NTNU
Norwegian University of
Science and Technology

The Intelligent Mirror

A Personalized Smart Mirror Using Face
Recognition

Johannes Moskvil

Master of Science in Computer Science
Submission date: June 2017
Supervisor: Asbjørn Thomassen, IDI

Norwegian University of Science and Technology
Department of Computer Science

Johannes Bredholt Moskvil

A Personalized Smart Mirror Using Face Recognition

Master's Thesis in Computer Science, Spring 2017

Data and Artificial Intelligence Group
Department of Computer and Information Science
Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology

Abstract

Smart mirrors are a new addition to the smart product family that has been getting a lot of attention in recent years by both commercial manufacturers and hobbyists. This thesis explores enhancing such mirrors with intelligence. The goal is to develop a user recognition system that is able to differentiate between the users of the system by using a camera hidden behind a two-way mirror.

The recent resurgence of deep neural networks have pushed the fields of computer vision and face recognition to create powerful neural networks that by far outperforms the methods that was before. The Raspberry Pi(RPI) will be used as the embedded computational device for the intelligent mirror. The question is if it is able to utilize the power of the new face recognition methods or if it is too slow. This thesis will compare a various of face detection and face recognition methods to explore if it is possible to build a satisfactory user recognition system on the RPI. Therefore I propose two different architectures for the intelligent mirror. Either a local architecture that only uses the RPI, or a remote architecture where the RPI simply forwards the camera stream to a remote device that computes the results.

The results obtained makes it clear that a user recognition system employing deep learning is today way too computationally complex for the RPI to be a suitable device. Using older methods resulted in a barely acceptable user recognition system that can be used, but will often predict wrong user.

When keeping in mind that user recognition is just the first step in developing the intelligent mirror the conclusion is that the system should either use a remote architecture or look for more powerful embedded devices with hardware capable of feed-forwarding deep neural networks.

Sammendrag

Smart speil er en ny tilføyelse av smart producter som de siste årene har samlet mye oppmerksomhet fra både kommersielle aktører og i fritidsprosjekter. Denne masteroppgaven utforsker hvordan slike speil kan bli tilført intelligente aspekter. Målet er å utvikle et brukergjenkjenningssystem som er i stand til å gjenkjenne forskjellige person ved å bruke et kamera gjemt bak et tosidig speil.

Den fornyete interessen og suksessen av dype neurale nettverk har gjort at datasyn og da spesielt ansiktsgjenkjenningsmetoder langt på vei har overgått hva tidligere metoder var i stand til. Raspberry Pi vil bli brukt som en innebygd datamaskin for det intelligente speilet. Spørsmålet er om den er i stand til å ta i bruk de nye metodene innen ansiktsgjenkjenning med tanke på dens regnekraft. I denne masteroppgaven vil jeg sammenligne forskjellige ansiktsdeteksjons og ansiktsgjenkjennings metoder for å utforske hvilke metoder som kan bli brukt på RPI i en tilfredsstillende grad. Derfor foreslåes to programvarearkitekturer for the intelligente speilet: en lokal arkitektur som kun bruker RPI, og en fordelt arkitektur der RPI vil strømme video fra kamera til en annen datamaskin som vil gjøre de nødvendige utregningene.

Resultatene gjør det klart at et brukergjenkjenningssystem som bruker dyp læring er i dag altfor avansert utførelsesmessig for at en RPI skal være en egnet enhet. Ved å bruke eldre metoder kunne systemet kjøre på en RPI, men treffsikkerhetstestene viser at det er vanskelig å få disse metodene treffsikre nok til at systemet er brukbart.

Når man også husker på at brukergjenkjenning bare er den første tilføyelsen til det intelligente speilet, så er konklusjon systemet enten burde bruke en fordelt arkitekture eller utforske kraftigere enheter enn RPI'en som vil være i stand til å kjøre dype nettverk.

Preface

This thesis is the result of my Master of Technology degree at the Norwegian University of Science and Technology. I would like to thank my supervisor Asbjørn Thomassen for helping me acquire all the necessary components and costs for creating a prototype intelligent mirror.

Johannes Bredholt Moskvil
Trondheim, 11th June 2017

Contents

List of Figures

List of Tables

1 Introduction

A smart mirror is a mirror that has been enhanced by technology. It is a very simple concept that is illustrated in figure 1.1. A screen is placed behind a two-way mirror [1]. By using a black and white graphical user interface (GUI) only the white colors will penetrate the mirror, resulting in an effect that makes it appear as if the mirror itself is a screen. The idea of an *intelligent mirror* is to further this enhancement to include artificial intelligence as well.

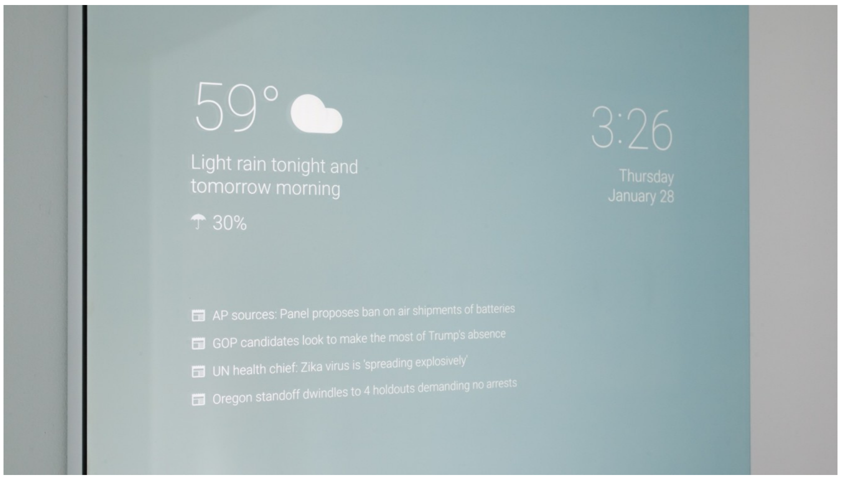

59°
Light rain tonight and tomorrow morning
30%
3:26
Thursday
January 28
AP sources: Panel proposes ban on air shipments of batteries
GOP candidates look to make the most of Trump's absence
UN health chief: Zika virus is 'spreading explosively'
Oregon standoff dwindles to 4 holdouts demanding no arrests

Figure 1.1: Illustration of what a smart mirror is

The ideas as to what an intelligent mirror could encompass is vast. It could encompass anything related to intelligence. It could be a mirror that recognizes people, talk to people, learns a person's habits, it could be used as a component in activity recognition as part of a smart home. Or as it is ultimately a tool used by people to see their own reflection - it could be used to analyze the emotions of the person in question. This could further be used as input for a machine learning algorithm to predict the person's current level of happiness. Or even monitor his emotions over an extended period of time to detect mental illnesses. Because a mirror is naturally used by most people at least twice a day (morning and night), it would be able to continuously monitor people's faces without requiring any explicit input.

The major advances done in the fields of computer vision in recent years makes the acts of facial recognition easily available to any software developer through frameworks.

[1]A two-way mirror is a mirror that does not reflect 100% of the light. In other word a mirror that can be seen through. More on this is chapter 5

The resurgence of neural networks has played its part; allowing previously complex computational methods to be used by cheaper hardware available to the general public.

As such the aim is to build an intelligent mirror that can recognize people and use this to tailor the experience for each individual user. In this iteration of the project the goal is to create a user recognition system that will show a different user interface for each user.

The Raspberry Pi (RPI) will be used as an embedded device to capture video from a camera hidden behind the mirror. Further we will explore if it is possible to build a user recognition system that can run at a satisfactory level on the RPI. Alongside that a remote processing architecture will be proposed in case running the user recognition system locally on the RPI does not produce satisfactory results.

Finally a physical prototype of the intelligent mirror will be built.

1.1 Background and Motivation

This thesis is motivated by the influx of smart products in today's market. There are smart phones, smart homes, smart lights, smart windows blinds - every common item is made smart. Internet of things has come to stay. Every item used in our homes are today connected to the Internet. Mirrors are everyday items that have so far eluded being made smart. This project aims to create a smart mirror, but also going an additional step to enhance it with some sort of intelligence. However, intelligence can encompass anything, from speech recognition systems, health monitoring systems, recommendation systems to user recognition system. The subfield of artificial intelligence that I have opted to explore in this thesis is that of user recognition in video. The domain of this thesis will therefore largely overlap with that of computer vision. It is also motivated by my desire to create a physical technical product and the required software, rather than a thesis consisting solely of programming.

1.2 Goals and Research Questions

As the introduction have explained; the creation of an *intelligent mirror* could mean anything. In this section the main goal of the thesis will be formulated as well as various subgoals we aim to answer by the conclusion.

Goal *Build a physical mirror enhanced with artificial intelligence.*

This involves physically building a device so that people seeing it will think of it as an *intelligent mirror* rather than a smart mirror. Of course, developing something deemed intelligent is the goal of the whole field of artificial intelligence, so the specific areas of artificial intelligence explored will be expanded in the following research questions.

Research question 1 *Explore the existing implementations of intelligent mirrors. If any, what types of artificial intelligence do they apply?*

Research question 2 *Can we build a user recognition system that can run on the RPI at an acceptable level?*

Research question 3 *Propose an alternative system architecture if research question 2 is unsuccessful.*

Research question 4 *Build a physical prototype of the intelligent mirror.*

The rest of the thesis will focus on thoroughly answering each of these research question. By the end the goal is to have created a prototype intelligent mirror that is able to respond to different persons using the mirror.

1.3 Research Method

In simple terms recognizing a person from a video is to: find the moving objects $\rightarrow$ analyze those objects for faces $\rightarrow$ predict who the face belongs to.

Most of the experiments conducted will be aimed at evaluating this pipeline. Each step has multiple possible methods that could be used. The aim is to find a combination of the algorithms that is able to consistently recognize the most users correctly. While simultaneously minimizing the computational cost in order to run the system on the RPI. This is a tradeoff between accuracy and execution speed.

In order to evaluate the user recognition system a test set of videos of people mirroring themselves will be gathered. The user recognition system should be highly modular so each component of the pipeline could be substituted by various methods at ease. The test set will serve as a benchmark for the various pipeline configurations in order to make a conclusion as to what methods will best serve our purpose.

1.4 Thesis Structure

Chapter 2 will showcase various usage scenarios for an intelligent mirror, in order to better understand what an intelligent mirror could encompass. Chapter 3 will go in-depth of the various algorithms that exists for each step in our user recognition pipeline. Our findings will be used to implement the modular user recognition system. Chapter 4 will explored the related works to intelligent and smart mirrors. Both commercial project and research projects will be discussed. In chapter 5 the user recognition system's architecture will be proposed and explained. Additionally chapter 5 will discuss the physical hardware components necessary to build the mirror. Chapter 6 will go in-depth of the results of the user recognition system when evaluated on the collected test data. Chapter 7 will wrap up the thesis and make any conclusion to the results obtained in the previous chapter if possible. Appendix A will thoroughly explain the process of building the mirror step by step. It will show what the end product look like and hopefully serve as a handy tutorial for anyone interested in creating their own intelligent mirror.

2 Scenarios

2.1 Usage Scenarios

To better understand the purpose and possibilities of an intelligent mirror, this section will introduce a few different scenarios detailing some hypothetical situations. Only a few of these were implemented and used in this iteration of the project. However as I had written many scenarios they are presented here to both: inspire ideas for future work on this project, and as references to the possibilities an intelligent mirror entails. The smart mirror's primary purpose is to expose information that its users can quickly and effortlessly make use of. This is achieved by having a modular software structure where a user can arrange his preferred modules to his preferences. This could be modules that show the time, bus departures, calendar, emails and so on.

Configuration swapping based on Facial Recognition

A user has configured facial recognition for his mirror. His name is John and his wife is named Jane. Both of them have setup their configuration profile and supplied training images of their faces that the mirror has used to learn to differentiate and recognize the two of them. Now the mirror will load the modules John would like to use when he is using the mirror, and likewise, the mirror will load Jane's profile when she is using the mirror.

