, and afterwards fine-tuning is being performed over all weights to calculate the correct outputs. DBNs are graphical models that can be trained with the aim of extracting a deep hierarchical representation of the training data. However, there is a significant disadvantage associated with DBN, namely that the two-dimensional structures of the input image are not taken into account, which can be related with the speed and applicability of the algorithm itself.

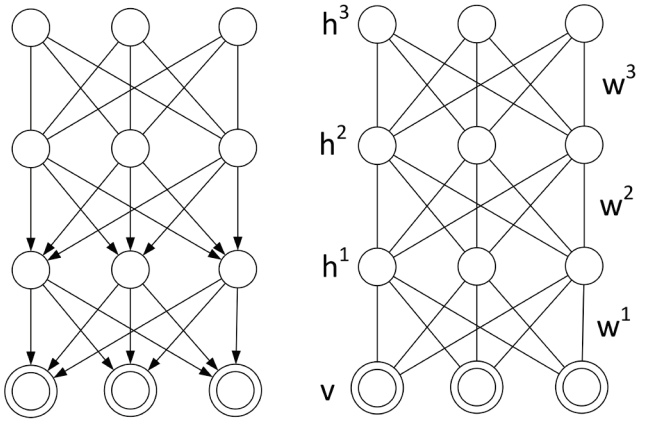

Figure 3 Schematic of the: a) DBN and b) DBM

3.3 Stacked Autoencoders (SAE)

Autoencoder is the the primary component of stacked autoencoders (SAE). The objective of an autoencoder is to encode the input $\boldsymbol{x}$ into reproduction $\boldsymbol{r}(\boldsymbol{x})$, which implies that the input is actually the target output. The reconstruction error is minimized in this process.

When training the network with one hidden linear layer using mean squared error (MSE) as a loss function, k hidden units project the input to first k principal components of the input data, similarly to the principal component analysis (PCA). The autoencoder does not operate as PCA if the hidden layer is nonlinear.

For example, in denoising autoencoders the input is "randomly" corrupted while the original input is used for target output reconstruction. The learning process begins by setting some inputs to zero after which the autoencoder tries to predict the corrupted values in relation to the uncorrupted ones. Two important steps in this process are:

- Autoencoder aims to encode the input by remembering the input data;
- Autoencoder aims to cancel the influence of random corruption applied to the input.

A deep network can be formed as a SAE by feeding the lower layer output features as an input to the next layer. Unsupervised pretraining is performed layer-by-layer, where each layer is trained as an autoencoder that minimizes the MSE by input reconstruction. Afterwards a logistic regression layer is added after each layer and then fine-tuning of the stacked encoding parts is performed, in a supervised manner.

A comparison of features of the mentioned models is presented in Tab. 1 [8].

Table 1 Comparison between different deep learning-based methods

Model features	CNN	DBN/DBM	SAE
Unsupervised learning	-	+	+
Training effectiveness	-	-	+
Learning features	+	-	-
Scaling, rotation, translation	+	-	-
Generalization	+	+	+

4 ALGORITHMS FOR FACE RECOGNITION

The output of facial recognition system is identification or verification of a subject or subjects that appear on an image or in a video.

In the Eigenfaces method, image is used as an input and the name of the individual is the output [13]. Eigenfaces method can be slower for larger datasets and larger dimensions of input images. Reducing the large images is required to accomplish effective performance. An image of dimension $m \times n$ is in fact $(m \times n) \times 1$ vector. Principal component analysis is one method for reducing dimensionality (PCA), which is based on determining the hyper-surface onto which all maximally scattered points are projected. When using PCA, the $m \times n$ image is reduced to a smaller space, allowing for quick implementation and resistance to noise and change. Since the face in the input image is not always centered, a cascade classifier is used to separate the face from the background of the image.

Fisherfaces method teaches a specific class of matrix transformation insensitive to illumination changes and different facial expressions [14]. Performance of this method depends on the input data, and the reconstruction of the projected image is made similarly as in Eigenfaces method.

A very straightforward but efficient visual descriptor, which can be used for face recognition tasks, is called local binary patterns (LBP) [15]. It marks a boundary between

neighboring pixels, assigning a binary number as a result. The procedure involves dividing the test window into cells (for instance, 16×16 pixels for each cell), comparing each pixel with its 8 neighbors, and then moving in a circle around the cells (clockwise or counterclockwise). If the value of the center pixel is higher than the neighboring pixel's value, the value 0 is written; otherwise, the value 1 is written. This produces an 8-digit binary number, which is usually converted to a decimal value. The cell histogram computation of each number's frequency is shown as a 256-dimensional vector. To obtain LBPH (LBP Histograms), LBP image is divided into multiple grids. Histogram is calculated from each grid (each having the size of 256, representing intensity of 0-255 after LBP calculation), and finally all histograms are concatenated to represent the characteristics of the whole image. In the face recognition process, LBPH from tested face is compared with the trained LBPH dataset and face (or a person identity) is returned with the closest histogram.

