

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Rait Kurg 178187

**AUTOMATED FACE MANAGEMENT
SYSTEM ON THE EXAMPLE OF SMART
ELEVATOR**

Master's thesis

Supervisor: Mairo Leier
Ph.D

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Rait Kurg 178187

AUTOMAATNE NÄGUDE HALDUSSÜSTEEM TARGA LIFTI NÄITEL

Magistritöö

Juhendaja: Mairo Leier
Ph.D

Tallinn 2019

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Rait Kurg

27.04.2019

Abstract

In this paper, an automated face management system algorithm is proposed for the Tallinn University of Technology Smart Elevator project, that is developed together with KONE Incorporate, one of the leading elevator construction companies in North Europe. The outcomes of this work is as a mechanism that can be used for memorizing and predicting destination floors for visiting people of ICT building elevator.

It's main core relies on the efficient use of convolutional neural networks, clustering and classification techniques, that is integrated into a traditional face recognition pipeline achieving 98.2% identification accuracy based on the especially recorded validation set. The designed system also includes modern user interface and integration options, which are designed and development specially respecting the individual properties of the system itself and its working environment.

This thesis is written in English and is 55 pages long, including 7 chapters, 27 figures and 5 tables.

Annotatsioon

Automaatne nägude haldussüsteem targa lifti näitel

Käesolevas magistritöös on loodud automaatne nägude haldussüsteem Tallinna Tehnikaülikooli Targa Lifti projekti jaoks, mis arendati koostöös AS KONE-ga, Põhja-Euroopa ühe suurima liftide ja eskalaatorite tootjaga. Tehtud töö tulemusena valmis lahendus, mida saab kasutada ICT maja lifti külastatavate inimeste tundma õppimiseks ja sellele tuginedes soovitud sihtkorruste ennustamiseks.

Algoritmi tuumik põhineb konvolutsiooniliste närvivõrkude, klasterdamistehnikate ja klassifitseerimisalgoritmide efektiivsele koostööle, mis on integreeritud traditsioonilisse näotuvastuskonveieri mudelisse. Loodud süsteem saavutas ennustustäpsuse 98.2%, mille treening- ja valideerimisandmed pärinesid salvestatud videokaadritest liftis. Lisaks on süsteemile loodud veebipõhine kasutajaliides ning liidestusmoodulid, mille arendamisel on arvestatud süsteemi olemust ja selle töökeskkonda.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 55 leheküljel, 7 peatükki, 27 joonist, 5 tabelit.

List of abbreviations and terms

RGB	Red, Green, Blue color system
USB	Universal Serial Port
REST	Representational State Transfer
API	Application Programming Interface
ROS	Robot Operating System
SDK	Software Development Kit
MVP	Minimum Viable Product
ID	Identifier
AFIS	Automated Fingerprint Identification System
ICT	IT building of Tallinn University of Technology
GDPR	General Data Protection Regulation
IO	Input/Output
GPU	Graphics Processing Unit
CUDA	GPU programming platform made by NVIDIA
NN	Neural network
RSA	Rivest–Shamir–Adleman, public key cryptosystem
POC	Proof of Concept
Dlib	Software toolkit composed by David King
UI	User Interface
LWF	Labeled Faces In Wild dataset
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
CNN	Convolutional Neural Network
SVM	Support Vector Machine
CPU	Central Processing Unit
RFID	Radio-Frequency Identification
HTTP	Hypertext Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
TalTech	Short version of Tallinn University of Technology
RFC	Request for Comment
GigE	Gigabyte Ethernet

Table of contents

1 Introduction	11
1.1 Motivation	11
1.2 Objectives	12
1.3 Thesis Organization	13
2 Background.....	15
2.1 Smart Elevator Project.....	15
2.2 Commercial products.....	16
2.3 Image processing	17
2.3.1 RGB frame.....	17
2.3.2 Neural networks.....	18
2.3.3 Convolutional neural networks.....	19
2.3.4 Training	22
3 System Design	23
3.1 Hardware selection	23
3.1.1 Camera positioning.....	24
3.2 Software selection.....	25
3.2.1 Face representation	26
3.2.2 Clustering	29
3.2.3 Performance comparison of clustering algorithms.....	31
3.2.4 Prediction.....	33
4 Automated face management system	35
4.1 Proposed algorithm.....	35
4.2 System overview.....	36
4.3 Experimental results	38
5 Integration and Interfaces	40
5.1 Communication with other system components.....	40
5.1.1 ROS	40
5.1.2 Interactions	41
5.2 User interface.....	43

5.2.1 Front end.....	43
5.2.2 Back end	46
6 Future development and suggested improvements.....	50
7 Conclusions	51
References	52
Appendix 1 – Software repository.....	55

List of figures

Figure 1. Traditional vs self-managed face recognition system.....	12
Figure 2. Biometric verification methods market share in 2017 [3].	15
Figure 3. RGB image representation [9].	17
Figure 4. Neuron and artificial neuron [10].....	18
Figure 5. Classical multilayer neural network [10].	19
Figure 6. Convolutional layer filter work principle [10].	20
Figure 7. ReLU and Leaky ReLU/PReLU [10].....	21
Figure 8. Max pooling [10].....	21
Figure 9. Face recognition pipeline [17].	26
Figure 10. Code block describing the Chinese Whispers clustering algorithm [28].	29
Figure 11. Code block describing the DBSCAN algorithm [29].	30
Figure 12. Code block describing the Mean-Shift clustering algorithms [31].	30
Figure 13. Code block for comparing different feature extractions with clustering algorithms.	31
Figure 14. Clustering accuracy change over the dataset size.	32
Figure 15. Computing power required over dataset size	32
Figure 16. Radius Neighbor classifier accuracy change over number of training images and radius.....	33
Figure 17. SVM accuracy over the number of training images.....	34
Figure 18. Saved clusters from elevator cycles.	35
Figure 19. Illustration of valid clusters.....	36
Figure 20. System layout with flows.	38
Figure 21. Test and training dataset for prediction.....	39
Figure 22. ROS.	41
Figure 23. Sequence diagram describing interactions between system components.	43
Figure 24. UI - face database.....	44
Figure 25. UI - timeline	45
Figure 26. UI - live overview of RGB camera and depth camera results.....	46
Figure 27. API architecture	47

List of tables

Table 1. Comparison of commercial facial recognition products [5] – [7].	16
Table 2. Comparison of different cameras [14] – [16].	23
Table 3. Comparison of considered camera positions.	24
Table 4. General overview of pretrained face recognition models [19] – [21].	28
Table 5. REST interface general overview.....	48

1 Introduction

One of the earliest facial recognition methods was proposed by Woodrow Bledsoe in 1966, which tried to compare the distances between unique landmarks on face. He also stated that the reason, that the problem is so difficult to solve due the great variability in head rotation and tilt, lightning intensity, different angles, facial expression and aging. After that, next several decades introduced several new improved methods of facial recognition, which tried to overcome the past results. One of the reasons that these methods did not made to wide usage was due to limitations of computer processing power. Finally, in early 2000s, the technology started to catch up with the proposed techniques and so the first major test was conducted by law enforcements during the 2002 Super Bowl. Although, reportable, there were many criminals detected by the software, the overall results were considered relatively bad, as the number of false positives was too high. During the following years, the sudden rise of social media made people to upload more photos of themselves to the Internet than ever and therefore made it is easy for the major providers like Facebook and Google to gather and sell enormous amount of data. This was a game changing opportunity, as previously those huge datasets of tagged photos had limited accessibility e.g. for government use only. After that, the doors started to open up for other new players in the market and even for the open source communities [1], [2].

1.1 Motivation

This thesis proposes a solution for the Smart Elevator project, providing the means how elevator could learn people through facial recognition. It introduces totally different approach from the traditional facial recognition systems, whose try to match known people from video feed. This means that known people database that is usually managed by administrator user will be replaced by totally automated face management system that overtime expands itself (Figure 1).

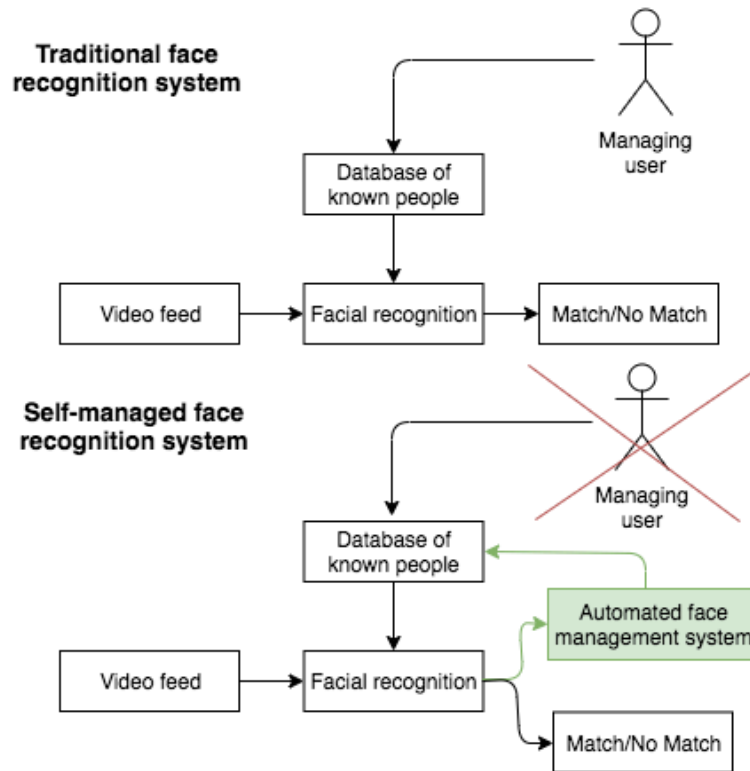


Figure 1. Traditional vs self-managed face recognition system.

1.2 Objectives

The Smart Elevator project was divided into two phases. For the first phase, main goal was to implement the proof of concept. During the second phase, the potential improvements for the system would be made. In order to validate the PoC (proof of concept) for the automated face management system, following goals were set:

- Develop system hardware and software architecture
- Select optimal hardware, open source software and algorithms
- Define methods, how software components are connected with each other
- System should be able to learn at least 100 people only from the videos recorded inside the elevator
- User interface developments to see the results (face database) and historical events

In order to demonstrate the results of the first phase, special demonstration flow was recorded and partly simulated as the all of the system components were not fully completed. The requirements for the demonstration were set by the project management

and covered the main principles needed for the stakeholders, proving the satisfaction criteria for the minimum viable product (MVP). The results of the tests were later placed on the user interface as mock of the live dashboard. This step helped to promote the system outside of the testing environment without worrying about the setup, connection issues and hidden bugs. Some parts of the validation use case like the elevator manipulation, people tracking and voice detection were not part of the author's work and are described just to give better overview of the system behavior.

The scenario starts with an empty elevator, which gets call to fifth floor. When the elevator has served the call and opens the doors, four people start to walk in one by one so that only one person is seen by the RGB (red, green, blue) camera. This limitation was set to avoid corner cases, which were meant to be solved during the next phase. The facial recognition software should be able to identify three persons that are predefined by the automated face management system and draw face boxes with their unique ID on the video stream. One person, that elevator has not "seen" before, meaning that this person has been not added to the face database earlier, should be considered as an unknown person. Aforementioned unknown person should have that referenced also on the video screen. During that time, the elevator waits for the voice commands and button pushes, which in this scenario will be request for third floor over the voice and request for second floor using the physical button. After the commands are received, the user interface should display elevator direction and first destination floor. Two random people are then stepping out and the elevator should start moving towards new direction. It is important that the user interface should decrease the value of the number of the people inside the elevator. Finally, when the elevator has reached its final destination and everyone is outside, the elevator will go to the idle state and save the results.