The mirror hangs in their entrance as they use it to mirror themselves before going out. John goes to work by bus, hence he has configured his profile to include a bus module that shows the next departure time for the bus. Because the mirror recognizes him automatically he can quickly check to see if he is on time to catch his bus, without having to spend time taking out his phone, open up the bus app, navigate to the correct bus and stopping place; to check when it will come. As Jane travels to work by train, she has configured her profile to include a module that shows the next train departures. Because she is also recognized by the mirror, she only needs look in it for the mirror to show her the departure time of the next train. The mirror is also set to a default configuration for strangers. If it does not detect the presence of a person, it will turn off and just look like a mirror. These four scenarios are summarized in figure 2.1

Expand set of recognized persons by speech interaction

The intelligent mirror is envisioned to be a product used on the same line as smart phones. This means that it must be easy to use and customize for the average user who does not know how to configure something with programming:

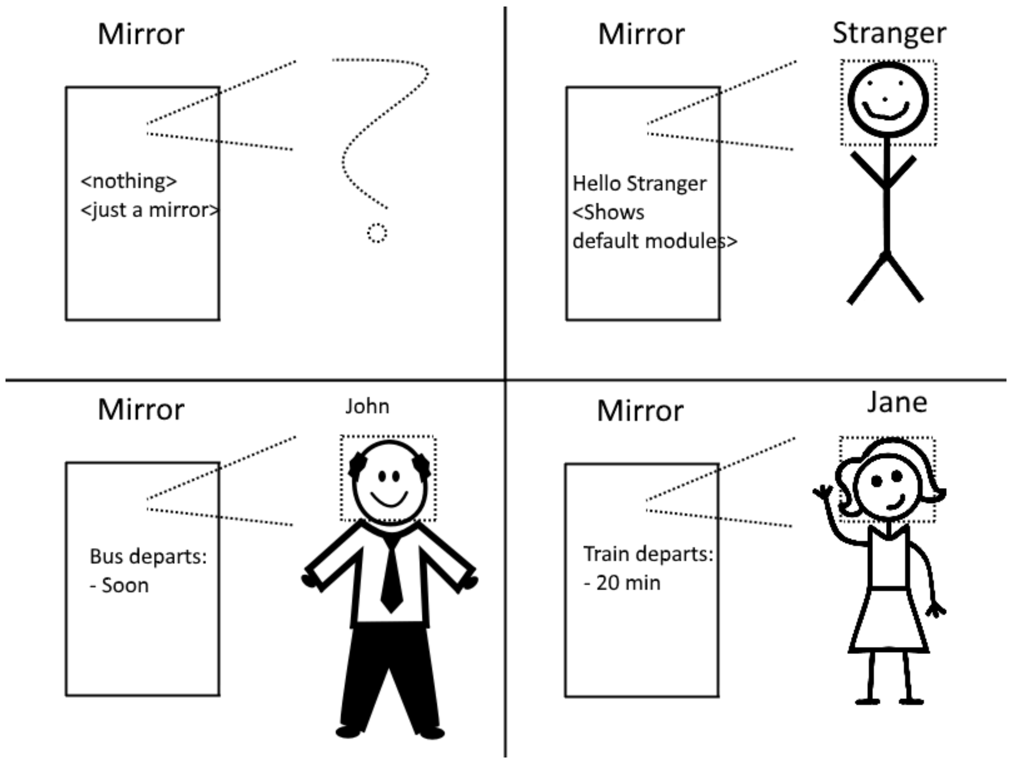


Figure 2.1: Illustration of the various scenarios for interface customization based on module swapping.

John's mother often comes over to visit. She is not yet recognized by the mirror, but it would be useful for her to quickly check the departure times for her bus when leaving for home. Through voice commands the mirror could learn to recognize people it do not yet know. Figure 2.2 outlines this scenario. The mirror does not recognize John's mother, so therefore it shows a default configuration for strangers. However, it is listening for commands to initiate the procedure of adding more people to the its set of recognized persons. At this stage, saying "hello" to the mirror will initiate the procedure for adding new persons to its set of recognized people. The mirror will proceed to ask for the new users name, take an amount of images, then retrain its model. Now the new person is included in its model and it will recognize this new person. I would like to point out that there is a jump between the bottom two frames in figure 2.2. The profile for this new person would still need to be configured with an app in order for the bus module to be correctly configured for this new user.

Navigation by gestures

It can be desirable to be able to interact with the mirror rather than it just being a static display of information. Most smart devices utilizes touch functionality for interaction. However, as this is a mirror other methods are warranted to not cover the mirror in

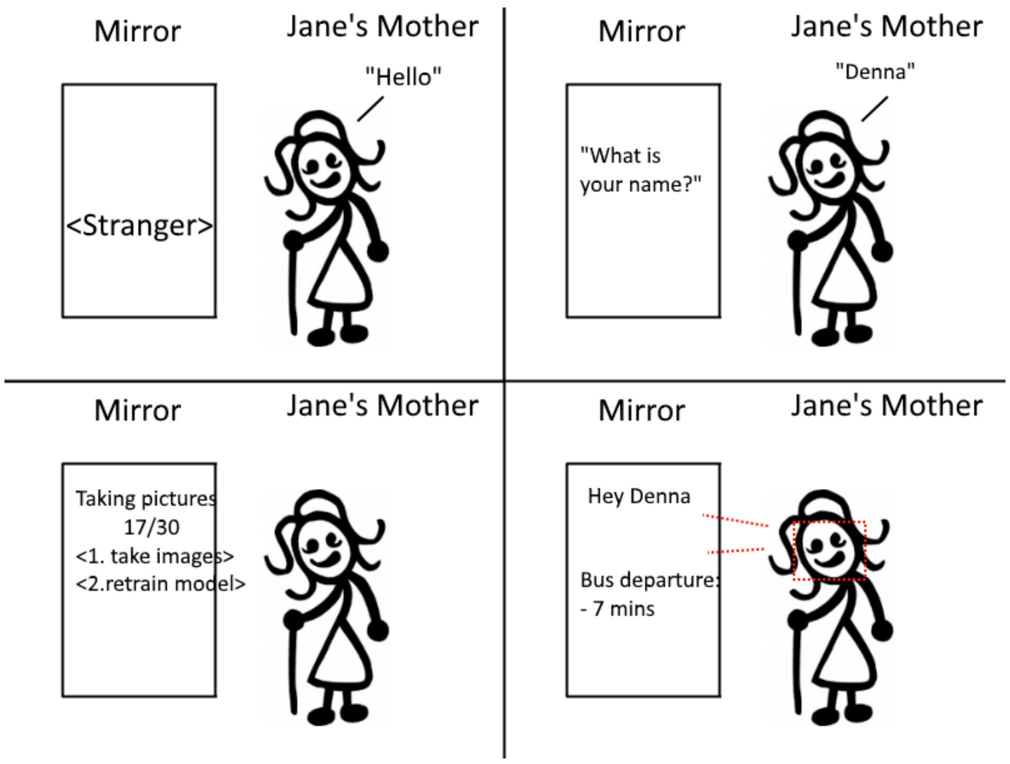


Figure 2.2: Steps explaining how the mirror can learn to recognize new people with minimal interaction. The text enclosed by angle brackets symbolize inner mechanisms of the mirror, else the text symbolizes what is actually shown in the UI.

fingerprint smudges.

The graphical display offer a limited graphical area. Consequently this will limit the functionality it can provide as it can only cram so many modules to show within that display. By offering methods for interaction - such as speech control and gestures - the user could initiate the mirror to show information at his request. The following scenarios will outline some situations where interactions either through speech or gestures are useful.'

Again we have John using his configured mirror. He is utilizing a great number of modules to provide the functionality he needs: bus schedule, calendar overview, e-mails, stock prices, clock, weather forecast, and of course his daily joke. Displaying all this information at once would clutter the screen and not be very aesthetically pleasing. Therefore he uses the swiping gestures to scroll through the active modules. The modules are arranged as a traditional carousel used in software user interfaces. Items are arranged in a view where the user can scroll left or right to show the next item. When at the end of the list it will scroll to the beginning. Hence why it is known as a carousel. When John makes a swiping motion towards the left the mirror recognizes this movement as a desire to shift the active module to the left - making the next module to the right the

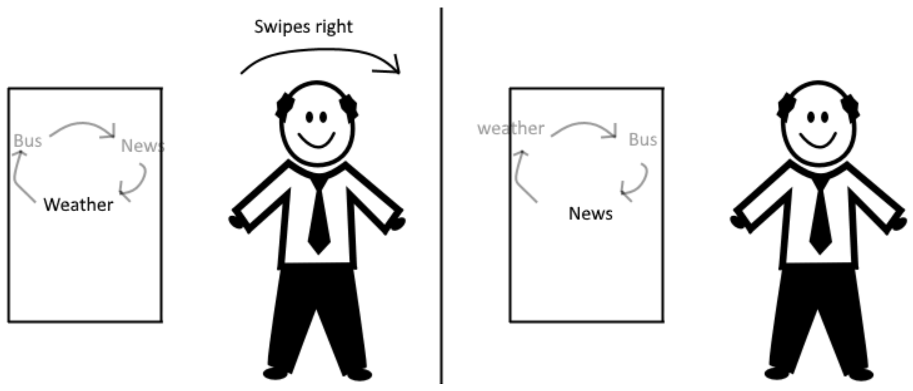


Figure 2.3: Illustration of swapping modules with swiping gesture.

new active module. This way John can make use of more modules than what is possible to display on a single screen.

Of course gestures are not limited to just left and right swiping motions. Up and down swiping motions could also be used to swap modules. This would require that the modules are laid out in a spherical grid rather than a carousel. Zoom-in and zoom-out pinch gestures could be used to enlarge or shrink certain graphics.

Navigation by speech

Another touch-free way of interaction is through speech. Speech can also help the mirror appear as more intelligent than it really is - as communicating through semantics is something that is inherently human, and consequentially regarded as a sign of intelligence. It also makes for a very practical way to interact with the mirror as it is so easy for humans to speak.

Each module could be assigned to a textual token. John has too many modules for it to be practical for the mirror to show them all. Therefore he has configured his mirror to show only the module bound to the word he speaks. On his way out the door he stops by his mirror. The mirror is still blank and shows nothing - as it has not been instructed to show anything yet. John wishes to quickly check his email and bus departures before leaving home. He speak "email" which the mirror recognizes as a desire for John to read his email. The mirror still employs facial recognition; so it knows that John is the person using the mirror, and is then able to show the email module connected to John's email account. After reading his email he speaks "bus". This time the mirror recognizes this as a request to display the bus departure module. John notices the bus leaves in a few minutes and hurries out the door. The mirror recognizes there are no people in front of it and continues to turn off its display; returning to being just a mirror.

Speech recognition could also be used to interact directly with the modules. For example, a weather forecast module could listen for input as to where to look up the weather. John's weather module is currently set to show the weather forecast for the town he lives in. Today, however, he is travelling abroad on a business arrangement and, on his way out, wants to quickly check the weather. First he stands in front of the mirror. The mirror recognizes John, and loads his configurations, but does not yet display anything. John speaks "weather" which the mirror recognizes as a request to display the weather module. The mirror displays the weather forecast for the usual town. But today John needs to know the weather for another location, so he speaks "weather, Berlin". The mirror recognizes this as a request to display the weather forecast for Berlin.

Long-term emotion analysis

The previous scenarios have been focused on use cases that are convenient and directly useful for the user. However the mirror could also play a more indirect role. One example is the use of the mirror as a medical tool. Most people uses a mirror at least twice a day: morning and evening. By extending the use of facial recognition this could be used to monitor and analyse the users in order to detect psychological illnesses such as depression or bipolar disorder. By using facial recognition, the mirror is able to capture a few pictures of each user daily and sort them in folders for each user. This way testing data is generated without requiring any explicit input from the users. Given that these images are analyzed for emotions over a timeseries of images - that could serve as input for a machine learning algorithm to attempt to recognize symptoms of psychological disorders. Of course, this is a huge research question and could serve as basis for another thesis all by itself. This example is meant to emphasis the possibilities in the context of an intelligent mirror.

Distributed interconnected mirrors

It is also possible to envision the use of an intelligent mirror in a bigger commercialized setting. This scenario is based on the functionality outlined in the first scenario explained. The intelligent mirror could be distributed so that all the mirrors share the same facial recognition model. This way, any user would be able to use any mirror as his own. Such a distributed model could be applied to the hotel business. Say all hotel rooms had a smart mirror that was connected to the shared data center used by all mirrors. The mirror would then activate whenever said businessman enters his hotel room and have instantly access to all his configured modules. Of course this is a huge challenge. It requires that the facial recognition service is able to perfectly differentiate between thousands of users with absolute accuracy. This scenario also contradicts the type of facial recognition explored in this thesis - how to get best recognition given that the pool of users is small. One step toward making such a product work is to have two layers of security instead of one. By for example applying voice-to-person recognition on top of facial recognition. This scenario is not considered in this thesis. But it is certainly an interesting idea from a commercial standpoint.

Summary

The focus of this thesis will be on the first scenario: *Configuration swapping based on Facial Recognition* The other scenarios are left in as inspiration for further work. I would like to note that I did make a software module to the system for the second scenario: *Expand set of recognized persons by speech interaction.* But I decided to scrap bringing any further attention to it in this report as it was simply a wrapper around Google's amazing speech-to-text API that was tailored so simple voice commands would initiate a process that would automatically make the system recognize new users. [1] While I think it is undeniably a cool feature, the rest of this thesis will focus fully on the user recognition scenario.

[1]Google Cloud Speech API (GCS)

3 Background Theory

This chapter will look into the current state-of-the-art technology relevant to the AI methods that could potentially be used to make the mirror more **intelligent**. The chapter will especially explore computer vision techniques within face detection, face recognition, extracting facial landmarks, align faces, and various classification techniques.

Many of the techniques described below have been implemented by the open source libraries dlib and OpenCV. It is not the goal to implement these from scratch, but rather find out what combination of known techniques or enhancement of them can be used to make a video based user recognition system run on an embedded device. In this case the embedded device is the Raspberry Pi 3 Model B. The corresponding library will be listed for each technique as they are introduced.

3.1 Moving Object Detection

The mirror uses a camera as its input source and it is the users face that provides the region of interest. The first step in the pipeline given (see figure 3.1) is to detect new moving objects - objects that was not there before. The most basic way to differentiate moving objects in a video feed is to use a single image as a *reference* frame. Moving objects are then detecting by doing a pixel by pixel subtraction proceeded by applying a threshold to create a binary image where black pixels indicates no change and white pixels indicates a change - or movement.