SIFT (Scale-Invariant Feature Transform) approach provides a solution regarding image rotation, scaling, intensity, changes in viewpoint, and changes in lightning [16]. The SIFT algorithm has four steps. The scale-space peak selection is used in the first step to determine the position and keypoints' ratio, by using difference of Gaussians (DoG) filter. In the second stage low contrast features are discarded. The third step is to assign an orientation, to establish keypoint orientation on a gradient image. The final step computes a descriptor for the local image region for each keypoint based on the size and orientation of the image gradient in each image pattern point in the region centered on a keypoint. This produces a 3D histogram of the location and orientation of the gradient with a 4×4 pixel neighborhood and an 8-bin cell for each pattern, producing the 128-dimension keypoint descriptor, which can be used to find matching pairs in different images. SIFT descriptors can be also used for face recognition, for example as in the paper [17].

SURF (Speed Up Robust Features) method [18], built on a multi-scaled space, consists of three basic components: detection, description and matching. Feature detector is based on the Hessian matrix. The SURF method can operate in parallel for different scalings and approximates LoG with the box filter, using integral image (described earlier). In the description part, SURF uses wavelet responses in both the x- and y-directions to assign orientation. Dominant orientation is estimated by adding up all responses within a $\pi/3$ sliding orientation window. Most applications do not require orientation, which further speeds up the procedure. This version of SURF method is called U-SURF ("upright" SURF). Finally, descriptors from different images can be compared to find matching pairs. Matching can be done only based on the contrast of the feature attributes, which additionaly speeds up the process. When compared with SIFT, SURF method has the same performance, but with the reduced computational complexity. SURF descriptors can be also used for face recognition, for example as in the paper [19].

For facial recognition, the FaceNet method can be also used [20]. In essence, it is a neural network that teaches how to map facial images into a compact space where distances represent the similarity of faces. The so-called triplet loss method is used to calculate the loss function. On the one hand, it increases the distance between an anchor and a negative (different identity), while minimising the distance between an anchor and a positive (same identity). Other deep learning models include DeepFace [21], VGGFace/VGGFace2 [22], FaceID-GAN [23] etc.

5 SETUP, TRAINING AND TESTING ENVIRONMENT

This paper uses the latest version of Raspberry Pi 4B device with all the required interconnected parts. VNC (Virtual Network Computing) remote control is used for visualization. A personal computer connected to joint network can be used for this purpose. Once a successful configuration of the device is performed, the installation of all necessary software such as OpenCV and other packages follows. The Viola-Jones algorithm was used as the algorithm for face detection, while the LBPH algorithm was used for face recognition [24]. Those algorithms were used to be able to run the algorithm on the Raspberry Pi device, which has limited hardware recources.

The main features of this program are placed in two files, one for testing and the other for recognition. The test file takes images located in Datasets directory's maps, indexes folder names (person names), and creates a *training.yml* file with all the necessary values that is used for comparison using the face recognition program. The file named *haarcascade_frontalface_default.xml* contains cascading classifiers associated with both files.

Program creates the file *oznaketrening.yml* from the images saved in the Datasets directory, based on which the data from the webcam video is later compared.

After launching a training program used to create and load values from all images from the corresponding folders and after comparing those same images with the video submitted from the webcam, the person's face in the frame is recognized and if it matches the one in the database then the name is printed in the recognition frame. It also sends a signal to the output of the Raspberry Pi device by turning off the red light on the LED module and turning on the green in a cycle time of 0.75 seconds (or 1.34 Hz). For example, this can be used as a signal to open the door.

Image dataset for 5 different people was created for the purpose of this paper. Characteristics of Raspberry Pi 4B devices were taken into account and dataset of 100 images (RGB format, dimensions 640×480 pixels with *.png* extension and in real/daily conditions) was made for each person, a total of 500 images. PNG format was used as a lossless compression, to be able to further process all images without adding additional artefacts. Higher-dimension images might be also used, but this resulted in additional heating of the Raspberry Pi device (up to 79° C without using additional coolers).

Half images per person were used for the training and the rest for testing purpose. Training was performed by running

Face_Trener.py file from the prepared dataset in order to train our program with the created dataset. The next step was to copy the entire dataset of images (250 in total) from which a slideshow was made (video file with *.avi* extension) for each of the 5 people. The settings for each image lasted 1 second, with an image editing transition of half a second, which resulted in processed 0.67 fps. Accordingly, separate video files do not overload the device. Each of the corresponding files is then loaded separately into the face recognition program using *Face_Prepoz.py*. In order to change the input to a video file it is necessary to change according to *cam=cv2.VideoCapture('Name_video_file.avi')* where the name of the file we want to load with the extension is specified. Loading these video files through the recognition system resulted in the expected result where faces were recognized in each of the images, although there were occasional inaccuracies in the comparison of the test images with the predefined ones.