1.3 Thesis Organization

This thesis is composed from seven main chapters and includes one appendix. It starts with a introductive background information about the biometric verification technologies, the project itself and main principles of image processing techniques using convolutional neural networks. Then, the system design is introduced, which breaks down, compares and evaluates the software and hardware components used for the development and their role in the face recognition pipeline. After that, the solution for automated face learning

system is proposed together with its layout and how it can be installed together with other components with an overview of its performance based on the conducted experiments. Finally, the integration methods and technologies for external services are presented together with developed user interface and its main functionalities.

2 Background

Nowadays, different technologies of human face recognition with computer programs have shown more than just great success. Although the method itself is considered to have less accuracy than other biometric verification methods, the market is still expected to reach over 7.7 billion dollars by the 2022. This is because over the last years it has become more and more accurate and it can provide huge value to wide range of different applications from government street monitoring systems to consumer level robots for personal interaction purposes. Their main advantage over other biometric verification methods is an existing infrastructure. Main streets, office room and even some elevators are almost always equipped with security cameras, which can be easily integrated with facial recognition software. In 2017, it was one of the popular biometric verification methods used in Europe together with fingerprint identification methods (AFIS and Non-AFIS). Detailed illustrations of biometric verification methods used by market share during that time can be found on the following pie chart (Figure 2) [3], [4].

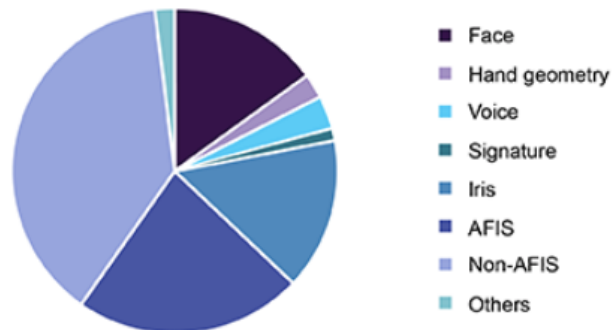


Figure 2. Biometric verification methods market share in 2017 [3].

2.1 Smart Elevator Project

The Smart Elevator project is part of the Tallinn University of Technology smart campus concept. It will be an elevator, which over time learn, which floors people visit and eventually will start to propose these floors to them. It will interact humans over voice commands and therefore will make elevator rides more comfortable than ever. The project itself is co-funded together with KONE Corporation, who is interested in making the elevators more intelligent and bringing people more emotions just by using their

elevators. The development team had 11 active members, including people from department of software science and system engineering.

2.2 Commercial products

Currently, there exist no such products on the market that could solve directly this concrete problem regarding the Smart Elevator, although, it is possible to implement the core features by combining different facial recognition APIs. Amazon, Microsoft and Kairos are only some of the companies that have the tools necessary to build such system. Although they all are in general really good products, the major downside of using them is that it leaves system depending on external service provider or network quality. Another limitation is that that there is no way to improve or customize the system. Following table (Table 1) brings out the features and the differences of these products [5] – [7].

Table 1. Comparison of commercial facial recognition products [5] – [7].

	Kairos	Amazon	Microsoft
Self-Hosting	Yes	No	No
Face Detection	Yes	Yes	Yes
Face Identification	Yes	Yes	Yes
Face Grouping	Yes	Yes	Yes
Video Input	No	Yes	No
Price per 1000 transactions (EUR)	2.1	0.80	0.76

Despite that all of these service provides have large feature sets, the Smart Elevator project would only require face detection, identification and grouping. The grouping feature is the most crucial as it forms groups out of pictures that are representing the same person. The input data for that could be the detected faces from the live feed saved after some period of time like a day or week. The formed groups can be used as training data for the identification. After that, the system could start making an identification API calls by querying “is the entered person part of any these groups” [5] – [7].

Above APIs rely on the face being on the regular RGB images. But the main setback for this approach is the presentation of the face does not guarantee detecting whether it is a person or images of person (on the provided image) as the traditional photo does not provide depth information (3D parameters of the face). Apple Inc has overcome this issue by detecting face using Infrared cameras. They are reflecting 30 000 dots over the human face and then try to match these dots pattern wise and claim the change of error is 1 in 1 000 000. Although their results are outstanding, the applied techniques for the recognition is a company's secret [8].

2.3 Image processing

Navigation through the image information is one of the crucial steps in machine learning. Following sub-paragraphs start with a description of RGB image representation inside computer followed by fundamental theory about the neural networks that process images.

2.3.1 RGB frame

Computers represent colored pictures as multidimensional matrix $n * m * 3$ where n and m are representing the height and the width of the picture in pixels and the 3 is number of colors represented in the RGB color model (Figure 3). This model is used due the physics of modern screens, where colors of light are additive for human eye and therefore light sources for spectrum are red, green and blue [9].

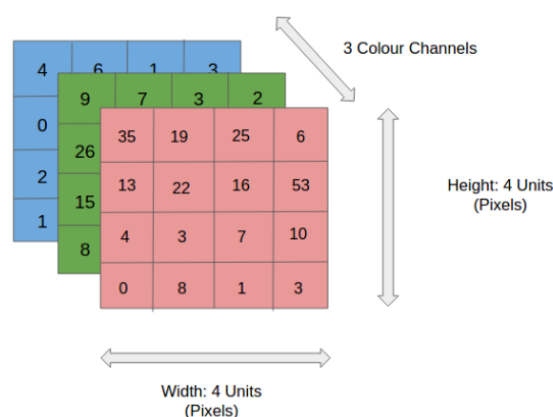


Figure 3. RGB image representation [9].

So in general, for each picture we have a structure, which is representing the impact of each color for each pixel. Finding out if two images contain the same object – an identified

person – would mean finding out if these structures of each image contain some kind of similar patterns. For face recognition, this would mean trying to carefully describe a pattern for each person and later check if the same pattern exists in another image. For example, providing a metrics of how eyes are most likely represented in the three color channels. It turns out, that in practice, this approach is not very efficient and would eventually require programmer to make far too many conditions and definitions for it to actually work. Moreover, even a different lighting conditions could mean totally different descriptions [9], [10].

2.3.2 Neural networks

Over the last decades, scientist have started to solve this task more efficiently with another method called artificial neural networks. This approach tries to mimic the reactions happening inside the human brain. Although, the full working principle of the brain is still unknown, some parts of it are discovered. For example, studies have found that the most basic elements of the brain, neurons, does not regenerate. It is assumed that they provide us with our abilities to remember, think, and apply previous experiences to our every action. Each neuron can be connected up to 200 000 other neurons and each human is expected to have around 100 billion total of them. Despite the fact, that there exists so many of them, each of the neurons is known to have four components - dendrites, soma (cell body), axon, and synapses. The components can be used to derive an abstract artificial neuron. In fact the very first model of an artificial neuron is introduced by Warren McCulloch and Walter Pitts in 1943. The representation of the mathematical model with derived properties from the real neuron can be found on the following picture (Figure 4) [11], [12].

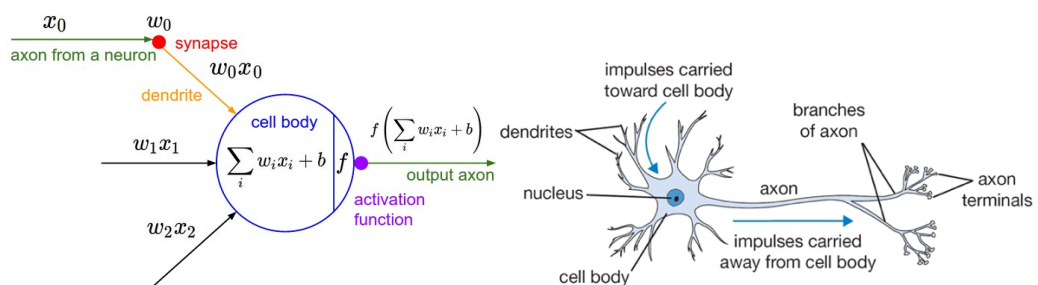


Figure 4. Neuron and artificial neuron [10].

Dendrites are accepting the input data, which can be represented as vector X ($x_0..x_n$). Then the vector will meet the cell individual properties - defined by weights W ($w_0..w_n$) and bias b . This process simulates the signal processing inside the real neuron cell body. For resembling the impulses along the axon, a normalization of the value is used through activation function. It also enables us to mimic the synapse at the end of axon terminal as this function describes, how the signal should pass through. What is more, this also helps to keep track on growing speed of the output values over the layers. It is worth mentioning that these descriptions are just abstractions and the actual relations and the processes could be far more complicated as there are still a lot of research happening in both fields [11], [12].

In artificial neural networks, the neurons are most commonly organized inside layers, where the output of the other layer will be used as input of another layer so that each neuron will be connected to each input of the next layer (Figure 5).

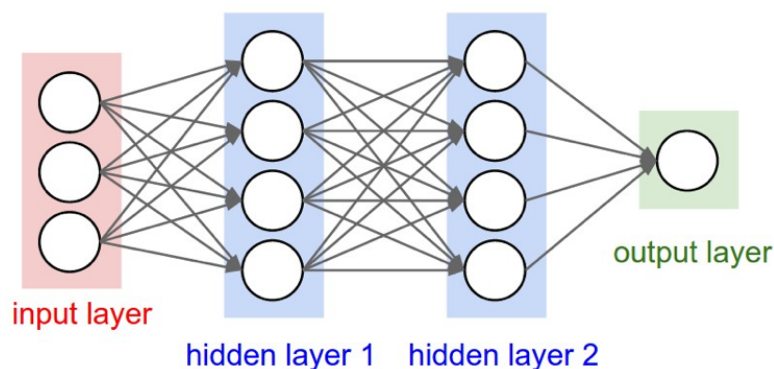


Figure 5. Classical multilayer neural network [10].

The first layer of the neural networks is called the input layer, last layer is the output layer and the ones between them are typically known as hidden layers. The number of neurons inside one layer is not determined by any rule and is usually chosen based on the need. Although, there are several methods available based on the task that the neural network has to solve [10] – [12].

2.3.3 Convolutional neural networks

Traditional multilayer neural networks filled with neurons do not work well with images. Suppose we have RGB images of faces in the size of 150x150 px. Therefore, a single fully connected neuron would have $150 \cdot 150 \cdot 3 = 67\,500$ weights. Taking into the

consideration, that normally solving a task that difficult would require several hidden layers, we would be spending enormous amounts of computing power. That has lead scientists to come up with a convolutional neural network architecture, which tries to process image as a 3D volume rather than turning it to one long 1D array. A typical CNN (convolutional neural network) consists of three types of layers – convolutional layer, activation layer and pooling layer [10], [13].

Convolutional layer processes the input matrix through filter. Filters are typically small (much smaller than the input) matrices, that produce dot product as they are sliding through the input. The length of the sliding window is called stride. When the size of the filter and the length of the stride makes it impossible to slide over the input without going outside of the input volume, padding is applied as it does not require changing the image dimensions. Commonly, zero padding is used, which means the that the input volume is extended by zero filled edges. The parameters inside the filter can be referred as weights, that are calculated during the training process. Figure 6 describes, how filter (K) is slided through (I) over the input matrix result a dot product as an output (I * K) [10].