Algorithm 1 simple background subtraction

```
image$_{video}$ ← newest frame
image$_{reference}$ ← reference frame
for i in range(image$_{size}$) do
    pixel$_{old}$ = image$_{video}$[i]
    pixel$_{ref}$ = image$_{reference}$[i]
    pixel$_{new}$ = 0
    if $|pixel_{old} - pixel_{ref}| > threshold$ then
        pixel$_{new}$ = 1
    end if
end for
```

This technique does however have a few shortcomings. Because the reference frame is never updated, any *newly added items to the scene* will trigger as motion - even though they are just stationary objects meant to stay where they are. Likewise different illumination settings will result in different pixel values, even though the scene is the same. The simple background detection algorithm would also see this motion. A solution to this is to transform the reference frame into a running average of a given number of previous frame. This way motion will decay and stationary objects added to the scene will after a short while be considered as part of the background.

Naturally these techniques will not work if the camera itself is moving. But as a mirror is stationary - simply hung on a wall - these techniques should be suitable. Even if the mirror is relocated to another room; the reference frame will eventually accommodate to the new room's background. There are different ways to update the reference frame. In Christopher Richard Wren et al. (1997) they proposed a method of recursively update the reference frame using an adaptive filter shown in formula 3.1.

$$u_t = \alpha \cdot y + (1 - \alpha) \cdot u_{t-1} \tag{3.1}$$

Where u_t is a given pixel at time t, y is the pixel from the new image, and α is a constant that regulates the update speed - how fast the model forgets about previous images. This filter function is then run for each pixel in an image to recursively update the reference model. Such a background reference model is implemented in openCV 2.4 as **accumulateWeighted**.

Other background subtraction methods include Stauffer and Grimson (1999) who used a mixture of gaussian distributions to model each pixel, rather than modeling each pixel's value as one particular type of distribution. A heuristic is then applied to predict which pixels are likely to belong to the foreground.

A problem with the previous approaches is that moving objects often cast a shadow. The shadow will cause the pixel values to differ from the reference frame and be classified as motion, even though the shadow is not part of the object itself. KaewTraKulPong and Bowden (2001) remedy this by using a color model that can separate chromatic and brightness components [1]. Pixels that has been classified as foreground pixels are then measured against the reference pixels again. If the difference in both chromatic and brightness components are within some thresholds, the pixel is considered as a shadow. Their paper is implemented as **BackgroundSubtractorMOG** in OpenCV 2.4.

Additionally OpenCV provides the **BackgroundSubtractorMOG2** implementation of the papers Zivkovic (2004) and Zivkovic and van der Heijden (2006). It provides better adaptability to variations in the scene by adapting the number of gaussian distributions for each pixel, rather than using a fixed amount for every pixel as in other algorithms.

However, as the intention of this project is to attempt to run the face recognition procedure on the Raspberry Pi 3 Model B, there is a trade-off between execution time and accuracy. This trade-off on the RPI will be discussed more in detail in chapter 6.

[1]In a HSL color model, chromaticity consists of the saturation and hue components and brightness corresponds to the lightness component

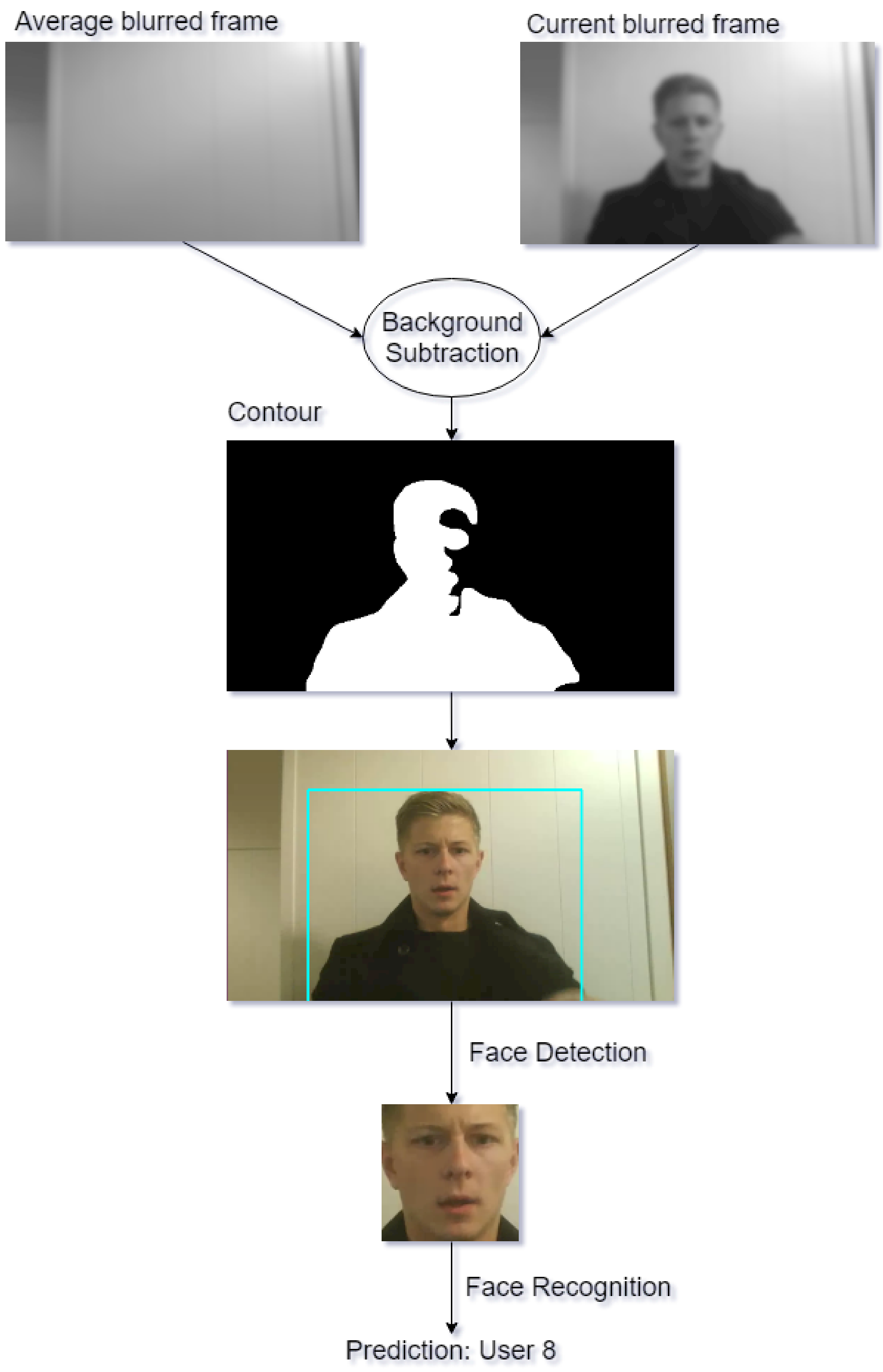


Figure 3.1: The user recognition pipeline

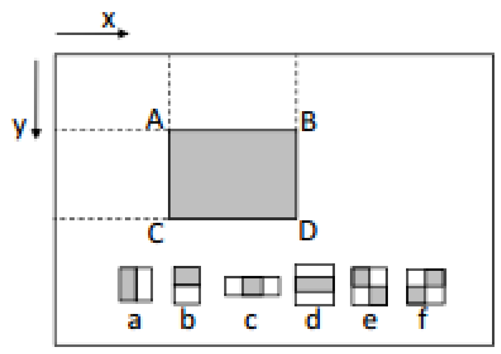
3.2 Face Detection

The next step is to detect which part of the moving object corresponds to a face. To reiterate: face *detection* is the task of finding faces in an image - disregarding who the person is - face *recognition* is the task of mapping the face of a person to their identity.

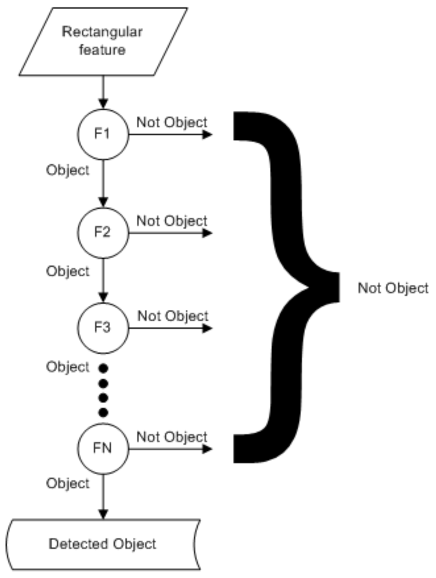
In the early 2000s the prevalent methods for face detection were *model-based face tracking* and *weak-classifier cascade*. Model-based methods employs edge detection and geometric models to locate the face. The most notable contributions to model-based methods was developed by Froba and Kublbeck (2001) and Jesorsky (2001).

However the big breakthrough that allowed for face detection at multiple frames per seconds even on cheap hardware was the use of training a haar-cascade classifier for face detection first proposed by Viola and Jones (2001).

The idea proposed is to train a cascade of haar-features for face detection. Examples of haar-features are given in figure 3.2a. Each feature corresponds to a single value that are calculated summing the pixel values of the pixels in the black area, and subtract the sum of pixel values from the white area. The haar-features can be compared to that of convolutional kernels, where the black rectangle signify 1's and white rectangle -1's. By applying the kernels at all possible locations and of multiple different size, many features can be calculated. However this is very computationally expensive, the huge success of the Viola & Jones method rely on it's fast execution speed. To remedy this Viola & Jones introduce the integral image and the concept of cascade of classifiers. The integral image only need to be calculated once and it allows calculation of the sum of pixel values within a rectangle to be done in $O(1)$.



(a) Illustration of the integral image (A,B,C,D) and a collection of haar features(a-f)



(b) Illustration of cascade classifier

Figure 3.2: Viola and Jones (2001) novel ideas for creating a real-time face detector

The method uses a sliding window over the image. For each window the algorithm asks "is this a face?[2]" Viola & Jones' algorithm employs Adaboost. Adaboost is a classification algorithms that combines many weak classifiers to create a strong classifier. For each haar-feature the algorithm learns a threshold value to minimize classification errors. These weak classifiers are then ordered by their weight, so the most distinguishing classifiers are tested first, hence the **cascade of classifiers** (figure 3.2b). Rather than calculate all features for classification at each sliding window frame, only the first few are calculated. If they fail the classification test the frame is discarded and the sliding window moves onto the next frame. Viola & Jones' classifier employed 6000+ features with 38 cascade stages, starting with 1, 10, 25, 25 and 50 features for the first five cascades. On average only 10 features out of the 6000+ are evaluated each sub-window.

Other methods for face detection includes the usage of histogram of oriented gradients (HOG) and local binary patterns (LBP).

Liao et al. (2007) proposed the use of Multiscale Block Local Binary Pattern (MB-LBP) as features for face detection. Like the Viola & Jones method it uses AdaBoost to select MB-LBP features - rather than haar features - for face classification. LBP are texture descriptors. For every pixel in the image its corresponding LBP value is calculated and replaced in a new image (so that other pixels will still use the original pixel values in their LBP calculation). A 3x3 square is chosen around the pixel to be calculated, every surrounding pixel is thresholded against that pixel, resulting in either 0 or 1. A 8-bit number is then calculated by going clockwise around the neighbours. It doesn't matter where on the circle it starts, as long as it is consistent throughout. A histogram with a minimum of 0 and max of 255 is then computed from the LBP pixels and this is the feature descriptor. However a pixel-by-pixel 3x3 neighbourhood does not capture larger structures that may be dominant features of faces. Liao et al. (2007) therefore suggest using LBP on averaged sub-regions rather than individual pixels, hence Multiscale Block LBP's. Likewise to Viola & Jones it utilizes the integral image to efficiently compute the average sub-regions.

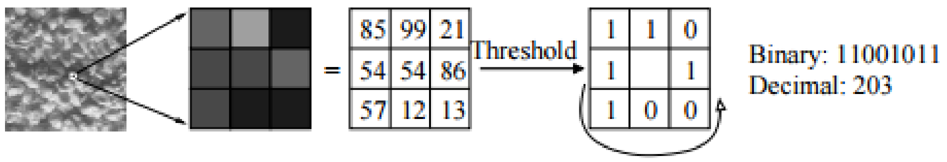


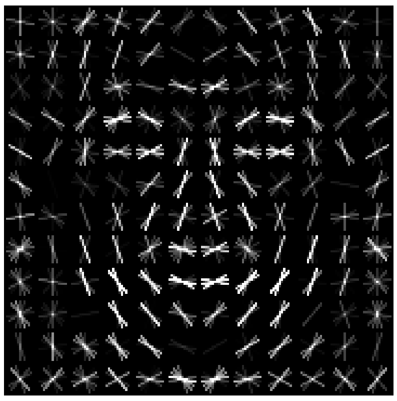
Figure 3.3: Illustration of the general LBP principle. Image from Ahonen et al. (2006)

The main advantage with using LBP features over haar features is significantly faster training speed (hours rather than days). It is also many times faster than haar as it operates on integers instead of floats, making it more suitable for embedded devices than the haar cascade approach.[3]

[2]or whatever object it is trained to detect

[3]Our experiments in chapter 6 reaffirms this statement.