Performance metrics for face recognition task can be calculated according to the following formulas, similarly as for the classification, using true positive (*TP*), false positives (*FP*), false negatives (*FN*) and true negative (*TN*) samples as defined in section 2.1.4.

Overall accuracy, Eq. (2):

$$ACC_{\text{overall}} = \frac{\sum(correct)}{\sum(overall)} = \frac{\sum(main\_diagonal)}{\sum(overall)} \tag{2}$$

Error rate, Eq. (3):

$$ERR_{\text{overall}} = 1 - ACC_{\text{overall}} = \frac{\sum(false)}{\sum(overall)} \tag{3}$$

Precision (Positive predictive value, *PPV*), Eq. (4):

$$PPV = \frac{TP}{TP + FP} \tag{4}$$

False discovery rate, *FDR*, Eq. (5):

$$FDR = 1 - PPV = \frac{FP}{TP + FP} \tag{5}$$

Recall (sensitivity or True positive rate, *TPR*), Eq. (6):

$$TPR = \frac{TP}{TP + FN} \tag{6}$$

False negative rate, *FNR*, Eq. (7):

$$FNR = 1 - TPR = \frac{FN}{TP + FN} \tag{7}$$

From the Fig. 4, we can see that the overall accuracy of the proposed algorithm is 95.2% (last row on the right, green number). Precision is given for each person in the last column on the right (green number), while recall is in the last row (green number). Opposite results (1-green number) are given as red numbers: Error rate, 1-(overall accuracy) is last row on the right, red number. False discovery rate (*FDR*) is given for each person in the last column on the right (red number), while false negative rate (*FNR*) is in the last row (red number).

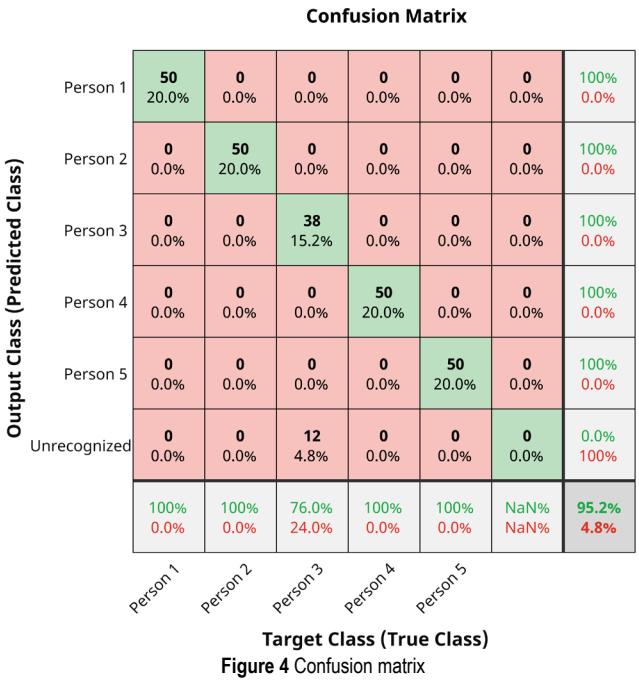

Figure 4 Confusion matrix

The results indicate that for the marked "Person 3" there is a total of 12 images who are not recognized, lowering its recall to 76%. In other words, the faces are detected on those images using Haar cascades, but they are not recognized compared to persons previously trained in the database – there were no matches. As a reason, it can be said that is was a woman with longer hair who in the unrecognized images had her head tilted to the side at an angle of approximately 45°.

6 CONCLUSIONS

In this paper, different algorithms for face detection and recognition are presented and discussed. In addition to the theoretical part, the procedure of preparing, installing and commissioning Raspberry Pi computer is described, together with all components, including software. The Haar cascades (Viola-Jones) were used as the algorithm for face detection, while LBPH algorithm was used for face recognition. Concerning the accuracy, it was noticed that the accuracy of face recognition can differ due to various parameters such as lighting, face rotation in all directions, distance from the camera as well as camera quality. System was tested to work with 0.67 fps processing time per 1 recognized identity.

Although higher-dimension images are more recognizable, they are also more demanding to handle. In

practice, this resulted in additional heating of the Raspberry Pi 4B device to an occasional 79° C without using additional coolers.

Due to its modularity, the Raspberry Pi device offers a foundation for further process in terms of controlling the entrance of selected persons through the door. The idea is to use the existing platform and upgrade it. In the future experiments, in addition to traffic control, network data collection (time, place, person) and remote control can be also enabled.