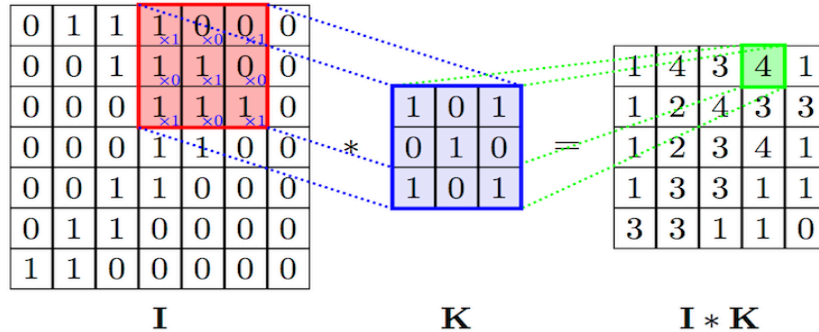


Figure 6. Convolutional layer filter work principle [10].

The purpose of the activation is to introduce non-linearity to the network because most of the real life problems are not linear. Activation functions are applied elementwise and followed by convolutional layer. Therefore, this layer is often mentioned in the literature as part of the convolutional layer. Most popular activation functions are Sigmoid, Tanh, ReLU and leaky ReLU. Although, most commonly in convolutional neural networks ReLU and leaky ReLU (Figure 7) are used as they tend to perform relatively better in training process [10].

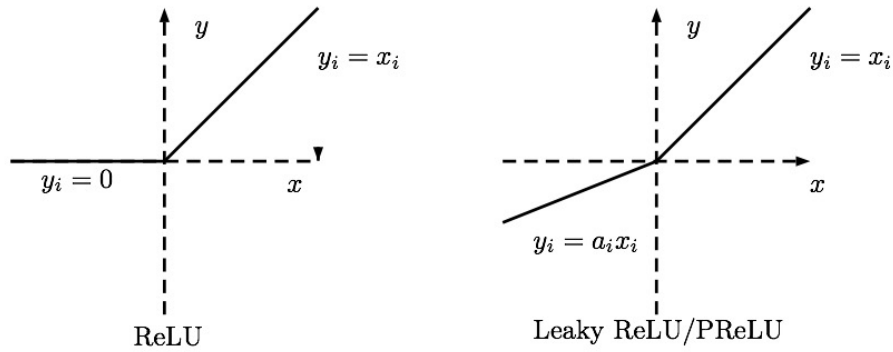


Figure 7. ReLU and Leaky ReLU/PreLU [10].

ReLU activation function basically removes the negative values from the network. Leaky ReLU performs essentially the similar means, except it allows a small slope (usually around 0.01) of negativity to be passed. They both saturate on the line between the linearity and nonlinearity [10].

Pooling layers are used to scale down the amount of computations happening the network. As a side effect, they also help to generalize the network for handling different data (prevent overfitting). It is dividing the input into smaller windows and producing some kind of down sampling calculation over that window. Following picture (Figure 8) describes max pooling, which passes the maximum element over the different windows to output layer [10].

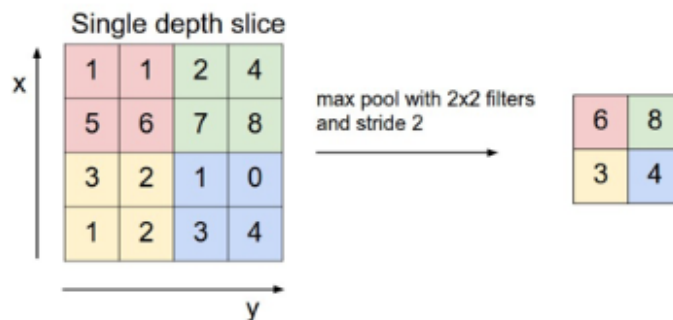


Figure 8. Max pooling [10].

Another pooling operation used very commonly inside convolutional neural networks is average pooling, which passes the average of the elements inside the windows to output layer. These layers are usually applied after every few convolutional layer [10].

2.3.4 Training

By training, people usually mean solving a problem, how each neuron (or part of the network) should configure itself to produce valuable effect, since only the definition of the neural network structure does not give it its computational properties. In order to give neural network some kind of behavior there are two general ways to teach it through - supervised and unsupervised learning. Both of these methods assume existence or gathering the correct training data. Exact amount of data needed depends highly of the solvable task [10], [13].

In supervised learning we give network many sets of samples how the output layer (vector) should look like, when having a specific input, considered that the input and the output vector are related. Without this, the network is unable to configure itself. During that period, the loss function, which calculates how much the desired output resembles to the current output, will adjust the weights for the next training sessions. Over the time the amount of weights needed to be changed will approach to zero and that can mean that the training process can be considered finished. Unlike for the supervised learning the unsupervised learning does not require the specification of correct values. Instead the goal of this training method is found out which output class should the input go. After several training samples, the core description of each output class becomes more clearer as the number of predicted values for these classes increases This method can be used for clustering applications, where the number of different classes and the exact description of each class is not known before the training process [10].

3 System Design

The properties of the individual elements define the properties and the abilities of the system. System design consists of hardware and software selection. The first part of the chapter describes, what type of camera should be used and where is the optimal position for face identification followed the by experiments with different software components.

3.1 Hardware selection

Author has compared three different cameras – Axis M1011W, Intel RealSense D435 and Basler ACE1440-73gm. They were selected mainly because these cameras were already available in the department and all of them were from the different price range, providing an overview (Table 2), what features various price ranges are offering.

Table 2. Comparison of different cameras [14] – [16].

	Axis M1001	Intel RealSense D435	Basler ACE1440
Resolution	640 x 480	1920 x 1080	1440 x 1080
Interface	USB 2.0	USB 3.0	GigE
Frame quality	Average	Good	Excellent
Suitable environment	No	Yes	Yes
Configuration options	Bad	Good	Good
Price (EUR)	168	215	425

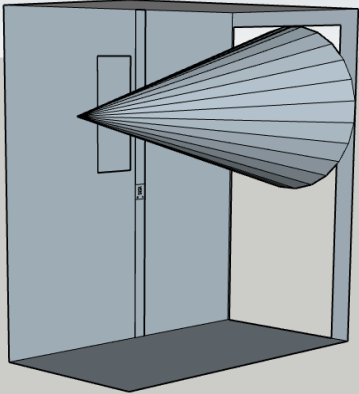
Finally a Basler ACE1440 was chosen, which provided more value than others. It is especially designed for industrial environment, is easily integrable and has a built in ability to capture focused and sharp frames even when recorded objects are moving. The camera itself also came with the special SDK (Software Development Kit), which enables programmers to configure the camera parameters inside the program and access raw frames directly using officially supported drivers.

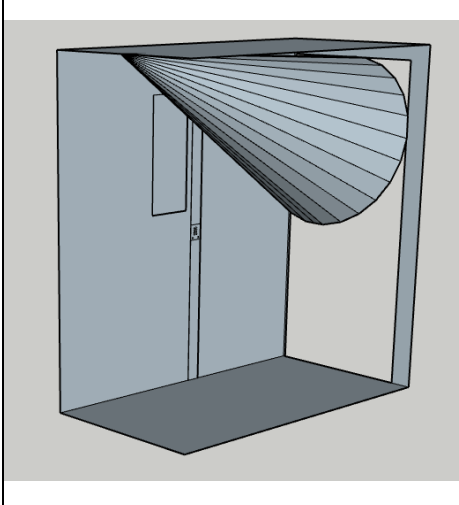
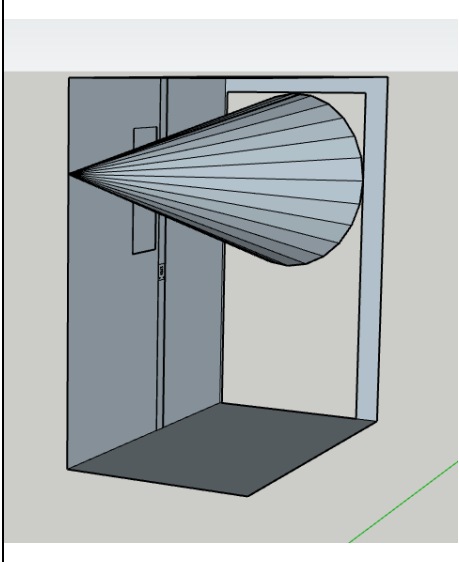
To capture whole elevator into one frame, an optimal lenses for the optics were needed. Author did borrow many different lenses from ISEAUTO project hardware depository to find out the best one suited for the elevator. First strategy was to use fisheye lenses, which can capture very wide area in small rooms and therefore are often found in small corridors on security cameras. Unfortunately the fisheye effect changed the quality of the people faces way too much and thus, made the prediction results very inaccurate. Therefore the only solution for the lenses was to find right balance between the view angle and image transform, which eventually turned out using focal length of 8 mm.

3.1.1 Camera positioning

Author created a model to test different positions for the camera in order to find out, which position and angle favored face detection the most. In total there were three strong potential candidates. After the deep analysis of the different positions, it was clear that the camera will go to position number 3 (see following table) for ICT elevator. It's main advantage was that this position enabled us to install the camera right to the corner behind the special corner pad, which could make it easier to hide all the cables and make it look more like a product, rather than a ugly prototype. Vertical position of the camera should be at 155 cm enabling the system to detect person between 140 - 210 cm tall. The overview of all the position candidates with their strengths and weaknesses that came to attention during the test recordings are described on the following table (Table 3):

Table 3. Comparison of considered camera positions.

Position	Strengths	Weaknesses	Model
Position 1. Camera is attached to the center directly towards doors	We can see people before they walk in Very good angle for detecting faces Camera draws attention, because it is centered	Difficulties with children and short adults Cannot see people when they are very close to the camera People tend to look at button console	

		Cannot see people when they are behind each other	
Position 2. Camera is attached to ceiling with a vertical angle of 30 degrees between the ceiling and the camera	Better to capture entering people, when elevator is almost full Does not get so much attention	Harder to identify people from above view, especially shorter persons People look toward button console when entering People look down when entering	
Position 3. Camera is attached in the corner with a horizontal angle of 12 degrees between the sidewall and the camera	People tend to look towards button console or guide board, therefore higher chance to capture face Good installation options	Other side will have large blind spot Works better if people directly look there When elevator is smart, then people should not have reason to look buttons	

3.2 Software selection

Despite the variety of different technologies the key steps for facial recognition are the same (Figure 9). First, the face detection from the input. It means finding and deciding whether the input, which can be a photo or video feed contains the human face. Secondly, the face signature features extraction, which gathers and forms a comparable pattern of the unique landmarks from the human face, like the size of the forehead, distance between the eyebrows, shape of nose and so on. And finally, the result comparison with other face signatures [17].

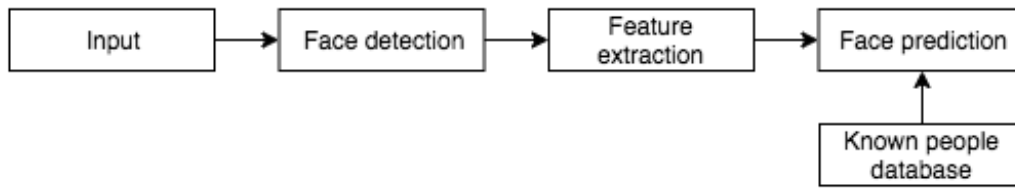


Figure 9. Face recognition pipeline [17].