Dalal and Triggs (2005) trained a linear SVM classifier to recognize persons based on HOGs. The general principle behind HOG features is that though the actual values of the pixels in an image are highly depended on lighting conditions, the direction of lighting changes will be the same both for really dark and really bright images. On a pixel level, its gradient is computed by looking at all adjacent pixel and create a gradient in the direction the image is getting darker for that pixel. However, doing this at a pixel level provides too much detail. Instead the image is divided into smaller sub-images (for example 16x16 pixels) that combines the individual gradients of the pixels into a single most prominent gradient for that 16x16 square. By generating a general HOG pattern of the desired object - in our case faces - that object can be recognized in images by comparing the image's HOG to the object's general HOG pattern.


(a) Human face HOG detector

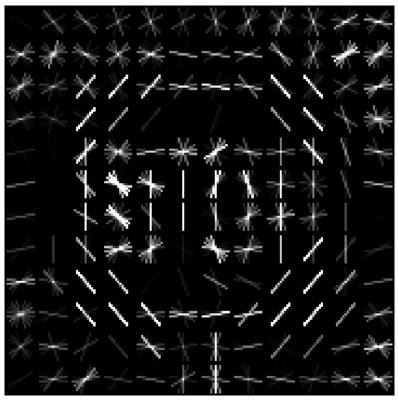

(b) Stop sign HOG detector

Figure 3.4: Example trained HOG detectors from the dlib documentation.

Pre-trained cascade classifiers for both haar-like features and LBP are provided in OpenCV. While the dlib library provides an implementation of HOG face detector.

3.3 Face Alignment and Landmark Extraction

After the face has been detected it is normal to go through another pre-processing step called alignment. Alignment is important because two images of the same person viewed from different angles or expressing different poses can look completely different to a computer. Alignment algorithms tend to this problem by skewing and rotating the image such that the eyes and other important facial regions are closely fixed to the same location within each image.

A well performing state-of-the-art algorithm for alignment was developed by Kazemi and Sullivan (2014). By using an ensemble of regression trees they were able to achieve face alignment in real-time with high accuracy. The algorithm will find facial landmarks as demonstrated in figure 3.5. When the facial landmarks have been identified, simple affine transformation (transformations that preserve parallel lines) can be used to standardize the face within the frame. Dlib contains an implementation of Kazemi and Sullivan (2014) paper.

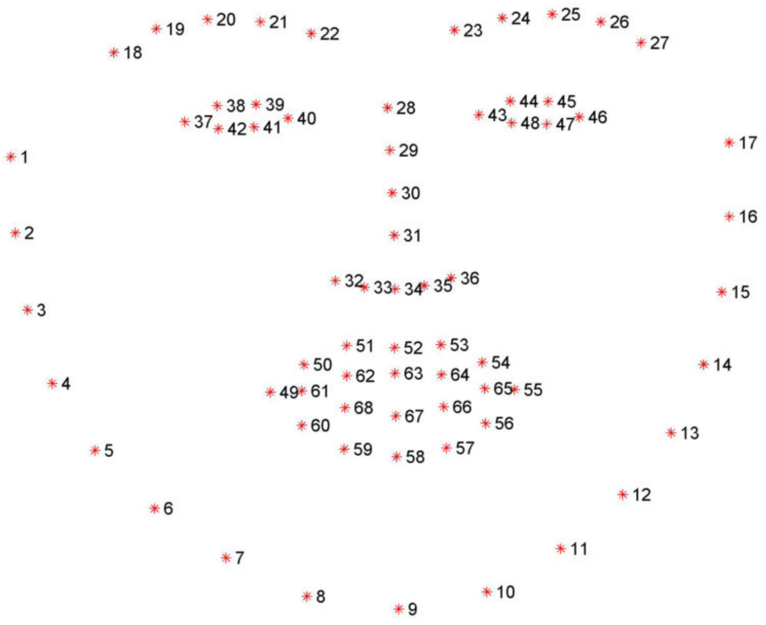


Figure 3.5: 68 facial landmark coordinates from the iBUG 300-W dataset from which dlib's implementation of Kazemi and Sullivan (2014) is trained on.

Simpler alignment techniques can also be applied by using the eye landmarks to rotate the image such that the line between the average eye height becomes horizontal.

3.4 Face Recognition

After the face region has been detected - and potentially aligned - the next step is to recognize who this person is. This is a two-step process. First a set of informative features must be extracted from the raw-pixel data of the face. These features are then used to train a classifier. Training a classifier is pretty straight-forward. Converting raw-pixel data(high dimension) to a set of discriminating features(low dimension) has for a long time challenged researchers.

One well known dimensionality reduction algorithm that has been successfully applied to a wide variation of problems is the Principle Component Analysis(PCA) algorithm. The PCA finds a linear combination of features that maximizes the variance of the data. This algorithm is used in constructing *Eigenfaces*, which was used in the first successful face recognition method in Turk and Pentland (1991). Eigenfaces is however very susceptible to changes in background and lighting in an image, and the authors already expressed concern for these weaknesses in the original paper. Some years later in 1997 Belhumeur et al. (1997) came up with another feature extraction method called

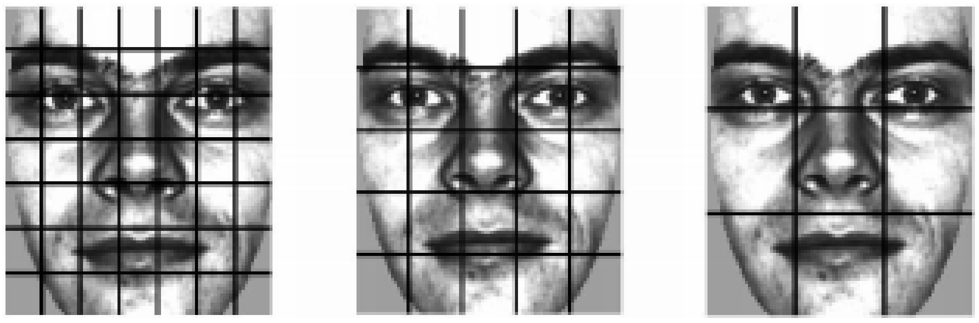


Figure 3.6: Face image divided in sub-squares. Ahonen et al. (2006)

Fisherfaces. This method is more robust than the Eigenfaces method and much more resistant to lighting and pose variations.

Ahonen et al. (2006) proposed using LBP as a feature descriptor for recognizing faces. The idea is to build a global feature vector from local LBP descriptors. The face image is divided into a number of sub-regions. The LBP histogram of each sub-image is then computed and concatenated (rather than merged) to create the feature vector.

In recent years however, both of these methods are severely outperformed by the introduction of *Deep Learning*. One breakthrough state-of-the-art method was developed by Schroff et al. (2015) at Google. Their algorithm learns a convoluted neural network that will output 128 measurements given an input image. These 128 features are also often called **embeddings** of an image. Google's faceNet method works by feeding the image a triplet of images. The first two images are two different images of the same person, while the third is a different person. The network will then tweak its weights such that the similarity measurement between the first two images are as alike as possible, while making the difference between the second and third image as large as possible. This process is illustrated in figure 3.7. The downside with this method is that it requires millions of training examples to output reliable discriminating features. However, once a network is trained the weights can simply be saved and loaded. When the network is trained, running images through the network is fast and straightforward. What the 128 features actually represents is impossible to know, but it is whatever the machine learning algorithm discovered to be significant discriminating features.

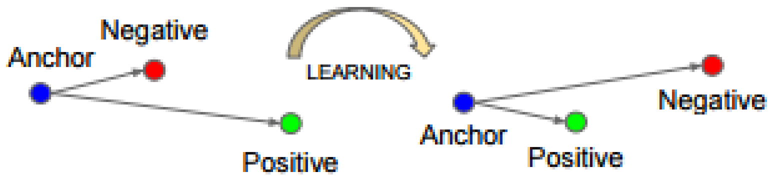


Figure 3.7: Minimizes distance between anchor and positive (different image of same person) while maximizing distance between anchor and negative.

Their implementation outperformed previous results on the LFW dataset and YouTube Faces DB by 30%, achieving a 99.63% accuracy on the LFW dataset. Since 2015 other publications have managed to push this boundary to 99.80%.[4] The LFW (Labeled Faces in the Wild) data set contains more than 13,000 images of faces collected from the web. Each face has been labeled with the name of the person pictured.

The Facenet only outputs a 128-length embedding. Actually classifying the face-embeddings is up to the user of the network. When using Facenet in this thesis a simply euclidean distance between the two 128-length embeddings. As our results in chapter 6 shows this works extremely well for its simplicity. The calculated euclidean distance is then thresholded against a value (0.6 in our experiments) to either output "yes this is a match", or "no, these faces do not match".

OpenCV 2.4 contains implementations of Eigenfaces, Fisherfaces and LBP for facial recognition. Another library OpenFace has trained the same deep neural network described in Schroff et al. (2015) and is publicly available.

3.5 Summary

Normally all of the methods outlined above are evaluated on large dataset of *images*. The application context in this project is applying these methods to video. The goal is to build a system that can fast and reliably log-in known users to the system and reject unknown users. Neither is it required to be able to track the movement of users in real-time. The PiCamera is able to sample video at more than 30 frames per second. In a normal evaluation setting this would correspond to evaluating the algorithm on 30 images and count the number of misses. However the goal of the system is simply to log-in the user correctly and not log-in the wrong user. The LBP generally has a 2-3%[5] lower test accuracy but can in return be several times faster than using haar-features. Even if the combined system of background subtraction $\rightarrow$ face detection $\rightarrow$ face recognition makes some individual errors, the hope is that the temporal context can be used to combine recent prediction. In other words the system should consider the outcome of the **N** past predictions before doing a log-in action.

The background theory discussed in this section will be experimented with to build a smart mirror with integrated face recognition to tailor the user interface on a user-to-user basis. The goal is to explore if this is possible on the Raspberry Pi 3 Model B, or if the intelligent mirror ought to be built with either more powerful hardware or to distribute its computations to a remote machine for processing.

[4]Overview of all submissions to the LFW dataset can be found at `http://vis-www.cs.umass.edu/lfw/results.html`

[5]I was unable to find any research document validating this number. However various discussions I have read on the topic mentioned 2-3% lower accuracy. This will be discussed in detail in chapter 6

4 Related Work

This chapter will discuss the related work that is relevant to this project. There are few research papers specifically on *Intelligent Mirrors*. The first section will focus on the hits found in a scientific literature search, using keywords such as: *magic mirror, intelligent mirror* and *smart mirror*. The next section will focus on the smart or intelligent mirrors found in a commercial setting. As we will see, many manufacturers are interested in the concept, even if there are ways to go before smart mirrors are on the same level of interest as other smart home devices. The final section will detail the findings of smart mirrors in the do-it-yourself(DIY) realm. The interest in smart mirror originally sparked as a cool DIY project for hobbyists.

4.1 Intelligent Mirrors in literature

FitMirror

Daniel Besserer et al. (2016) created a smart mirror for adding interactive fitness exercises to a person's morning routine. Their project utilizes the Microsoft Kinect v2 for tracking gestures and a Wii Balance Board for presence detection.

Interactive Mirror for Smart Home

Chidambaram Sethukkarasi et al. (2016) created an intelligent mirror that identifies users based on facial recognition, recognizes emotions, records health parameters and gives clothing advise. Their paper does not go in-depth on any of its subjects, but rather try to unite the ideas under the concept of an intelligent mirror. They use Viola & Jones for face detection and eigenfaces features for recognition. No results on the performance speed of neither detection or recognition was reported, but as shown in this thesis the speed of detecting faces with either haar-cascades, lbp-cascades or HOG classifiers can be greatly reduced by doing background substitution, as smaller image regions are faster to search. The experiments done in chapter 6 does however suggest that eigenfaces recognition method does not compare to newer neural network methods. Considering this paper is from 2016 it is a bit strange they opted for using eigenfaces.

Wize Mirror

The most intelligent of the smart mirror projects is likely to be the research project Semeoticons. Its purpose is to analyze facial signs to measure the health level of the person in front of it. The paper by Franco Chiarugi et al. (2013) discusses the motivation and rationale behind the project. Their idea was to extract quantitative features of facial expressions related to stress, anxiety and fatigue and use those features to quantify an individual's well-being. The features would be extracted from data collected from multisensory devices. The data would be collected in the form of videos, images, 3D face scans and breath samples. The project is first and foremost a research project to digitalize semeiotics - the physical signs produced by diseases - from facial images. As is evident from the title of their paper, the aim is to make the system non-intrusive to the user, but still being able to assess and monitor the individuals well-being. Because a mirror is a device most people use daily, either as part of their morning routine or bed routine, Franco Chiarugi et al. (2013) idea is to hide all sensory input devices in a mirror and provide the user with relevant feedback shown in the mirror - dubbed the Wize Mirror.