7 REFERENCES

[1] Lee, R. S. T. (2020). Artificial Intelligence in Daily Life. 1st Edition. Springer. https://doi.org/10.1007/978-981-15-7695-9

[2] Szeliski, R. (2022). *Computer Vision: Algorithms and Applications*. 2nd Edition. Springer. https://doi.org/10.1007/978-3-030-34372-9

[3] Unar, J. A., Seng, W. C., & Abbasi, A. (2014). A review of biometric technology along with trends and prospects. *Pattern recognition, 47*(8), 2673-2688. https://doi.org/10.1016/j.patcog.2014.01.016

[4] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools, 120*, 122-125.

[5] Viola, P. & Jones, M. (2001). Robust real-time object detection. *International journal of computer vision, 57*, 34-47. Springer.

[6] Dalal, N. & Triggs, B. (2005). Histograms of oriented gradients for human detection. *IEEE computer society conference on computer vision and pattern recognition (CVPR'05), 1*, 886-893. https://doi.org/10.1109/CVPR.2005.177

[7] Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine learning, 20*(3), 273-297. https://doi.org/10.1007/BF00994018

[8] Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience, 2018*, Article ID 7068349, 13 pages. https://doi.org/10.1155/2018/7068349

[9] Lewis, J. P. (1995). Fast template matching. *Vision Interface 95*, 120-123.

[10] Freund, Y. & Schapire, R. E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. *Lecture Notes in Computer Science*, 23-37. https://doi.org/10.1007/3-540-59119-2_166

[11] Gama, J. & Brazdil, P. (2000). Cascade Generalization. *Machine Learning, 41*(3), 315-343. https://doi.org/10.1023/A:1007652114878

[12] Crow, F. (1984). Summed-area tables for texture mapping. *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer Graphics and Interactive Techniques*, 207-212. https://doi.org/10.1145/800031.808600

[13] Sirovich, L. & Kirby, M. (1987). Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A, 4*(3), 519-524. https://doi.org/10.1364/JOSAA.4.000519

[14] Belhumeur, P. N., Hespanha, J. P., & Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on pattern analysis and machine intelligence, 19*(7), 711-720. https://doi.org/10.1109/34.598228

[15] Ojala, T., Pietikäinen, M., & Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern recognition, 29*(1), 51-59. https://doi.org/10.1016/0031-3203(95)00067-4

[16] Lowe, D. G. (1999). Object recognition from local scale-invariant features. *Proceedings of the seventh IEEE international conference on computer vision, 2*, 1150-1157. https://doi.org/10.1109/ICCV.1999.790410

[17] Geng, C. & Jiang, X. (2009). Face recognition using sift features. *The 16th IEEE International Conference on Image Processing (ICIP)*, 3313-3316. https://doi.org/10.1109/ICIP.2009.5413956

[18] Bay, H., Tuytelaars, T., & Gool, L. V. (2006). Surf: Speeded up robust features. *European conference on computer vision*, 404-417. https://doi.org/10.1007/11744023_32

[19] Du, G., Su, F., & Cai, A. (2009). Face recognition using SURF features. Proc. *SPIE 7496, MIPPR 2009: Pattern Recognition and Computer Vision*, 749628. https://doi.org/10.1117/12.832636

[20] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 815-823. https://doi.org/10.1109/CVPR.2015.7298682

[21] Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification. *IEEE Conference on Computer Vision and Pattern Recognition*, 1701-1708. https://doi.org/10.1109/CVPR.2014.220

[22] Cao, Q., Shen, L., Xie, W., Omka Parkhi, M., & Zisserman, A. (2018). VGGFace2: A Dataset for Recognising Faces across Pose and Age. *The 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, 67-74. https://doi.org/10.1109/FG.2018.00020

[23] Shen, Y., Luo, P., Luo, P., Yan, J., Wang, X., & Tang, X., (2018). FaceID-GAN: Learning a Symmetry Three-Player GAN for Identity-Preserving Face Synthesis. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 821-830. https://doi.org/10.1109/CVPR.2018.00092

[24] Dubovecak, M. Face detection and recognition using Raspberry PI. (2020). *Master's thesis*. https://zir.nsk.hr/en/islandora/object/unin%3A3197, Accesed: 6.10.2022

Authors' contacts:

Mario Dubovečak
University North,
104. brigade 1, 42000 Varaždin, Croatia
madubovecak@unin.hr

Emil Dumić, PhD, Associate Professor
University North,
104. brigade 1, 42000 Varaždin, Croatia
edumic@unin.hr

Andrija Bernik, PhD, Assistant Professor
(Corresponding author)
University North,
104. brigade 1, 42000 Varaždin, Croatia
andrija.bernik@unin.hr