3.2.1 Face representation

To compare, cluster or process face in the image in any way, we need to transform the face to specific representation form, which will maintain the distinguishable landmarks of the human face. This can be done by using convolutional neural networks and the process itself is often referred as feature extracting. It has one good advantage - there is no need to retrain the network with new faces. Instead we train the neural network to see the similarities between two persons, which is done by mapping each face to some kind of embedding space where the distances between these faces are comparable so that pictures of the same person will be closer than the the pictures with different persons [18].

Author has used pretrained feature extractor convolutional neural network mainly for three reasons. First, the goal of this thesis was to create automated face management system rather than propose convolutional neural network for face identification. Second, the training of the state of the art model can take up 500 hours, what is more, the parameters of the training are very crucial. Meaning that there is no guaranteed result of creating a better model than already available ones. And finally, this project had very strict time limits, therefore the proof of concept had to be delivered very fast, which meant that there was not enough time to study and develop our own convolutional neural network [18].

There are many good pretrained models available in the Internet. Although, their publishers usually provide very popular LFW (labeled faces in wild) benchmark result, author wanted to try some of them (Table 4) out to realize, how well do they perform in different environment and dataset. All the feature extraction algorithm were benchmarked based on their ability to create distinguishable vectors for clustering algorithms since the plain LFW benchmark is just an verification benchmark and does not provide overview

how well similar and non-similar faces can be grouped together. What is more, author used the face detection and alignment methods on which these different feature extractors were initially trained and benchmarked on to maximize their performance. This was of high importance since the David Sandberg's FaceNet implementation expected to have a background margin around the face, but the Dlib and the OpenFace expected only the face area [18] – [22].

D. Sandberg hosts one of the most successful FaceNet implementation around the open source community. FaceNet is a system proposed by engineers in Google, Inc. Their method proposes a deep convolutional network that extracts and directly optimizes the face features itself down to only 128-bytes per face, while providing accuracy of 99.63% on the the facto LFW benchmark. Author of the repository has implemented the whole FaceNet project using popular machine learning library Tensorflow, which enables to developer to train, test and use their models on many different platforms. The open source package in GitHub contains two different face recognition models (referred as version 1 and version 2 respectively), which are based on the Inception Resnet V1 architecture proposed by Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi in the paper “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning” [23]. The reason, why they both were included to tests is that, they are trained with different training data, use different distance metric and produce different size of embeddings - 128 bytes and 512 bytes [22], [24].

Another very popular open source implementation of FaceNet is provided by Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. This project was supported by the US National Science Foundation and had assistance from many great technology companies including Intel, Vodafone and NVIDIA. Containing toolset is crafted using PyTorch, which is a Python machine learning library developed by Facebook Research Team. It is also referred in many other open source face recognitions implementations as seed of architectural inspiration. Their most accurate model (nn4.small2.v1) is trained on the FaceScrub and CASIA-WebFace and the output feature vector representation size is 128 of bytes from a face on the 98x98px image. Although the claimed LFW accuracy is a little bit smaller than others, their main advantage over other models is that they have managed to cut down the processing time by cutting some parts of the original model. Their model is NN4, which is described in the FaceNet paper with 4b, 4c, and 4d layers removed and smaller 5 layers [21], [25].

Dlib itself is a modern C++ toolkit containing machine learning algorithms and tools to solve real world problems. Moreover, it is also possible to apply its tools from Python applications using special API, which makes it more easier for people, who do not have strong programming background. The library itself is also equipped with the NVIDIA CUDA support, which means that many parallel computations are executed on the GPU rather than a CPU. The face recognition model provided in the toolkit is a variation of the ResNet-34 deep convolutional neural network, which was proposed by the He, Zhang, Ren, and Sun in paper “Deep Residual Learning for Image Recognition” with some custom modifications. More precisely, toolkit author has remove some of the layers are completely all and cut the number of filters used per layer. These modification helped enable to create more faster network than the original one. The model was trained on a dataset of about 3 million faces composed from Face scrub and VGG datasets and an amount of faces from the Internet [20].

Table 4. General overview of pretrained face recognition models [19] – [22].

Name	Network	LFW accuracy	Training dataset	Speed on Nvidia Xavier
FaceNet by David Sandberg v1	Inception-ResNet-v1	0.9905	CASIA-WebFace	~0.16s
FaceNet by David Sandberg v2	Inception-ResNet-v1	0.9965	VGGFace2	~0.16s
Dlib	Resnet-34 with few layers removed and number of filters per layer reduces by half	0.9938	FaceScrub, VGGFace, hand-picked images from the Internet	~0.13s
OpenFace	FaceNet's NN4 network with 4b, 4c, and 4d layers removed and smaller 5 layers	0.9292	FaceScrub and CASIA-WebFace	~0.09s

3.2.2 Clustering

The final product should be able to create clusters based on the face embeddings from video feed. Idea behind clustering is to organize unlabeled data into similarity groups called clusters. This makes desired clustering algorithm to solve two main tasks - decide, how many different clusters we have and put similar samples together. Author has compared three algorithms - Mean-Shift, DBSCAN (density-based spatial clustering of applications with noise) and Chinese Whispers. These algorithms are suggested in multiples sources in both published papers and guidelines from the Internet [20], [26], [27].

Chinese Whispers clustering algorithm was introduced by Chris Biemann from University of Leipzig for solving natural language processing problems. One of the strengths of this algorithm are speed (linear in the number of edges) and simplicity. It aims at finding groups of nodes that broadcast the same message to their neighbors, thus can be viewed as a simulation of an agent-based social network. The algorithms is outlined in the following code block (Figure 10) [28]:

```
forall  $v_i$  in  $V$ : class( $v_i$ )= $i$ ;  
while changes:  
    forall  $v$  in  $V$ , randomized order:  
        class( $v$ )= $\text{highest ranked class in neighborhood of } v$ ;
```

Figure 10. Code block describing the Chinese Whispers clustering algorithm [28].

At the beginning, every node is treated as a separate cluster. After that n number of iterations are applied so that each node will be combined to the closest class in the local neighborhood. This will be the class whose sum of edge weights (distance in our case) to the current node is maximal. When multiple such matches happen, random class could be chosen. Regions of the same class stabilize during the iteration and grow until they reach the border of a stable region of another class. Only important parameter in this algorithm is the threshold between similar examples. In current thesis, an implementation of this algorithm is used from Dlib machine learning library [28].

Another very popular clustering method is density-based spatial clustering of applications with noise (DBSCAN). It was proposed by Computer Scientist group in University of Munich. Its main advantages are that it can detect clusters with arbitrary shape and is able

to detect data that does not fit to any cluster, so to say noise. This algorithm is outlined in the following code block (Figure 11) [29]:

```
Find neighborhoods N that, which have more than MinPts members;  
Recursively find the neighborhoods inside found neighborhoods;  
Combine results to clusters;  
Treat left out point is noise;
```

Figure 11. Code block describing the DBSCAN algorithm [29].

So, the algorithm arbitrarily chooses a random point, then it starts to discover its neighbors, which are closer than Eps (similarity threshold). If neighborhood is bigger than MinPts (minimum samples per one class), cluster is formed, otherwise another point is chosen. When discovering a cluster, it starts to search neighborhoods in the view of each point in the found cluster and repeats the process on every newly discovered point. Every point left out of clusters, will be treated as noise. Therefore, the main parameters DBSCAN are Eps and MinPts. Analysis in this thesis regarding the DBSCAN used the implementation found in Python Scikit toolset [29], [30].

Mean-Shift clustering algorithm was proposed in 2007 by Kuo-Lung Wu and Miin-Shen Yang from Kun Shan University of Technology. It aims to find the mean densities between data points. Like DBSCAN and Chinese Whispers, it does not require to specify the shape of the data nor the amount of clusters to be found. Its core algorithm is described in the following code block (Figure 12) [31]:

```
For every data point:  
    Until no shifting:  
        Find points inside the bandwidth  
        Calculate the mean of these points  
        Shift bandwidth to mean  
    Turn converged points to clusters
```

Figure 12. Code block describing the Mean-Shift clustering algorithms [31].

As the description of the algorithm reveals, the main setback of this algorithm is the computing time. Beginning similarly to the previous algorithms, at first, the neighbors are found inside bandwidth (similarity threshold). Then, the mean point of the neighborhood calculated and the neighborhood is shifted to that point. This process is repeated until distance of the shift is zero. After iterating over all the data points, the

points that ended up at the same final mean point will form a cluster. Analysis of this thesis is using the Mean-Shift implementation is used also from the Python Scikit toolset [31], [32].

3.2.3 Performance comparison of clustering algorithms

For testing, author has mixed the LFW dataset (90%), photos personal collection and faces gathered while recording (10%) having total of 2000 different persons. Main purpose of the comparison script was to find out, how well the feature extractors work with different clustering algorithms under different parameters. Working principle of the comparison script in metalanguage is described using following code block (Figure 13) and the actual implementation in Python can be found in software repository */identifier/utis/benchmark_clustering.py* (Appendix 1).

```
For each dataset size (10, 100, 1000, 2000):
  For every iteration do:
    Select n random classes
    For different threshold value do:
      Apply all clustering algorithms
      Calculate difference from dataset
    Save best threshold value with accomplished accuracy
  Calculate the average accuracy and threshold over all iterations
```

Figure 13. Code block for comparing different feature extractions with clustering algorithms.

Author repeated the clustering benchmarks for 20 iterations each time with different batch from the whole dataset to make sure, that the benchmark would meet different inputs. The parameter searched for the clustering algorithms was the “similarity threshold”, which was used in different ways depending on the algorithm specific principle. The results showed on the following graph (Figure 14).

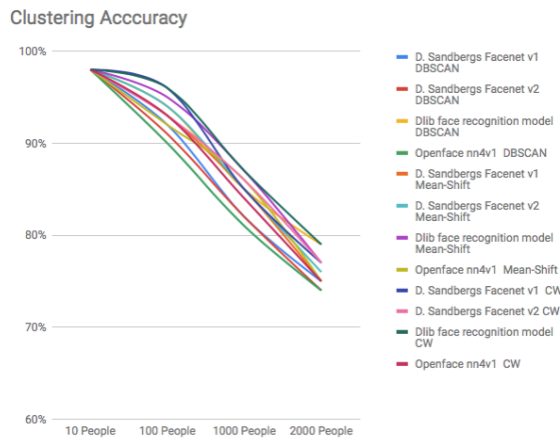


Figure 14. Clustering accuracy change over the dataset size.

Experiments showed, that the clustering accuracy is dropping drastically as the number of people rises. What is more, additional inspection revealed that in most cases the errors were caused because of the sunglasses, quality of the picture and face alignment for the camera meaning that faces which looked directly towards the camera had higher chance to be labeled correctly. During the tests, author also saved the speed of the execution for each algorithm. It has to be mentioned that the speed was captured using the standard Python time function, meaning that the real computing times may vary. Following graph (Figure 15) illustrates how the time needed for computing rises as the number of people increases:

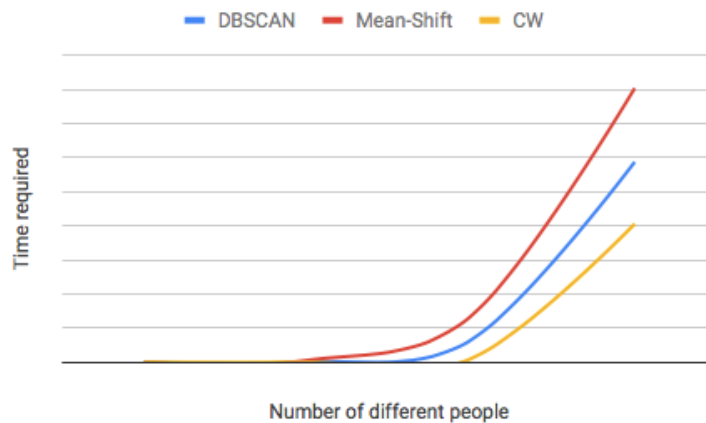


Figure 15. Computing power required over dataset size

As a result, author selected out Dlib face recognition model together with Chinese Whispers clustering. Their main advantages were that they produced the highest accuracy, had smallest computing time and the threshold hyperparameter for clustering algorithm remained the same over the dataset changes staying stable around 0.475. Having a stable

parameter gives great advantages over the dynamic one, which could add additional complexity to the system.