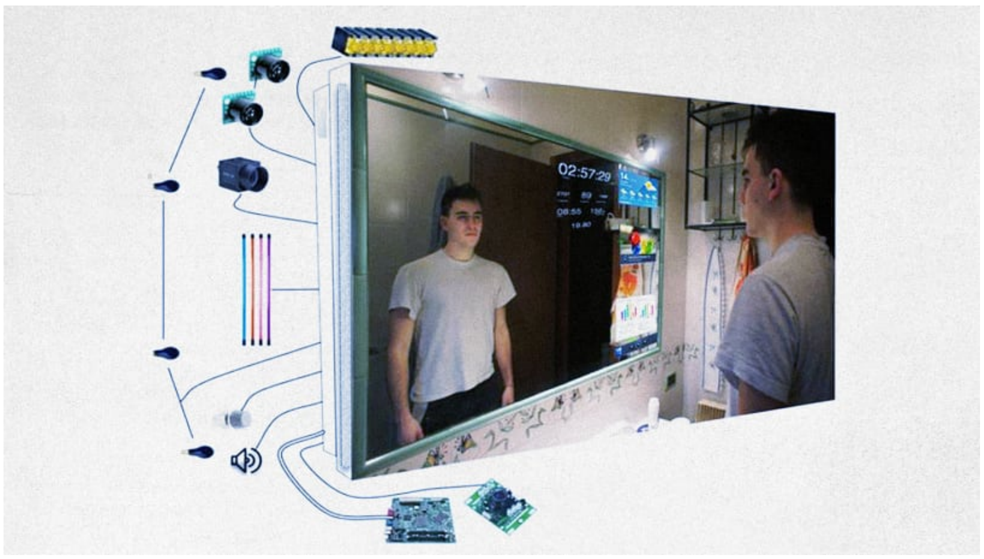


Figure 4.1: Semeoticons intelligent mirror concept utilizes numerous hidden sensors to gather information from the user without requiring any explicit input from them

Franco Chiarugi et al. (2013) also aims to customize the feedback given to each individual based on an acquired user profile for each individual. The feedback could either be displayed as simply as a comic or avatar - or more scientifically yielding more information. Figure 4.1 shows Semeoticons intelligent mirror concept.

The project began in 2013 and ended 2016. In Yasmina Andreu et al. (2016) the techniques, experiments and results were explained. The in-depth details are unimportant to this thesis. They concluded that the research project was a success and that creating a device for health self-monitoring and self-assessment is a reality. Prototypes of the Wize Mirror were evaluated in the fall of 2016 and the researchers aim to bring the product to the market in the future.

4.2 Commercial Intelligent Mirrors

This section will give an overview of the different types of physical intelligent mirrors that has been built. The term intelligent mirror and physical constructions of such mirrors were first mentioned back in the middle of the 2000s. The applications started as simple heads-up-displays (HUDS) and has in the later years evolved to include multimodal applications both for home entertainment and as health monitoring devices.

Phillips Mirror TV and MyHeart

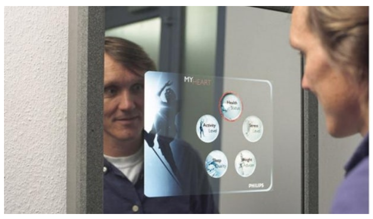


Figure 4.2: Phillip's MyHeart concept.

In 2003 Phillips unveiled their Mirror TV that was built using the same principle as that of smart mirrors. Their product was a normal TV that was put behind a two way mirror so that the TV would appear as a mirror when turned off and as a TV when turned on. They also had a option to have the mirror be larger than the TV. A usage example presented by Phillips was to have the children watch cartoons while brushing their teeth at the same time. Later in 2005 Phillips announced their research project MyHeart that built upon the idea of an informative mirror. While their original Mirror TV was simply a TV that also functioned as a mirror, the MyHeart project would integrate a display to showcase various medical statistics. However this project required on-body electronics to collect and analyze the data. The mirror itself simply served as an informative display.

Cybertecture Smart Mirror

James Law Cybertecture developed a commercially sold smart mirror in 2011. This mirror is more in line with the smart mirror we've come to know today. The product consists of a 32" LCD-display covered by a 37" two way mirror. The display can show weather forecasts, stream internet TV, the current time and various widgets. The smart mirror has numerous input methods such as remote controller, smartphone app and on-screen virtual keyboard. However Cybertectures' mirror does not enhance the mirror with any sort of intelligence. The mirror simply shows static information as setup by the user. While being a commercially available product, a price tag ranging from US$3600 to US$7700 makes it a smart home appliance reserved for the smart home niche.

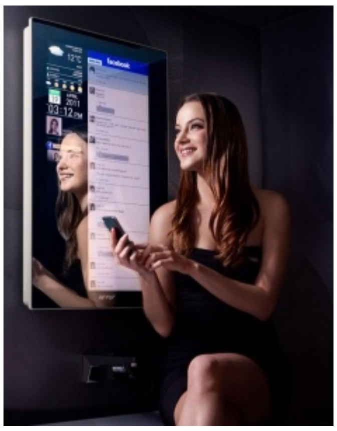


Figure 4.3: Cybertecture's Smart Mirror

Toshiba Smart Mirror

At the 2014 International Consumer Electronics Show
(CES)Toshiba showcased their smart mirror concept.[1] Their smart mirror concept was similar to previous smart mirrors but utilized gesture control as an input method. Toshiba showcased their smart mirror in different home environments. Their idea was that the smart mirror would be customized for the purpose it would serve in each room. The bathroom smart mirror would show information such as weather forecast and a personal fitness monitor. Meanwhile the kitchen mirror would assist in cooking meals - in the form of finding, adjusting and preparing recipes. Since the convention in 2014 there have been few details on the future of the project and whether it will be commercialized.

Microsoft Smart Mirror

In 2016 Microsoft released details on the smart mirror they have been working on. Their intention does not seem to be to create a commercial smart mirror to sell to consumers, but rather they unveiled all the details on how to build one and made all the code publicly available at a github repository. Rather than selling a finished product consumers have the option to assemble their own mirror as a do-it-yourself project, without needing to do any explicit coding. However a certain knowledge of computers and technology is obviously needed to assemble the mirror. The interface can show such things as the weather, time and date, stocks, and traffic. It also supports facial recognition and will load customized interface profiles custom to each user.

The mirror is run by a Raspberry Pi 3 with Windows 10 IoT Core that runs a hosted web app. This way the software is able to perform heavy tasks that would otherwise be unsuitable with the Raspberry Pi 3's hardware. Microsoft's solution shines some light on the situations on the state of smart mirrors as of 2017 - the major technology companies are yet to create smart mirrors where the trade-off between its appeal and its price is worthwhile

HiMirror

Again at the 2017 CES convention (that I personally attended) there were multiple smart mirrors on display. A company called New Kinpo Group launched their take on the smart mirror called HiMirror. This smart mirror has a camera to specifically monitor your skin health. The mirror will scan your skin and give you metric to tell you what to improve. The mirror uses facial recognition to log a users skin firmness, texture, clarity, brightness and health on a day to day basis. Unlike many of the previous smart mirrors this product hit the market in october 2016 at a price of \$189. While considerably cheaper than previous smart mirror entries, it's features are very specific and does not offer much beside skin health monitoring.

[1]CES is a yearly technology convention in Las Vegas where more than 200 000 attendees gathers. There's an entire exhibition hall devoted to smart home gadgets and has in the last few years had multiple companies showcasing their take on smart mirrors (as of writing this in 2017).

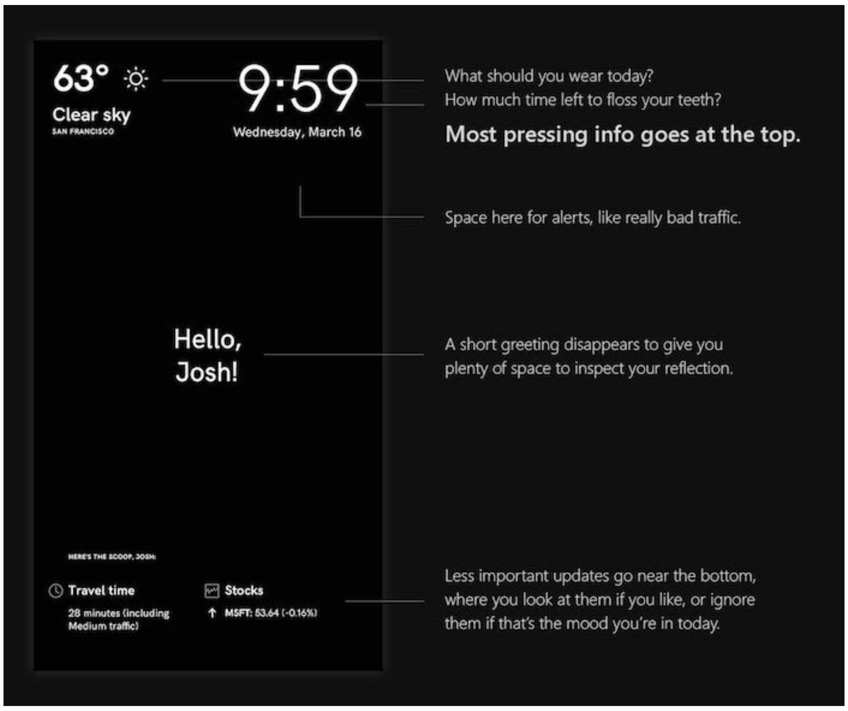

63°
Clear sky
SAN FRANCISCO
9:59
Wednesday, March 16
What should you wear today?
How much time left to floss your teeth?
Most pressing info goes at the top.
Space here for alerts, like really bad traffic.
Hello, Josh!
A short greeting disappears to give you plenty of space to inspect your reflection.
HERE'S THE SCOOP, JOSH:
Travel time
28 minutes (including Medium traffic)
Stocks
MSFT: 53.64 (-0.16%)
Less important updates go near the bottom, where you look at them if you like, or ignore them if that's the mood you're in today.

Figure 4.4: Microsoft smart mirror HUD

Griffin Technologies Smart Mirror

Griffin Technologies unveiled their take at the smart mirror at the 2017 CES convention. They call their product the Connected Mirror and it will serve as the smart home hub for several smart home appliances made by Griffin Technologies. The mirror can display local time and weather, notifications from your phone and statuses from other Griffin smart home tech connected to the mirror. The mirror does not employ any user recognition, but the interface can be customized through a smart phone app that is also used to control any other Griffin smart home devices. Slated for a late 2017 release the smart mirror has a price tag of $1000.

Two Way Mirrors LLC

However, a company called *Two Way Mirrors LLC*[2] who, naturally, specializes in two way mirrors have introduced a smart mirror product line to their categories. Here they offer see-through mirrors that comes pre-configured with a raspberry pi - mimicing that of most do-it-yourself projects outlined previously. It is not a out-of-the-box smart mirror for the average customer as a certain amount of computer knowledge is required for configuring the software. But it proves that manufacturers see a potential market to exploit.

[2]`http://www.twowaymirrors.com/`

Summary

As should be clear from these related works are that the idea of commercially available intelligent mirrors still has a way to go. The price ranges from anywhere between $200 and $7700 and are unacceptably high for what is acceptable for the average household to spend on a gadget that, while cool and futuristic, offers nothing the PC or smartphone does not already offer for a much cheaper price - and that are way more convenient.

As exemplified in this section, numerous companies have produced smart mirror prototypes to showcase at the CES, but few has been actually put in production. Those that have been put in production does come with a pricetag only the smart home niche would be willing to pay.

Most of the mirrors are smart mirrors - not intelligent mirrors. They display information on a screen without any form of interaction or additional sensors to capture data of its user. Toshiba's prototype used gestures as a mean of interaction, HiMirror and Microsoft's platform used a camera for facial recognition to provide a user customized experience. Phillips pioneered the idea of creating an intelligent mirror for health monitoring in the middle of the 2000s. Though in the later years the intelligent mirror in the works by Yasmina Andreu et al. (2016) offer health monitoring capabilities that, unlike the MyHeart, does not require any wearable sensors but collects all information using hidden sensors in the actual mirror.

4.3 Do-it-yourself Smart Mirrors

The concept of smart mirrors truly became popular as a do it yourself(DIY) type of project and still is mainly such a project. That Microsoft as late as 2016 decided to publicly release the details on their smart mirror project for people to build their own rather than commercialize the project are a good indicator that smart mirrors will continue to be a DIY project for some time. This section will discuss some of the most prominent work done in the DIY field that has inspired the popularity of such mirrors.

The most known project on smart mirrors is done by Michael Teeuw. His project was among the first to embrace the idea of using a RPI to power an informative mirror. His project is also featured on the official raspberry pi homepage [3]. He has created tutorials on how to physically build the mirror, as well as providing a community-driven open source framework for creating a module-based interface called *MagicMirror*^{2}. This framework makes it easy to customize the interface with interchangeable modules that can be created by anyone. Examples of existing modules are: weather, news, time or daily comic strip. I will be extending this framework by writing my own modules that will be responsible for the intelligence in the mirror. Therefore this framework will be examined in much greater detail in chapter 5.

Last year (2016) Ryan Nelwan gathered much interest on the internet (his youtube video showing off his project currently has more than a million views[4]). His project is a

[3]https://www.raspberrypi.org/blog/magic-mirror/

[4]https://www.youtube.com/watch?v=sh2EJzplkpM&feature=youtu.be

smart mirror, same as Teeuw's, however it has touch capability. It serves mostly as an entertainment system where a user can use the touch controls to control music or run different program, but does not implement any forms of artifical intelligence. He has not provided any details about how his project is implemented or what technologies he is using.

From my literature review it is apparent that the smart mirror concept is becoming increasingly popular among commercial manufactures and hobbyists alike. However the main focus is on creating a smart home appliance - not an intelligent mirror. Smart home applications are just growing more and more popular and it is inevitable that smart mirrors will be commercialized and integrated in the smart home environment.

5 Architecture

This chapter will detail the software architecture of the intelligent mirror. First the architecture for doing user recognition based on faces will be presented. Next an alternative architecture where heavy computational operations will be off-loaded to an external machine will be introduced. This architecture will be used if the RPI is unable to process all AI algorithms fast enough on its own. Additionally, if the system is to be extended to include other forms of intelligence in the future, it is unreasonable to believe the RPI will be able to scale with the increasingly heavy computations. Finally the smart mirror framework $MagicMirror^2$ will be thoroughly explained. Rather than creating everything from scratch a module to this framework will be created so the mirror can utilize the constantly growing number of applications created for the framework.

5.1 Hardware

This project also involves actually building an intelligent mirror. To be precise, the intelligent part is supplied by the software. The actual physical device built would more correctly be coined a *smart mirror* according to the distinction between smart and intelligent mirrors made during the introduction. There are many properties to be considered when building the mirror: weight, mirror size, screen size, hardware selection, power supply, mounting technique, heat generation, and aesthetics. A mirror is also a decorative piece so creating something that is both functional and aesthetically pleasing is considered important. This section will explain many of the choices to be made for building a smart mirror, the options available, and why the final choices were made.

Two-way Mirror

A two-way mirror[1] is a mirror that is partially reflective and partially transparent. This property allows the mirror to be reflective from one side (if the room is bright) and transparent from the other side (if the room is dark). Typically these types of mirrors are found in interrogation chambers. This property is crucial for the project to work. It is exactly its ability to be transparent from one side that allows the monitor to shine through to the other side while still behaving as a mirror.

Two way mirrors are made by coating the glass with a thin and almost-transparent layer of metal (usually aluminum). The effect is that some light is reflected while the rest will penetrate the glass. The mirror is bidirectional - initially it does not matter

[1]Funnily enough these types of mirrors are also called one-way mirrors, but I will refer to these types of mirrors as two-way mirrors throughout this report.

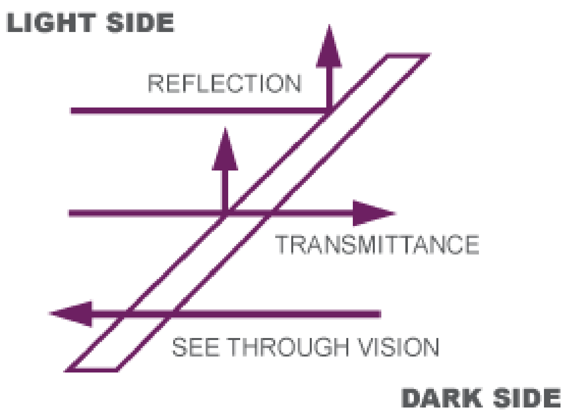


Figure 5.1: Illustration of how two-way mirrors work.

which side you are looking from. The light passes equally in both directions. However to achieve the desired effect one side must be kept dark while the other is bright. The backside of the mirror will be coated in black to simulate the darker room. The monitor creates light on its own so its screen contents will be able to shine through from the dark side. By using a black and white interface only the white color will travel through the mirror. However a problem with LCD monitors is that it uses background lighting to light up the diodes, displaying colors. However a side-effect is that the black pixels will not be truly black because of the back-lighting. Ideally the black pixels would be totally "off" - emitting no light at all. Newer types of monitors that employs technology such as OLED monitors does not use back-lighting. Unused pixels will completely dim themselves. The downside is however that such monitors are relatively new to the market and rather expensive. For this project an older LCD laptop monitor was used. However, the mirror was designed to be highly modular - allowing replacing most of the hardware. Appendix A gives a detailed description of the design and step by step instructions for building the mirror.

There are a few options when it comes to selecting the correct type of mirror. The easiest is to simply buy an interrogation mirror that has the properties described above. These are unfortunately also quite expensive. They exists in both acrylic and glass. Acrylic is potentially cheaper and easier to work with than glass. Alternatively there also exists two-way-film that can be used on ordinary windows to achieve the same effect. Unfortunately the selection in Norway is very limited and the only option I could find was a two-way mirror made of glass.

Embedded Devices

In order for the mirror to do anything useful at all (beside being a standard mirror) a computational device is required to operate the camera, provide the GUI and potentially analyze the captured video for known users. The embedded device chosen is the Raspberry Pi 3 Model B. RPI is probably, alongside Arduino, the most famous embedded device for its cheap price, active community and numerous accessories that makes the RPI suitable for all kinds of fun DIY projects from making a homemade Gameboy to making a security monitoring system. One of the research goals of this thesis was to examine if the RPI is a suitable embedded device for the intelligent mirror, or if other potential embedded devices should be researched. That is left to future work based off the results obtained in chapter 6.

Screen

There are three main possibilities in the choice of screen selection: laptop replacement screen, desktop monitor or a usb-powered monitor. Each have their pros and cons listed in table 5.1.

Screen-type	Pros	Cons
Replacement screen	thin light cheap	requires additional controller board for video signal and power often less than 16"
desktop monitor	24" and up integrated power supply integrated video signal	heavy expensive
usb-powered	thin light no need for external power video and power in one cable	expensive compatibility issues with linux systems

Table 5.1: Overview of the pros and cons of the possible screen options

Early on in the project a usb-powered monitor[2] was acquired as it had the most appealing properties for building the smart mirror. However, most usb-powered monitors employ the displaylink technology which has very limited support on Linux platforms. It also requires that the source is able to provide enough power through usb to power the monitor. Both of these issues proved to be fatal for my success in using this monitor with the RPI. It appeared the RPI was not able to provide enough power from its usb ports, and neither was its operating system[3] compatible with the displaylink technology. After a week of trial and error, I realized it was futile to get this monitor to work and gave it up.

[2]Asus MB169B+
[3]Raspbian Jessie

Camera

A camera is the system's primary input device. The computer vision techniques used to recognize a user needs a video feed to do operations on. There are two ways to record video on a RPI. Either with a USB-camera or a PiCamera attached to the CSI port. Both have their pros and cons as shown in table 5.2.

Camera	Pros	Cons
USB-camera	quality may exceed PiCamera may have integrated microphone	often more expensive slower
PiCamera	fast	poor in darkness
NoIR PiCamera	fast needs less light to see additional IR-lights enables night-vision	odd daylight colors

Table 5.2: Overview of the pros and cons of camera options

Ideally the camera would be hidden behind the mirror. This way the user won't feel like they are being recorded and the mirror will appear to just be a normal mirror until it is activated by the camera. However, this makes it more difficult for the camera to take images. Two way mirrors are designed to reflect a certain percentage of the light. That means only a fraction of the light will reach the camera behind the mirror. The reflection/absorbed ratio of the procured mirror was not given by the manufacturer, but based on other two way mirrors I have seen it seems to reflect relatively much of the light. As a result the regular PiCamera was not able to take image of sufficient quality to enable face recognition. This lead to the acquisition of a NoIR camera. That camera provides sufficient image quality for most of the day, but still suffers at night unless the room is exceptionally well lit. As the camera has no IR filter, the daylight colors will be rather bleak and odd. But this does not matter to us as most computer vision tasks uses the greyscale image to do computations. The important details of the image remains the same. The current version of the mirror does not use any additional IR lighting. But future iterations of the projects could install this to make face recognition viable at any given time. A high standard USB-camera could potentially give high enough quality images throughout most of the day. But this has not been tested. However the use of USB-cameras captures images at a much lower FPS (frames per second).

The major advantage with using the native RPI cameras are that the CSI port runs directly to the GPU, using virtually zero CPU time. On the other hand USB-cameras are connected by USB. This requires the CPU to pull data from the USB bus. Achieving a decent FPS this way requires heavy work by the CPU. But we already requires the CPU for doing heavy computer vision tasks for user recognition. By offloading the reading of video frames to the GPU, we free the CPU to do our other intensive tasks.

5.2 Embedded Architecture

This section will discuss the architecture for running the system on an embedded device. All computations will be made locally on the RPI.

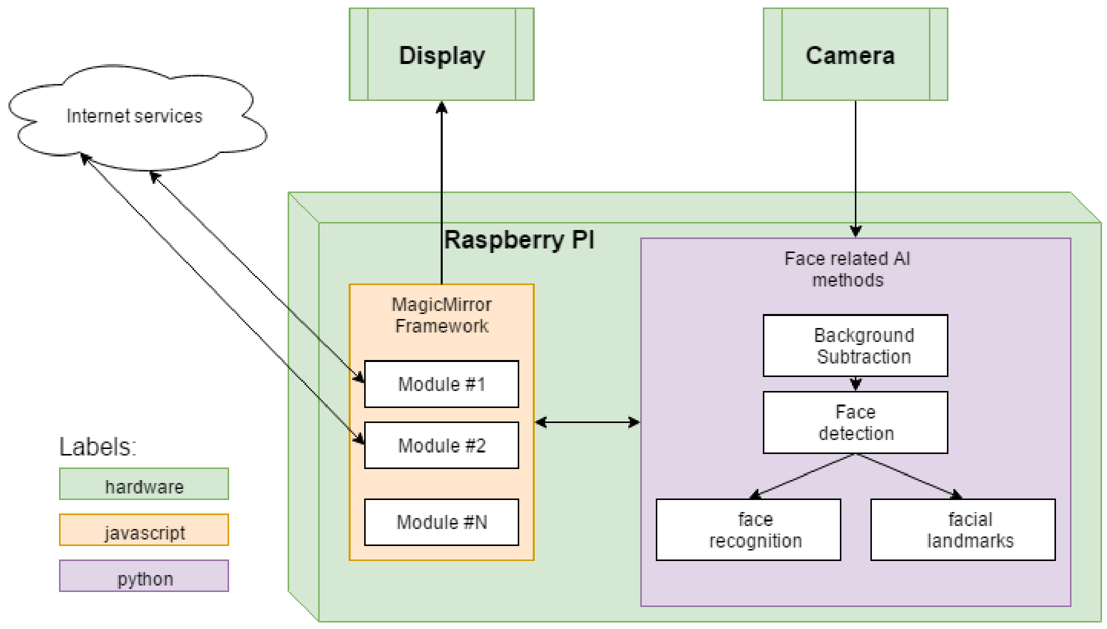


Figure 5.2: System view of the local architecture

As discussed in the background chapter, the pipeline for facial recognition consists of five primary steps: background subtraction → face detection → alignment → feature extraction → classification. The system will use a local architecture - all computations will be done on the raspberry pi. Therefore it is vital that methods used are not unnecessarily high in computational cost. The experiments are done in order to find what methods are most effective in facial recognition given limited computational power and limited amount of faces that needs to be discriminated. Further work on this project will involve more artificial intelligence incorporated to the the mirror. Most artificial intelligent methods are more or less complex. Hence, hopefully it is possible to do effective facial recognition with computational resources to spare. If the results of this experiment shows that a facial recognition system is too computationally complex to do well purely on the RPI, a remote architecture should be considered to free up some resources on the RPI. It is important to keep in mind that it is also responsible for running the software that hosts the user interface.

A modular facial recognition system will be implemented so that the different feature extractors and classification algorithms can be easily swapped. The algorithms will be evaluated based on their execution times and accuracy. Most facial recognition systems built and researched are concerned with discriminating a huge amount of people. But it is important to remember that in the context of this project, the amount of people is that of a household (or an extended family). In most cases likely to be less than ten people.

Figure 5.2 shows the local architecture. The $MagicMirror^2$ framework is responsible for the interface that will be displayed on the monitor. When the raspberry pi is booted the $MagicMirror^2$ framework starts up and orders the AI module to start analyzing frames. The AI framework then continuously polls image frames from the camera, analyzes them and sends the results asynchronously back to the $MagicMirror^2$ framework as results come in. This architecture clearly separates responsibilities. GUI tasks are left up to the $MagicMirror^2$ framework. Miscellaneous AI tasks are contained in its own module. Therefore the AI capabilities of the mirror can easily be extended to offer more functionality and also makes it easy to make a remote architecture as discussed in the next section. The downside of the local architecture is as mentioned the limited computational power of an embedded device. Neither will it scale well if the AI module is to be expanded in the future. On the other side the intelligent mirror will be all self-contained; there is no need for an external machine to run at all times.

5.3 Remote Architecture

If the RPI is unable to handle computer vision tasks as shown in the local architecture a solution is to employ a remote architecture. The remote architecture is shown in figure 5.3. Here the RPI is only responsible for rendering the GUI and streaming the video feed on the local network. An external machine will read the video stream, process the images and return the acquired results to the $MagicMirror^2$ framework on the RPI.

This requires the external machine to be on the same network as the intelligent mirror. Alternative solutions is to use hosted cloud services that allows you to use their clusters for doing computations. Though this poses a problem of cost and security. When the video feed is only streamed locally it will be safe and only accessible on your local network - provided the network itself is secure. If the images are to be processed on a remote commercial service, privacy cannot be guaranteed. Of course using major trusted services such as Amazon Web Services (AWS)[4] or Google Cloud Computing[5] are generally trusted.