3.2.4 Prediction

Published papers and guidelines over the internet tend to guide people to use two methods on top of the face feature extractors - NN (nearest neighbor) algorithms and SVM (support vector machines) [17], [18], [33], [34].

K-Nearest Neighbor was one the first classifiers used in machine learning. The algorithm itself is very simple - it finds K nearest matches (neighbors) to input and then checks the majority of the class members present in that field. The problem is that this approach makes it impossible to find unknown faces as it always finds nearest ones to given input [17]. A variation of this algorithm is called Radius Neighbor algorithm, which provides us to query neighbors within specific radius [32]. To make the prediction even more accurate, we can define the prediction function only to be accurate, when specific size of majority presence is achieved in the given radius. Following two graphs (Figure 16) describe, how does the number of training images and the selected radius affect the prediction accuracy:

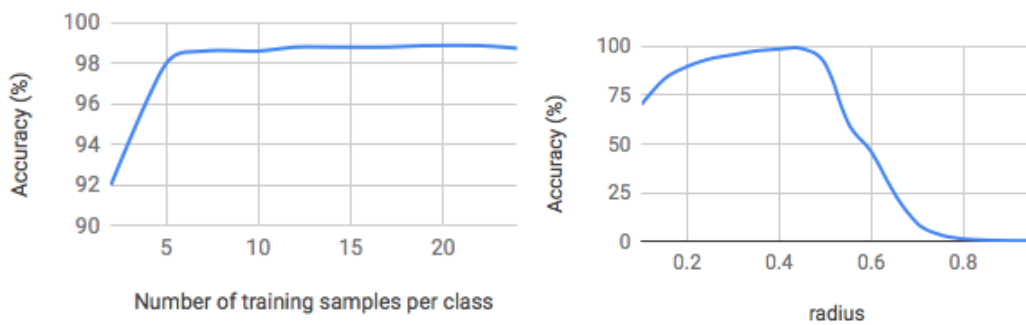


Figure 16. Radius Neighbor classifier accuracy change over number of training images and radius.

SVM is one of popular pattern recognition classifiers. Unfortunately, the main setback of this algorithm is that it is really difficult to detect unknown people. One way to solve this problem is to use threshold value on predicted class probability score. Author has used this functionality implemented inside Python Sklearn library [32].

Following table (Figure 17) describes, how the number of training samples per person affected accuracy of the results:

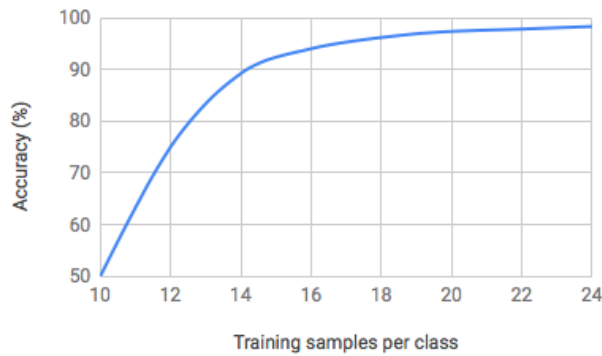


Figure 17. SVM accuracy over the number of training images.

Although the SVM showed remarkable results in our experiments with calibrated parameters, the general key finding over the different tests was that the more classes we have, more training data is needed to differentiate between known and unknown person as the probability scores were dynamic depending the size of training data and classes. That said, it would be technically difficult to use SVM classifier in this project as the training data is generated by the system over time from different elevator sessions to gather different samples per person, meaning that after some time it would take too long to save person for recognition.

Software implementation in Python comparing both of these algorithms can be found under software repository in `/identifier/utis/benchmark_classifiers.py` (Appendix 1).

4 Automated face management system

This chapter introduces proposed algorithm, how the automated face management system can be implemented for the Smart Elevator project and breaks down its role in the facial recognition pipeline together with all the other components. It also describes the test methodology used for evaluating the initial version of the implementation.

4.1 Proposed algorithm

Knowing from the conducted tests that the clustering algorithm can work really well with small number of different people, it is wise to perform first level grouping in the scope of elevator cycle. Elevator cycle in this work is considered a period of time, during which there are people inside the elevator. In the end of each cycle a clustering is performed. This will produce small number of clusters with relatively big accuracy. To ensure that the system gathers different samples of each person, it filters out only random ten faces from each cluster and throws rest of them away. Illustration of the results after N elevator cycles can be seen on the following picture (Figure 18):



Figure 18. Saved clusters from elevator cycles.

After these “mini-clusters” in the scope of each cycles are produced, the next task is to find out, whether exists situation where same person is represented in many different batches. This is where the higher level clustering comes in. The idea is to combine all the

data saved over the cycles and find out if cluster bigger than the size of 50 can be formed. If such clusters are created, 30 nearest samples from the cluster center are taken out and marked as a collection of samples for each person and therefore are saved to known people database. On the following picture (Figure 19), the clusters centers are marked as a red dot whereas the red background represents the area of closest 30 samples from it.

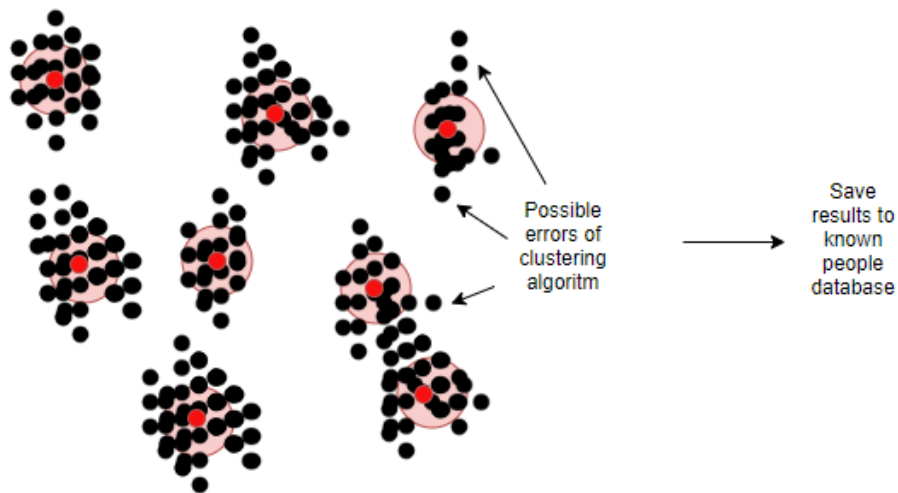


Figure 19. Illustration of valid clusters.

The main purpose why cluster size > 50 is reduced to closest 30 is to filter out the errors of the clustering algorithm, leaving it have 40% error margin for edge cases, which in author's experiments have shown to be the root cause of faulty cluster predictions. Also, the 40% margin adds "extra space" for errors considering the clustering accuracy of Chinese Whisper algorithm with a lot of different people. If a known face is saved to database, then the rest of the samples left to the cluster are deleted as they are not required anymore. Since, the higher level clustering can take a lot of time processing all the feature vectors gathered from the different elevator cycles and also known people database, it will be run during the night period. To keep the data collection amount reasonable, all the feature vectors older than two months will be also discarded.

4.2 System overview

During the elevator cycles all the face captured will be recorded and saved together with the timestamp and the current floor (this information was unavailable in the current development phase). To detect faces from the video, HOG (histogram of oriented gradients) feature extractor combined with linear classifier is used, which was included

the Dlib toolkit. The detection algorithm had to be chosen so that it would only detect faces directly aligned to the camera to increase the accuracy of the face predictor and the clustering algorithm. In the end of each elevator cycle, Chinese Whispers clustering algorithm is applied. This clustering layer enabled the system to group visited people together by cycle with relatively high accuracy as the analysis showed, that the less people is included, the better the results are. Afterwards filtering is applied to save only few samples of the visited persons to database to prevent it oversizing and to catch the errors of the face detector, which were typically presented as clusters with one to three samples.

In the end of every day, the results of the currently known people and groups inside captured elevator cycles are combined. This processing is assumed to take up to several hours and therefore is not executed during the day. If the data is older than two months (this period came directly from the overall system requirements), it will be discarded. It is mainly because there is no reason for the system to memorize people who visit elevator temporary, instead this design allows elevator to predict for everyday users.

For combing, another layer of Chinese Whispers algorithm is applied. After that, the goal is following filtering is to find out whether or not a known people class can be formed. A detailed description is written in the previous sub-paragraph (4.1). If so, the person will be added to the known people directory with a generated unique ID and if not the group will be put back to elevator cycles to wait for another try, which can happen, when enough samples from the person is gathered over future elevator cycles. When the person is added to the known people, is used as input data for the Radius Neighbors Classifier and therefore can be recognized. Also, this person will be seen in the user interface and the information about the user can be accessed over the API. The overall system layout is described on the following image (Figure 20) and sequence diagram describing the more detailed software representation is on Figure 23.

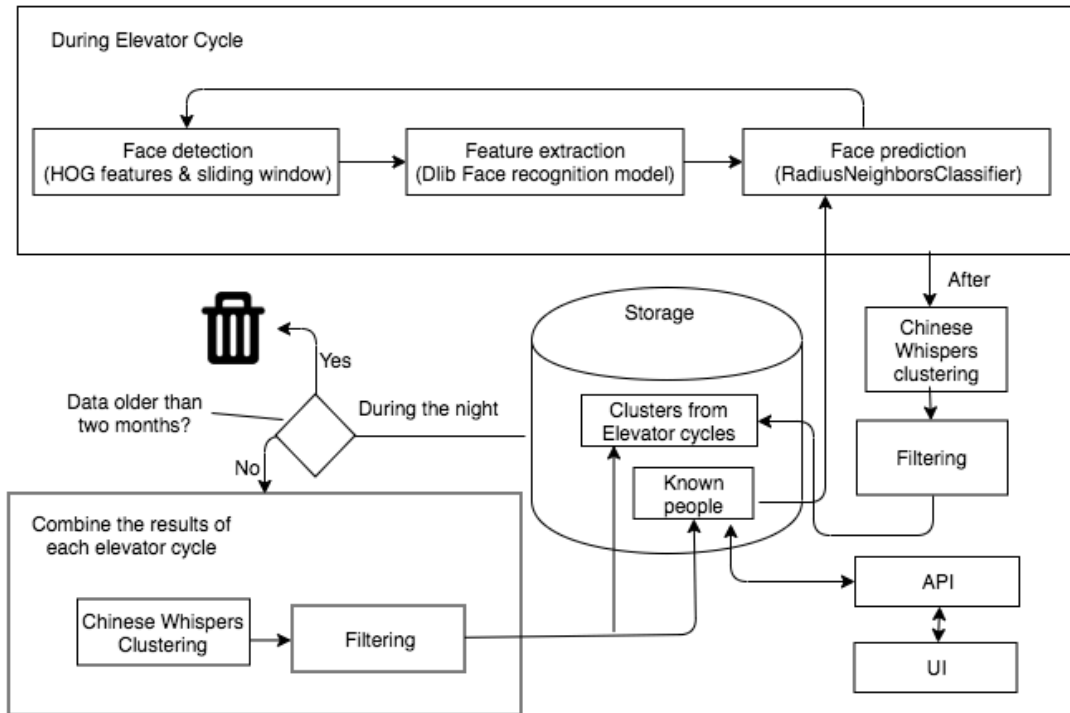


Figure 20. System layout with flows.