In this architecture the RPI camera will serve as a IP camera. The term IP camera are generally used for surveillance cameras that are accessed at a specified IP address. The RPI will upload the images in the MJPEG format on the local network that the external machine can read from. This way the RPI need not be concerned with knowing the IP addresses of the external machine(s). Only the external machines need to know the RPI's IP address for returning the result. This way other external machines can easily be added to the system if needs be.

[4]`https://aws.amazon.com/hpc/`
[5]`https://cloud.google.com/`

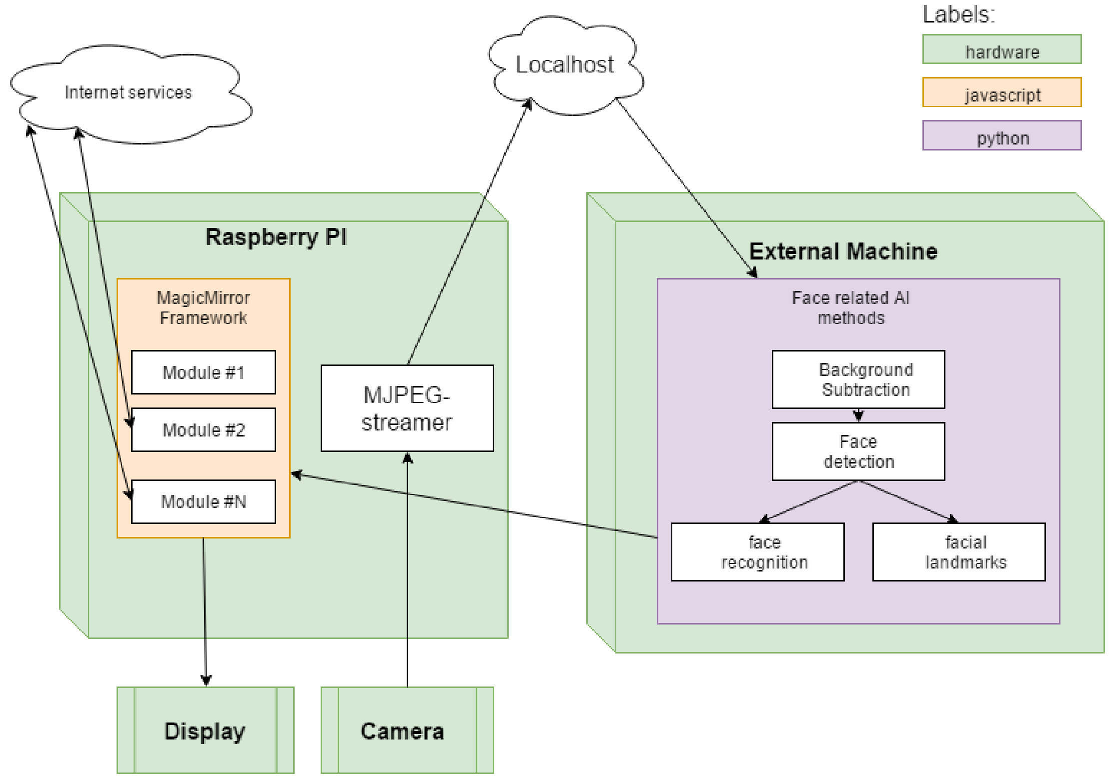


Figure 5.3: System view of the remote architecture. Motion JPEG (MJPEG) is the video format used for streaming the video.

5.4 User Recognition Software Architecture

This section will introduce and discuss the software module that has been built in order to recognize users. The idea was to build a modular framework where individual pipeline steps could be independently customized. An overview of the system can be seen in figure 5.4.

The **MainLoop** will continuously poll either the RPI camera (local architecture) or the locally hosted IP camera (remote architecture) for frames. Then the frame will be processed through the user recognition pipeline: background subtraction $\rightarrow$ face detection $\rightarrow$ face recognition[6].

The **BackgroundSubtractor** is responsible for keeping track of the background and moving objects. The first step in the main loop is to ask the **BackgroundSubtractor** for sub-regions in the given image that are likely to contain a person. Sub-regions below a certain size are filtered out. It is safe to filter out small moving areas as the sub-region that contain our person of interest is likely close and in focus of the mirror. Still the current area size limit is very forgiving as the computation time of searching

[6]Earlier this pipeline also included alignment, but this step was not implemented

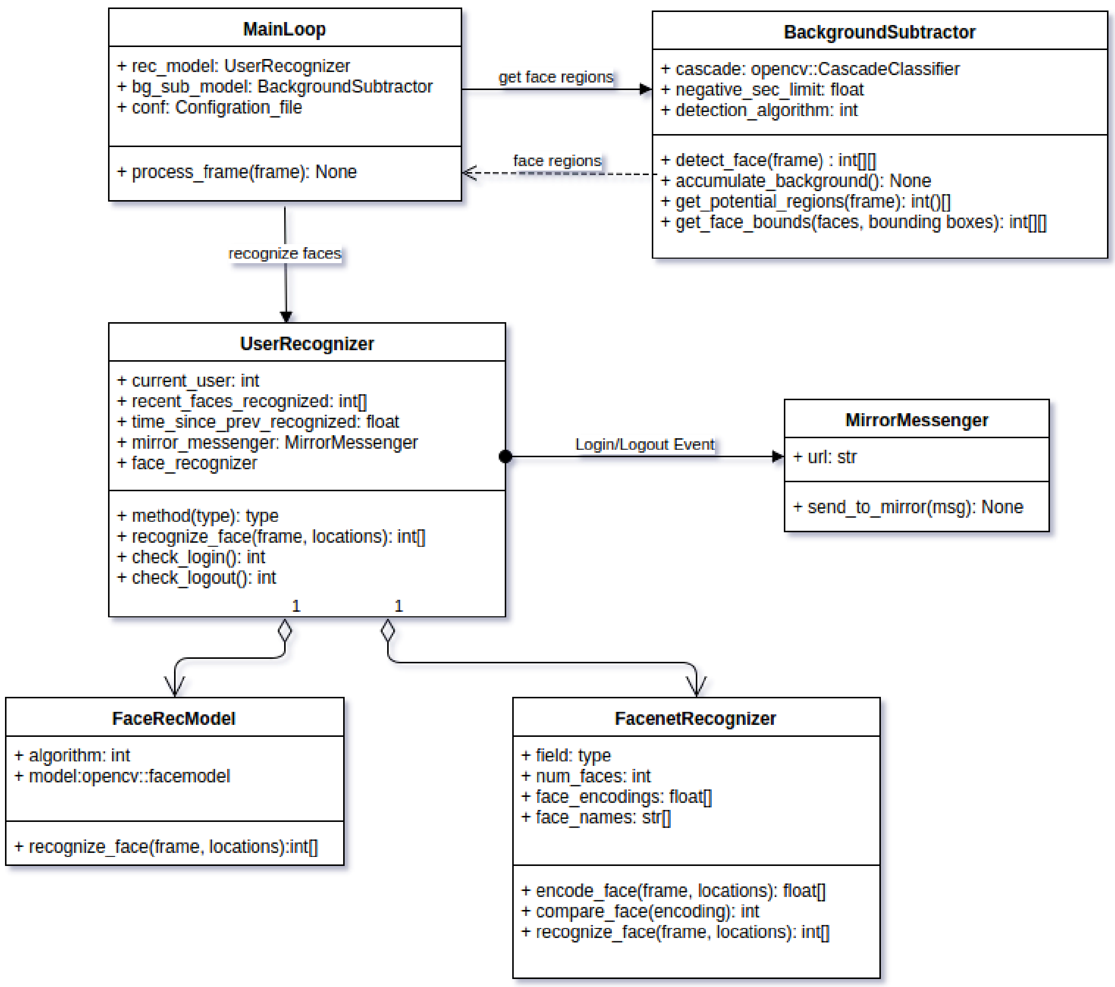

MainLoop
+ rec_model: UserRecognizer
+ bg_sub_model: BackgroundSubtractor
+ conf: Configration_file
+ process_frame(frame): None
get face regions
face regions
BackgroundSubtractor
+ cascade: opencv::CascadeClassifier
+ negative_sec_limit: float
+ detection_algorithm: int
+ detect_face(frame) : int[][]
+ accumulate_background(): None
+ get_potential_regions(frame): int()[]
+ get_face_bounds(faces, bounding boxes): int[][]
recognize faces
UserRecognizer
+ current_user: int
+ recent_faces_recognized: int[]
+ time_since_prev_recognized: float
+ mirror_messenger: MirrorMessenger
+ face_recognizer
+ method(type): type
+ recognize_face(frame, locations): int[]
+ check_login(): int
+ check_logout(): int
1
1
Login/Logout Event
MirrorMessenger
+ url: str
+ send_to_mirror(msg): None
FaceRecModel
+ algorithm: int
+ model:opencv::facemodel
+ recognize_face(frame, locations):int[]
FacenetRecognizer
+ field: type
+ num_faces: int
+ face_encodings: float[]
+ face_names: str[]
+ encode_face(frame, locations): float[]
+ compare_face(encoding): int
+ recognize_face(frame, locations): int[]

Figure 5.4: UML diagram of the most important classes, methods and variables of the user recognition system.

a small negative region is near negligible. In order to filter out moving sub-regions **BackgroundSubtractor** maintains an averaged background image of the N previous frames. This way newly placed stationary objects will not count as moving regions. To avoid incorporating a user into the background model (say someone uses 20 minutes to fix their hair) the background model will not accumulate unless a given amount of time has passed since a face was last detected. The image is blurred before being accumulated into the background model to smooth out any fine-grained details. The potential sub-regions are then processed by the chosen face detector and a list of locations are sent to the chosen face detector that ultimately returns a list of (top, right, bottom, left)-tuples of detected face regions.

Next the image and the list of face locations are sent to the **UserRecognizer**. The **UserRecognizer** then further delegates the face recognition task to the chosen face recognition method, **FacenetRecognizer** for using facenet, or **FaceRecModel** for using either *fisherfaces, eigenfaces or LBPH.* The **UserReocnizer** keeps track of the N latest recognized faces and uses this array to determine whether the newly recognized user should be logged into the system or not. The idea is that even if the system fails to find a face or recognizes the wrong person, it should still be able to log in the correct user, or keep that user from being prematurely logged out of the system. To achieve this a limit is put on the number of consecutive recognitions that must be equal. For each frame the recognized users of that frame is added to the list of recently recognized faces. Note that the system could detect more than two users in a single frame, in which case both will be recorded as users for that frame. If the consecutive user limit is reached - that is the same face is present in the last N frames - a login event will be sent to the **MirrorMessenger**, given that no user is already logged in. Should the N previous frame involve several persons in each and every frame, the user is chosen randomly. However this scenario is highly unlikely. The **MirrorMessenger** will then propagate the login request to the $MagicMirror^2$ which is then responsible for updating the GUI. Note that this setup works exactly the same whether a local or remote hardware architecture is applied.

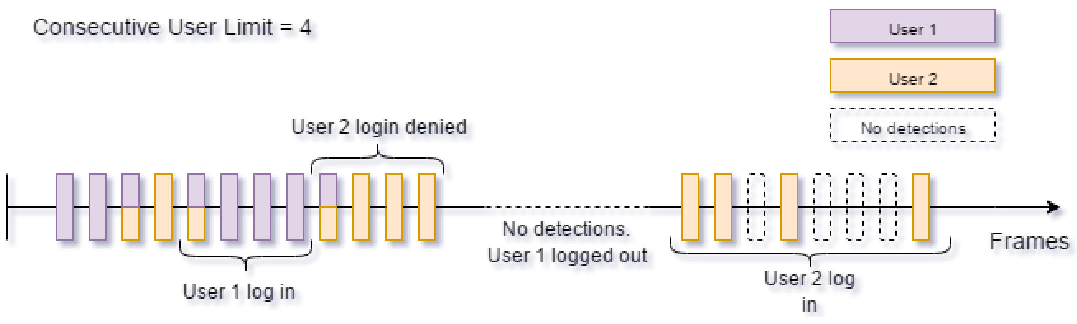


Figure 5.5: Illustration of various login scenarios

Further the **UserRecognizer** is responsible for protecting that user's login session. If another user known to the system passes by in the background we do not want the mirror to suddenly recognize that person as the logged in user. To do this the **UserRecognizer** has a logout timer. If a certain amount of seconds has passed since the system last recognized the logged in user in a frame that user will be logged out. The system is then back to its original state and other users are free to log in. Whenever the current user is recognized the logout timer is replenished to its max value.

Figure 5.5 demonstrates some of these scenarios. Each colored rectangle represents the detected user for that frame. If the rectangle is multicolored that means multiple users were recognized for that frame. Dashed rectangles means no faces were detected for that frame. In the figure the required number of consecutive recognitions is four. In that case the system keeps track of the four latest recognitions, and as seen in figure 5.5 the purple user is able to login first as it is the first to reach four consecutive recognitions. Though immediately after the orange user fulfill the requirements to login. However since the purple user logged in first his login session is protected and the orange login is denied. Then follows a period of no detections. The timeout limit is reached and the purple user is logged out. The system then eventually detects four orange recognitions and he is logged in. The final login in figure 5.5 demonstrates that frames where no recognition occurs is not added to the recognition queue. It could be that the user is brushing his teeth so that some of the frames are occluded. But as long as the frames where a face is detected leads to the same recognized face.