4.3 Experimental results

In the scope of the first part of the development, author has gathered and concluded the first results of the described algorithm above. In total, there were recorded hundreds of different elevator cycles. Following algorithm managed to gather 133 persons out of 190 with a accuracy rate of 99.2%. Some people were not added because the algorithm could not gather enough samples of each person during the recording sessions as it only saved limited amount of samples per elevator cycle. Therefore, the ungrouped faces were not counted as error matches. Also, in these experiments the faces were learned a lot faster than they would have been in real life, because during the records author asked people to use elevator more frequently as they would have done normally and duplicated some of the cycles to simulate recurrent usage of the elevator. The amount of visiting persons and errors in created clusters were counted manually using the help of Chinese Whispers algorithm and thus the real performance of the system may be slightly different accuracy due human error.

To get a first overview of the system identification accuracy, author has used the dataset generated by the system itself. Author admits, that having such small dataset may give somewhat inaccurate results, but the main goal of these tests was to test out, how well the system currently behaves and get the first actual overview of the system performance. For testing the dataset was divided as described on the following picture (Figure 21):

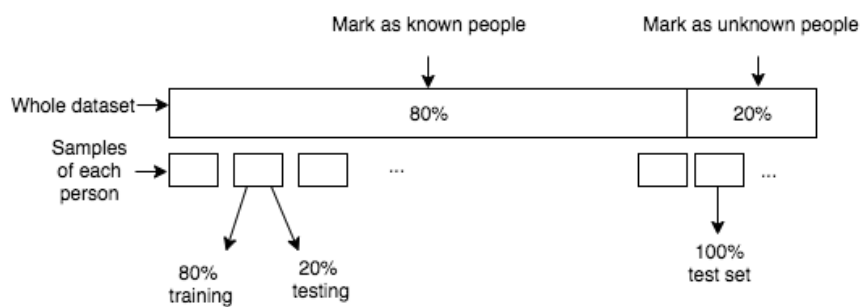


Figure 21. Test and training dataset for prediction.

As the system has to face known and unknown people, 20% of the dataset was marked as unknown people. The specific percentage was chosen for two reasons - first, it is common practice the split datasets to 80/20 and second, after some time, expected average number of unknown people entering to the elevator over time can be around that area [9]. Then, samples of each person from the known people were be split to training and test set. The training set simulated the known people gallery in the database and the test set was treated as the live camera input. This method showed identification accuracy of 98.8%. Software implementation used for benchmarking can be found under software repository in */identifier/utls/benchmark_classifiers.py* (Appendix 1).

5 Integration and Interfaces

This chapter contains descriptions about the communication protocols and methods used for integration. It also introduces the architecture, capabilities and the features of the created user interface.

5.1 Communication with other system components

Smart Elevator is a large complicated system containing different pieces of software that need to transmit information with each other to serve a higher goal. Next paragraphs introduce the communication middleware shared between all of the components and an overview, which kind of messages are interacted with the proposed system.

5.1.1 ROS

Smart Elevator consist of many different software components that are not always running on the same machine (Figure 22). ROS (Robot Operating System) helps to overcome the problem how to pass information from one software component to another. It is a middleware, that helps real time systems to publish and subscribe information, which requires very specific timings and is asynchronous. What is more, it gives availability to control passed information from one central unit. This means that when applying new security rules for connections, all the system components will automatically will start to use it without tweaking each of them individually [35].

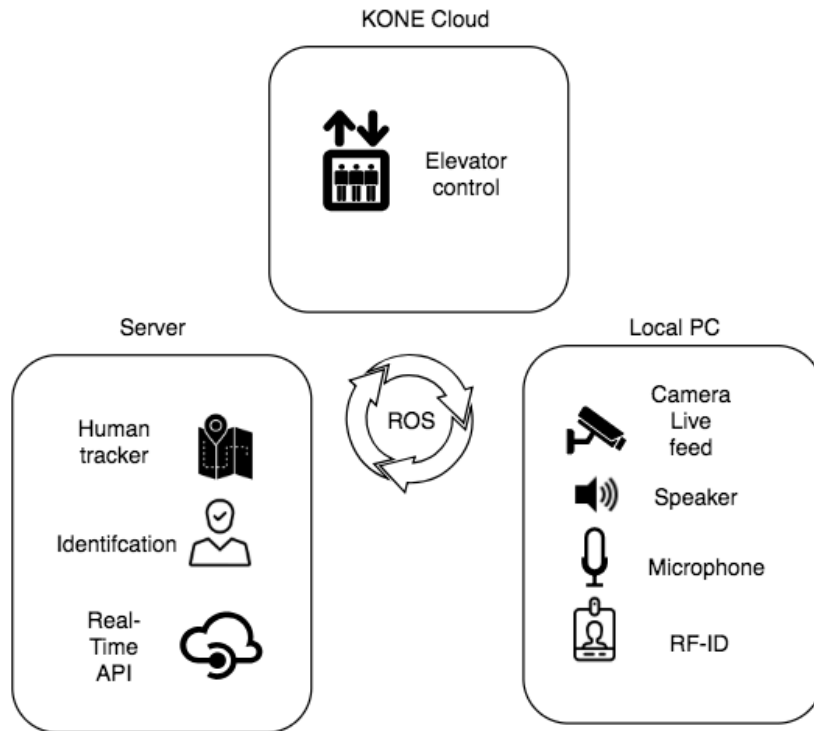


Figure 22. ROS.

This thesis covers the working principles of three components seen from above picture - camera feed inside the elevator, face identification and real-time API. The video feed inside the elevator is transferred to server, where the identification procedures happen. This is mainly because the server had more computing power for that purpose. This architecture also enabled subscribing components to keep track on the timestamp of the identified face in order to prevent processing expired data. Handling expired data is a very common problem in the real time systems and this usually can happens due the connection issues between different components [35].

5.1.2 Interactions

Although the face recognition system has its own working principles and flows, it also provides services for other system components. Generally, it has two main tasks - answering to the identification request made by person tracking component and writing recognition results to live video stream. In the end of the current development phase, some of the interaction are mocked (yellow arrows of Figure 23) and can be simulated using the keyboard commands.

When the person tracking component detects that someone is entering to the floor, it immediately makes identification request with a unique ID. After that, the face identifier will immediately search for the first face inside the door boundary box that meets with the assumed size (the size of the faces at the door area from the camera perspective) and makes an initial prediction. The result of the prediction will be sent back with the same ID to querying component. This enables asynchronous communication that can handle multiple request without receiving an individual response before making another request. The identifier can also be used to update the prediction result, which can happen, when the initial prediction that was provided turns out to be false as overtime the identification system could have more time to process the face over several other frames.

The prediction results have to be also visible to the operating user, who can then validate how the system reacts in real-time. The input frames from the camera are published for the identification system by other system component using one ROS topic and after the frame is processed, the results of are propagated back to ROS on different topic. When a face is discovered, a green rectangle will be drawn on the frame together with the prediction result. The new topic is essentially made for the user interface, but the system architecture supports to use these results in other parts as well.

Complete overview of all the main interactions (including inner ones) is described on the following sequence diagram (Figure 23). Inner interactions are happening between three software components – main processing unit, which is responsible for handling all the service calls made for ROS, Identifier, that is processing the image and makes predictions and the Automated Face Management System, that takes care of face database based. As the jobs of the face management system can take a lot of time, they are executed on a separate thread.

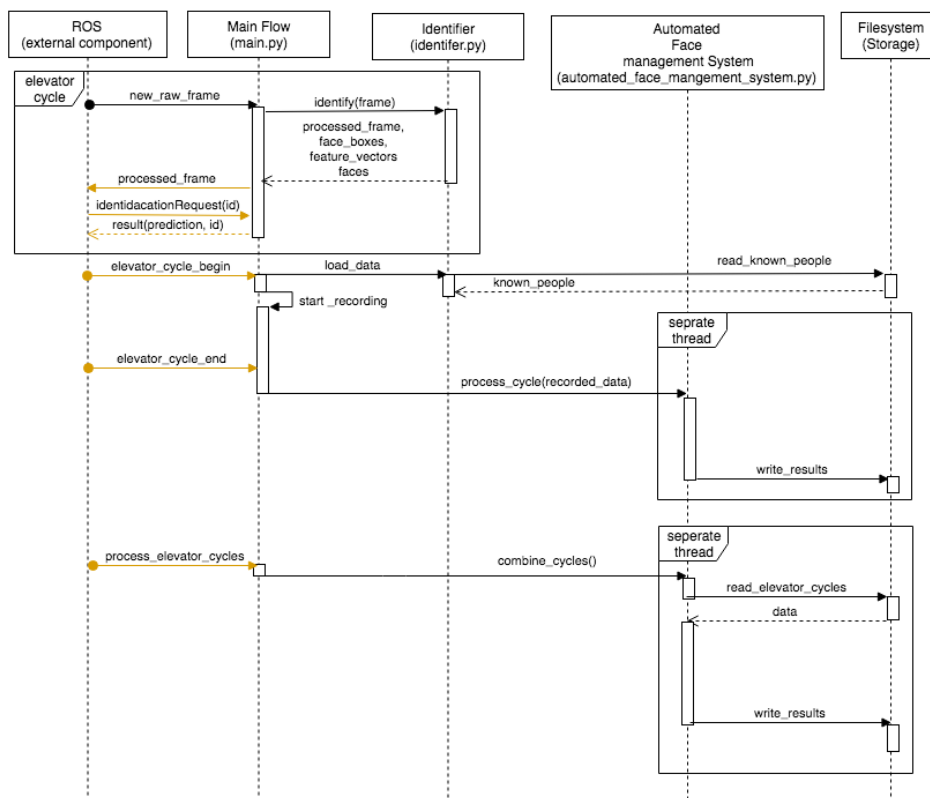


Figure 23. Sequence diagram describing interactions between system components.

5.2 User interface

This sub-paragraph describes building blocks and the main features of the developed user interface. It is divided into two sections - front end, which is everything that the operator can physically see and the back end, that drives it on server side.

5.2.1 Front end

To make the user interface compatible with different kinds of devices and operating systems, it was written to be a web application. This approach is a common practice in modern application development. One of the most famous examples of that is Google Docs, which year after year is taking users away from traditional desktop applications like MS Word and LibreOffice as it is portable and works the same way on every device.

In order to follow as much standard practices rather building something “on the knee”, the developed user interface was written in one of the most popular today’s front end libraries - ReactJS. This toolset was published by Facebook in 2013 and was built to overcome the maintenance problems of web applications with complicated business logic

an over the last few years it has grown more popular than its main competitors (VueJS and AngularJS) [36].

Although the main goal of this project was to automate the face management system, meaning the the faces should be added and deleted automatically from the system, the user interface provides an option it to do manually. This feature was added in order to use it in other projects as well, where there is need to control the the face database manually for example security related fields. One very strict requirement that came from the law department was that there should be an option to delete the person from the database. Following picture (Figure 24) illustrates, how author’s implementation of this page.

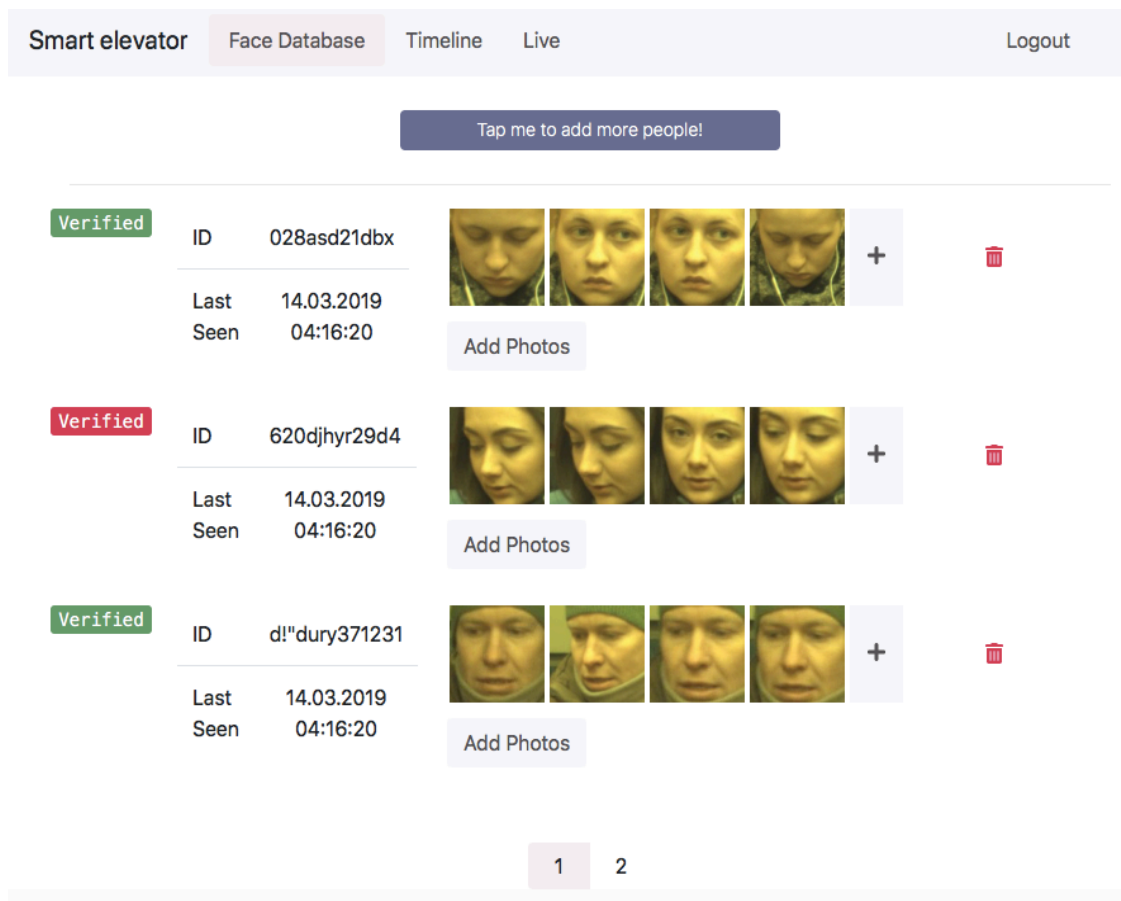


Figure 24. UI - face database

The green “Verified area” means that the user has been verified by staff card. As the RFID verification integration was not part of the first phase development, then currently, the verification status will be set randomly in order just to introduce the feature. The goal of

the staff card integration was to add employees ability to replace the random ID generated by the system (as a requirement of GDPR) with valid name.

All the events happening inside the elevator will be stored to the database. Timeline page (Figure 25) was created to view these events. User can limit the events by specifying a date. One problem of this page was that there could be a lot of events happening during one day, which means that the loading times for rendering all of them could leave operator with bad user experience. Author has solved this problem by implementing it to be continuous scroll. This means that only small portion of the events will be loaded initially and next ones will be only displayed, when user starts to scroll over the page. As the event management system was not development in this phase of the project, the events in the user interface are mocked manually.

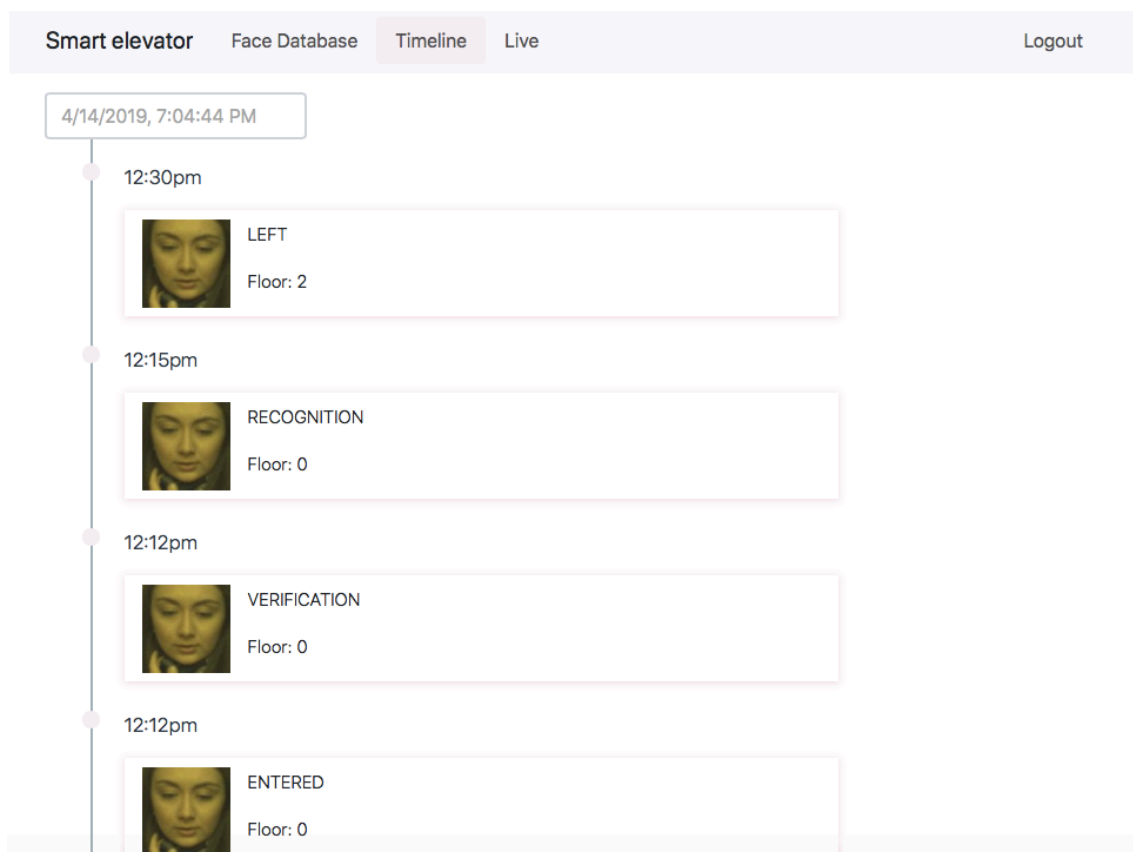


Figure 25. UI - timeline

The function of live page was to user an overview what is happening inside the elevator. This means the face identification (left side on the Figure 26) and person tracking (right side on the Figure 26) results with elevator state and detected command. The higher goal of the page was to give participating developer an overview about the status of the system and what decision it makes.

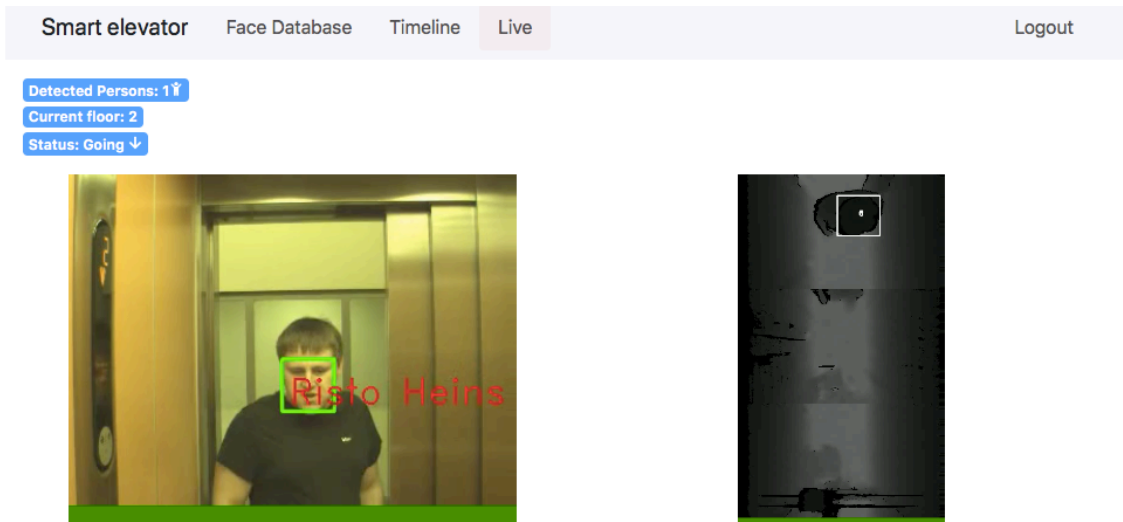


Figure 26. UI - live overview of RGB camera and depth camera results

5.2.2 Back end

Created back-end system has five core functionalities - authentication, image serving, propagation of elevator live status and lets user to request filtered data. Backend functionalities are served through API. Whole backend is written in NodeJS ver. 8, which has non-blocking IO, which enables easier integration of asynchronous events happening in real-time systems. It runs inside Chrome V8 JavaScript engine, which means it can run on any modern operating system the same way [37]. The general API architecture is described on the following picture (Figure 27) and the detailed overview is summarized on the next paragraphs.

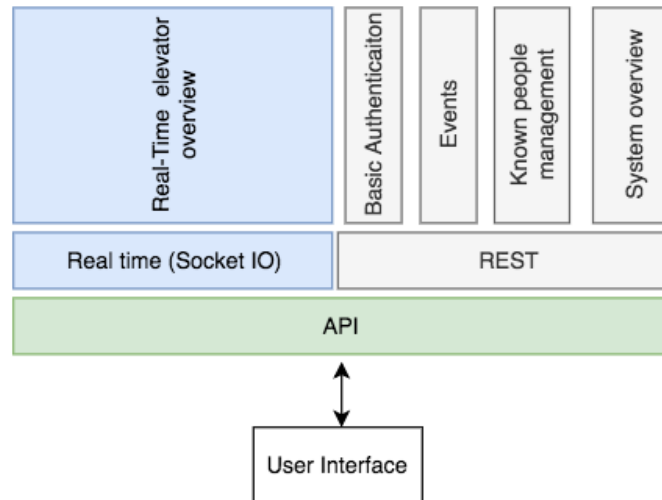


Figure 27. API architecture

The main reason for building a back-end on top of authentication system was that the information that the server has is very sensitive and cannot be accessed by public to prevent TalTech from violating the data protection rules. Our team has been in close cooperation with TalTech law department and asked for security advice from other experts.

All the API calls (except the login request) are protected by 'Basic authentication', which is described in RFC 7617. Its working principle is very simple - server should only provide access to the data for clients (users) who have provided special token, which basically is a string containing various symbols. This token is issued only if client has provided valid credentials and is expiring after certain period of time has passed. One of the core weaknesses of this method is that server only checks the token rather than finding out if the client really is who it claims. This means that if attacker manages to steal the token, the server will reply to any request [38]. In this project, this risk can be mitigated by allowing only HTTPS connections, which means that the token is always encrypted by RSA algorithm.

Most commonly made API requests inside traditional web browser are done using REST API, which enables programmers to define sensible queries using HTTP stack. It generally means a way, how browser can ask something from server. All the API calls made for this project are described in the following table (Table 5).

Table 5. REST interface general overview

Path	Method	Description	Parameters	Response
/api/auth	POST	Authentication	username, password	token
/api/timeline	GET	Get events by day	datetime	result, totalEvents
/api/gallery/known-people/thumbnails	GET	Get 5 from each random samples of the person	page, pageSize,	result, totalPages, page pageSize
/api/gallery/known-people/	GET	Get the whole gallery of the person	id	person_id, files
/api/gallery/known-people/upload/	POST	Upload new sample to known person	id files	msg
/api/gallery/known-people/add	POST	Add new people to database	files	msg generated_id
/api/gallery/known-people/delete	POST	Delete samples of person from database	person_id images	msg
/api/livestream/basler-stream	GET	Get access to camera stream	-	stream
/api/livestream/realsense-stream	GET	Get access to depth camera streams	-	stream

Opposite to the REST API, with WebSockets it is possible for server to send (push) notifications to user interface. This means that when backend notices data change, it can immediately notify the client, rather than waiting until client specifically asks for information update. In Smart Elevator project, the data changes are detected speech, number of persons detected, current floor of the elevator and elevator moving direction.

If a client has subscribed to these data changes, the back-end system automatically creates the link between client and the ROS topics, where these data changes actually are propagated. Although WebSocket and REST are totally different communication methods, author has developed an integration for these technologies to use same authentication interface.

6 Future development and suggested improvements

Smart Elevator project definitely will be developed further as the even the first phase one the development was not ended by the time of this thesis was written. This paragraph provides several suggestions, how to improve the proposed system and how to mitigate the errors for some of the critical parts.

Currently the system uses pretrained feature extraction model from the Dlib toolkit, which works relatively well. In order to further improve the accuracy of the model or introduce the elevator lightning condition to the network, some parts of the model should be fine-tuned. This is a common practice, since training the neural network from scratch can take a lot of time, without guaranteed result. This can be done by retraining some layers of the neural network with new dataset, which can be the face database generated by the system [10].

Providing identification information for people tracker component is relatively simple task if there is only single person facing toward the camera. Unfortunately, there are usually more than just one person in the elevator therefore, the identification algorithm should know, which face has just entered to the elevator and which is already inside. This can be done by keeping track on the face movements during the whole elevator cycle. When the boundary box is left door area, then it is possible to assume, that this face should not be detected as the entering one beside the doors and opposite.

The core of the automated face management system depends highly on the accuracy of the face detector since it is the only source of input data. Currently, if the face detector makes a mistake, the only way for capturing it relies on the condition inside first filtering after elevator cycle, that requires formed cluster size bigger than n (detailed description in sub-paragraph 4.2). It is definitely worth trying out if these false positive detections can be spotted using some kind of anomaly detection algorithm.

7 Conclusions

The goal of this thesis was to create automated face management system, which over time would learn to differentiate and recognize people directly from cameras installed inside the elevator together with an user interface, which enables to interact with the system and provides an overview of recognition results and real time events.

During the research, author did compare Mean-Shift, DBSCAN and Chinese Whispers clustering algorithms combined with three different feature extraction convolutional neural networks. After the experiments the best combination turned out to be Chinese Whispers together with Dlib face recognition model, considering the performance and execution time.

Integration with other system components was implemented using real-time operating system, which enabled asynchronous communication by default. To get an overview of the composed system in action, a web-based user interface was created, which enables configuring the face database, shows the events over the history and provides operator means to monitor the live system.

With composed methodologies the produced system managed to gather 133 people with a clustering accuracy of 99.2% by processing the recorded videos, that were saved and captured during the research. These results were then passed as a training input data for the face identification component implemented using the Radius Neighbor classifier, achieving the identification accuracy of 98.2%. Therefore, all the requirements for the thesis problem are satisfied.

References

- [1] A. K. Jain, K. Nandakumar and A. Ross, "50 years of biometric research: Accomplishments, challenges, and opportunities," *Pattern Recognition Letters*, p. 25, 2016.
- [2] J. D. WEST, "FaceFirst," FaceFirst, [Online]. Available: <https://www.facefirst.com/blog/brief-history-of-face-recognition-software/>. [Accessed 27 April 2019].
- [3] "GrandViewResearch," September 2018. [Online]. Available: <https://www.grandviewresearch.com/industry-analysis/biometrics-industry>. [Accessed 27 April 2019].
- [4] L. Siwik and L. Mozgowoj, "Server-Side Encrypting and Digital Signature Platform with Biometric Authorization," *Computer Network and Information Security*, vol. 4, pp. 1-13, 2015.
- [5] "Amazon," Amazon, [Online]. Available: <https://aws.amazon.com/rekognition/>. [Accessed 27 April 2019].
- [6] "Azure," Microsoft, [Online]. Available: <https://azure.microsoft.com/en-us/services/cognitive-services/face/>. [Accessed 27 April 2019].
- [7] "Kairos," Kairos, [Online]. Available: <https://www.kairos.com/features>. [Accessed 27 April 2019].
- [8] "Apple," Apple, [Online]. Available: <https://www.apple.com/lae/iphone-xs/face-id/>. [Accessed 27 April 2019].
- [9] S. Saha, "towardsdatascience," 15 12 2018. [Online]. Available: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. [Accessed 27 04 2019].
- [10] University of Stanford. [Online]. Available: <http://cs231n.github.io/convolutional-networks/>. [Accessed 2019 April 2019].
- [11] University of Toronto, "Artificial Neural Networks Technolog," [Online]. Available: <http://www.psych.utoronto.ca/users/reingold/courses/ai/cache/neural2.html>. [Accessed 27 April 2019].
- [12] THE UNIVERSITY OF TEXAS AT EL PASO. [Online]. Available: <http://wwwold.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/node12.html>. [Accessed 27 April 2019].
- [13] S. Albawi, T. . A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," IEEE, Antalya, 2017.
- [14] Axis, "Axis," [Online]. Available: https://www.axis.com/files/datasheet/ds_a1001_door_controller_t10091714_en_1901.pdf. [Accessed 30 April 2019].
- [15] Edmundoptics, "Edmundoptics," [Online]. Available: <https://www.edmundoptics.com/p/basler-ace-aca1440-73gm-monochrome-gige-camera/40324/>. [Accessed 27 April 2019].
- [16] Amazon, "Amazon," [Online]. Available: <https://www.amazon.com/Intel-Realsense-D435-Webcam-FPS/dp/B07BLS5477>. [Accessed 27 April 2019].

- [17] A. Kumar, "appliedmachinelearning," [Online]. Available: <https://appliedmachinelearning.blog/2018/10/30/yet-another-face-recognition-demonstration-on-images-videos-using-python-and-tensorflow/face-recognition-pipeline/>. [Accessed 27 04 2019].
- [18] J. Philbin, D. Kalenichenko and F. Schroff, "FaceNet: A Unified Embedding for Face Recognition and Clustering," 2015.
- [19] E. Learned-Miller, G. B. Huang, A. RoyChowdhury, H. Li and G. Hua, "Labeled Faces in the Wild: A Survey," *In Advances in Face Detection and Facial Image Analysis*, pp. 189-248, 2016.
- [20] D. E. King, "Dlib-ml: A Machine Learning Toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755-1758, 2009.
- [21] B. L. M. S. B. Amos, "OpenFace: A general-purpose face recognition library with mobile applications," CMU-CS-16-118, CMU School of Computer Science, 2016.
- [22] D. Sandberg, "Github," [Online]. Available: <https://github.com/davidsandberg/facenet>. [Accessed 27 April 2019].
- [23] A. Alemi, V. Vanhoucke, S. Ioffe and C. Szegedy, "nception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," arXiv, San Francisco, 2016.
- [24] "Tensorflow," [Online]. Available: <https://www.tensorflow.org>. [Accessed 30 April 2019].
- [25] "PyTorch," [Online]. Available: <https://pytorch.org>. [Accessed 30 April 2019].
- [26] A. Bijl, "A comparison of clustering algorithms for face clustering," University of Groningen, Groningen, 2018.
- [27] H. Yanagisawa, T. Yamashita and W. Hiroshi, "Manga character clustering with DBSCAN using fine-tuned CNN model," Society of Photo-Optical Instrumentation Engineers, 2019.
- [28] C. Biemann, "Chinese Whispers - an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems," University of Leipzig, Leipzig, 2006.
- [29] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, "226 KDD-96 A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," University of Munich, München, 1996.
- [30] Developers of Scikit-learn, "scikit-learn," [Online]. Available: <https://scikit-learn.org/stable/modules/clustering.html#dbscan>. [Accessed 27 April 2019].
- [31] K.-L. Wua and M.-S. Yangb, "Mean Shift-Based Clustering," *Pattern Recognition*, vol. 40, no. 11, pp. 3035-3052, 2007.
- [32] s.-l. developers, "scikit-learn," [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MeanShift.html>. [Accessed 27 April 2019].
- [33] A. F. M. Agarap, "An Architecture Combining Convolutional Neural Network (CNN) and Support Vector Machine (SVM) for Image Classification," ArXiv , 2017.
- [34] P. J. Phillips, "Support Vector Machines Applied to Face Recognition," *Advances in neural information processing systems*, vol. 2, pp. 803-809 , 1999.

- [35] Y. Tawil, "allaboutcircuits," 26 05 2017. [Online]. Available: <https://www.allaboutcircuits.com/technical-articles/an-introduction-to-robot-operating-system-ros/>. [Accessed 27 04 2019].
- [36] C. Technologies, "Medium," 26 April 2018. [Online]. Available: <https://medium.com/cuelogic-technologies/top-3-best-javascript-frameworks-for-2019-3e6d21eff3d0>. [Accessed 27 April 2019].
- [37] N. Foundation, "NodeJS," [Online]. Available: <https://nodejs.org/en/about/>. [Accessed 27 April 2019].
- [38] J. Reschke, "The 'Basic' HTTP Authentication Scheme," Internet Engineering Task Force, 2015.
- [39] S. Symanovich, "Norton," Norton, [Online]. Available: <https://us.norton.com/internetsecurity-iot-how-facial-recognition-software-works.html>. [Accessed 27 April 2019].
- [40] Pyimagesearch., "Pyimagesearch," 09 July 2018. [Online]. Available: <https://www.pyimagesearch.com/2018/07/09/face-clustering-with-python/>. [Accessed 27 April 2019].

Appendix 1 – Software repository

The code for the project was stored in <https://gitlab.pld.ttu.ee/Rait.Kurg/master-thesis.git>. This repository has internal access level limited only to people with TalTech Uni-ID as it contains sensitive face database and test data of individuals.