This system raises one question though: what if the first four frames incorrectly recognizes a person, while all the following frames consistently recognizes another person? As it is now the first - incorrect - user would be logged in and block all other users for the duration of the logout timer. However as we will see in the next chapter this did not prove a problem for the Facenet recognition algorithm on the test data. Though the results suggest it may pose a problem for other face recognition methods. But still, the method should be enhanced so that the first incorrect login request could be overridden without waiting for the timeout - if recent recognition strongly suggest that the initial login decision was wrong. Due to time constraints this is considered future work.

5.5 The Magic Mirror Framework

$MagicMirror^2$ is a modular framework initiated by Michael Teeuw that is deployed by most hobbyists when creating their own magical mirror. The framework provides easy integration of third-party modules and has a active community that helps expanding the functionality available to magic mirrors. My contributions will be a module developed in this framework so that it can be shared with other magic mirror users. This section will explain the framework in detail.

The $MagicMirror^2$ framework is written in NodeJS. NodeJS is an asynchronous event driven JavaScript runtime. Primarily it is used for developing network applications, but the magic mirror framework is not to be used as a hosted server available to others, but rather simply as a localhost. The $MagicMirror^2$ uses Electron as an application wrapper. In simple terms, this allows the software to be run as a standard desktop application.

The framework is module based. A user will customize a javascript configuration file that tells what modules to include and their relative placement on the screen. The core system will read this file and initialize the required modules and handle all the work for positioning the modules correctly.

Figure 5.6 shows a simplified view of the $MagicMirror^2$ system. Every module consists of a frontend part and a backend part. The **NodeHelper** is able to communicate between model and forward updates to its frontend. In the figure only the **UserRecognition** module is expanded to showw the frontend and backend. When the **MirrorMessenger** issues a login request to the framework the NodeHelper listens on a specified port for message. Whenever a message is received it lets its parent module know so it can update its GUI.

When a login request has been processed the module will show or hide other modules

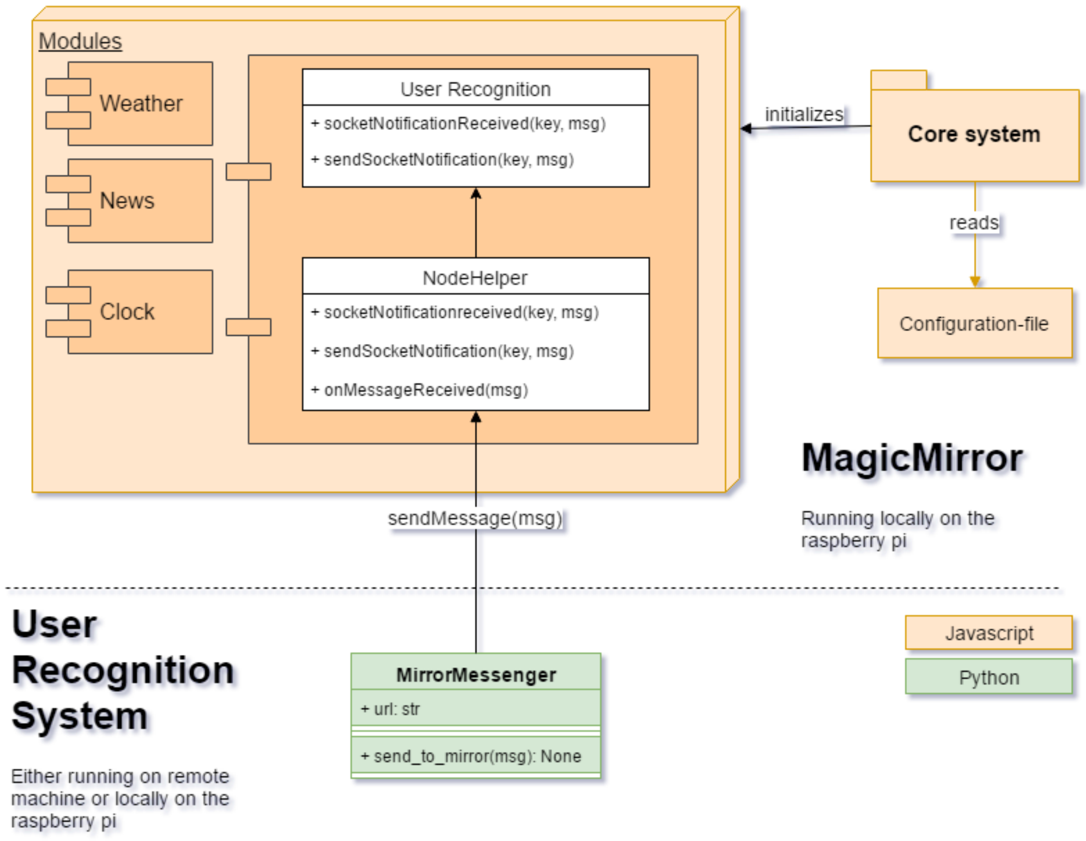

Modules
Weather
News
Clock
User Recognition
+ socketNotificationReceived(key, msg)
+ sendSocketNotification(key, msg)
NodeHelper
+ socketNotificationreceived(key, msg)
+ sendSocketNotification(key, msg)
+ onMessageReceived(msg)
initializes
Core system
reads
Configuration-file
MagicMirror
Running locally on the raspberry pi
sendMessage(msg)
User Recognition System
Either running on remote machine or locally on the raspberry pi
MirrorMessenger
+ url: str
+ send_to_mirror(msg): None
Javascript
Python

Figure 5.6: Simplified UML diagram of the $MagicMirror^2$ framework

based off the specifications in the $MagicMirror^2$'s configuration file. This is where all the metadata and module initialization variables are stored. Every module has an array that lists the users for which that module should be shown. Additionally modules can be set to display for unknown users or to be displayed regardless of logged in user. It is important that the list of users are identical across both systems, so that the sent index matches the same user when sent from the remote **UserRecognition System** to the $MagicMirror^2$ module. The user indicies are 1-indexed rather than 0-indexed. This is due to the implementation of the eigenfaces, fisherfaces and LBPH face recognition methods which are 1-indexed. As such the index 0 corresponds to an unknown user.

6 Experiments and Results

This chapter will explain and discuss the experiments done with regards to user recognition. The goal is to examine the execution speed and accuracy of the user recognition system implemented in order to determine which of the architectures, local or remote, are best suited for the user recognition system of the intelligent mirror. First the test data will be presented. Next we introduce the parameters of the experiments, and conclude by showing the result of the experiment

6.1 Experimental Plan

The goal is to find the best user recognition pipeline with regards to the accuracy and execution speed trade-off. From this we will conclude whether a local or remote architecture is best suited for the intelligent mirror - in other words, whether the Raspberry Pi 3 Model B is suitable for this project. As mentioned we are interested in testing both the execution speed and accuracy of the system. The execution speed will determine the hardware requirements. The accuracy will quantify the usefulness of the user recognition system. The system should log in the correct user as quickly as possible, and not log out that user before the current user intends to log out. Oppositely, the system should not log in the incorrect user, logout a user while he/she is still using the system, or allow passerby's to "hijack" the login session. The system does only support one active user at a time.

6.2 Test Data

In order to be able to quantify the performance of the user recognition system a dataset was collected. The use case is people mirroring themselves. Most datasets used to measure the performance of facial recognition systems are images, such as the LFW. Or on videos on datasets such as YouTube Faces DB. However these datasets are not an appropriate performance measure for the user recognition used for this intelligent mirror. It is not vital that the system is able recognize the correct user from every frame of the video feed. Nor is it vital to recognize users from odd angels or massive obstructions. The system does however need to login the correct user at the correct time, endure obstructions caused by everyday mirroring tasks such as brushing teeth, fixing the hair, and getting dressed before going out.

The dataset consists of 35 videos, featuring 10 unique subjects, 7 men and 3 women. The data is summarized in table 6.1. The age of the participants ranges from 20-50

Number of videos	Unique subjects	Average age	Gender Distribution
35	10	27 years	3f 7m

Table 6.1: Collected test data summary

years. [1] The dataset created consists of video of people mirroring themselves while performing various common mirroring tasks. The tasks performed include: brushing teeth, subject fixing their hair, cleaning their face, getting dressed, putting on a hat, or just checking out their looks. Some video clips also includes other subjects moving around in the background. Either just passing by or stopping to look in the mirror.

Each video contains a single login timestamp and a single logout timestamp and features only one subject (though others may pass by in the background). The dataset comes with an annotated JSON-file that includes the login-frame number, logout-frame number, and user ID for each video with its name as the dictionary key. An excerpt of the annotated JSON-file format is shown in table 6.2. The timestamps represent the number of the frame in that video file where the event starts, rather than time elapsed. [2]

key	login-frame	logout-frame	user-ID
001.webm	160	350	4
002.webm	100	470	8
003.mp4	200	1150	2

Table 6.2: Excerpt of test video annotations

Each test video is then analyzed by the user recognition system. A list of all the login timestamps and the user logged in, and a list of all logout timestamps and logged out user is returned. This data is then matched against the corresponding data in the annotated JSON file. The measurements shown in table 6.3 is subsequently calculated.

The statistics from all the individual test videos are then combined to produce the measurements used to evaluated the system. In addition to summing the statistics from table 6.3, the mean, median, variance and standard deviation are calculated for the login-delay. This is an important statistic that determines the *smoothness* of the system. A user does not want to wait five seconds before the mirror logs him in.

It is though worth noting that the test data does not truly replicate the conditions of the conditions the actual intelligent mirror will be facing. As mentioned the test data is

[1]Originally the test set includes videos of people older than 80. However none of the tested face detection algorithm could detect a single face in the test examples involving the elderly. Those test examples were ultimately removed as they did not provide any meaningful results to this experiment. Of course the user recognition system should be usable by people of all ages, but face detection for the very young and the very old is another research topic of its own, and not discussed any further in this thesis.

[2]The system runs at as many frames per second as possible. Measuring events in elapsed seconds is thus inaccurate.

Statistic	Description
Accuracy statistics	
Key	The filename serves as the identification key
Login-delay	The number of frames passed since the ideal login frame to the actual login frame
Incorrect logins	How many times did the system issue an incorrect login event
Premature logouts	How many times did the system issue an unwarranted logout event
Login-takeover	How many times was a logged in users session hijacked by another user
No logins	How many times did the system fail to log in a user at all
Execution speed statistics	
Average Face Detection Time	How much time was spent detecting the face for each frame
Average Face Recognition Time	How much time was spent recognizing the face for each frame

Table 6.3: Explanation of statistics calculated for each video

collected at various locations, with various cameras. These cameras are not obstructed by the mirror the same way the RPI camera is, hence the test data videos are far more clear and bright than the video feed collected by the mirror's camera. Though it is more difficult to gather testing data with the mirror as it is not exactly easy to bring around with you. Still this aspect only applies to the accuracy aspect of the tests. The execution speeds will naturally remain the same. Hopefully the results obtained from the test data translates to the conditions of the mirror's camera.

6.3 Experimental Setup

The experiments are done in order to examine if any combination of face detection and face recognition algorithms yield viable results to enable the system to run satisfactory on the RPI, both in terms of accuracy and execution speed. The methods have been introduced in chapter 3 and the methods that have been tested are summarized in table 6.4.

It is expected that the Facenet recognizer will perform the best. It is a modern deep neural network algorithm that outperforms LBPH, Eigen-faces and fisher-faces on the LFW dataset. However, even though the neural network is pre-trained it is still a considerable heavy computational task to propagate an image through the network. Therefore it remains meaningful to compare the accuracy vs speed trade-off of the older face recognition algorithms.

Detection methods	Recognition methods
HOG face classifier (dlib)	LBPH face recognizer (opencv)
Haar cascade (opencv)	Eigen faces recognizer (opencv)
LBP cascade (opencv)	Fisher faces recognizer (opencv)
	Facenet recognizer (dlib)

Table 6.4: Detection and recognition algorithms experimented with. Implementation found in library written in parenthesis

Next the benefit of applying background subtraction will be examined for each of the face detection methods. Every method should see some sort of speed increase as it follows that a smaller image region to search should be faster than a significantly larger region. Naturally this must be compared to the time it takes to extract these smaller regions.

6.4 Experimental Results

This section will detail the findings of the experiments done in order to be able to draw a conclusion as to what user recognition pipeline and architecture should be applied. First the common parameters for all the experiments will be introduced, followed by a discussion of the results both in terms of accuracy and execution speed.

6.4.1 Parameters

Table 6.5 lists the common parameters that remain untweaked in all of the experiments.

Parameter	value	explanation
Eigen threshold	3000	predictions above this threshold are considered unknown
Fisher threshold	600	predictions above this threshold are considered unknown
LBPH threshold	110	predictions above this threshold are considered unknown
Minimum area	2000	Moving region sizes below value are ignored
Consecutive detections	5	Required number of consecutive predictions for login to occur
Logout time	5	Amount of seconds to pass before logout occurs

Table 6.5: Static parameters

6.4.2 Accuracy

First the accuracy of the system will be discussed. All combinations of the detection and recognition algorithms were tested on the dataset. The results from these tests can be seen in figure 6.1.

At first look there are two things that stand out: 1) *Facenet* is superior and 2) the *HOG* detector gives surprisingly poor results. The first is not surprising as we expect

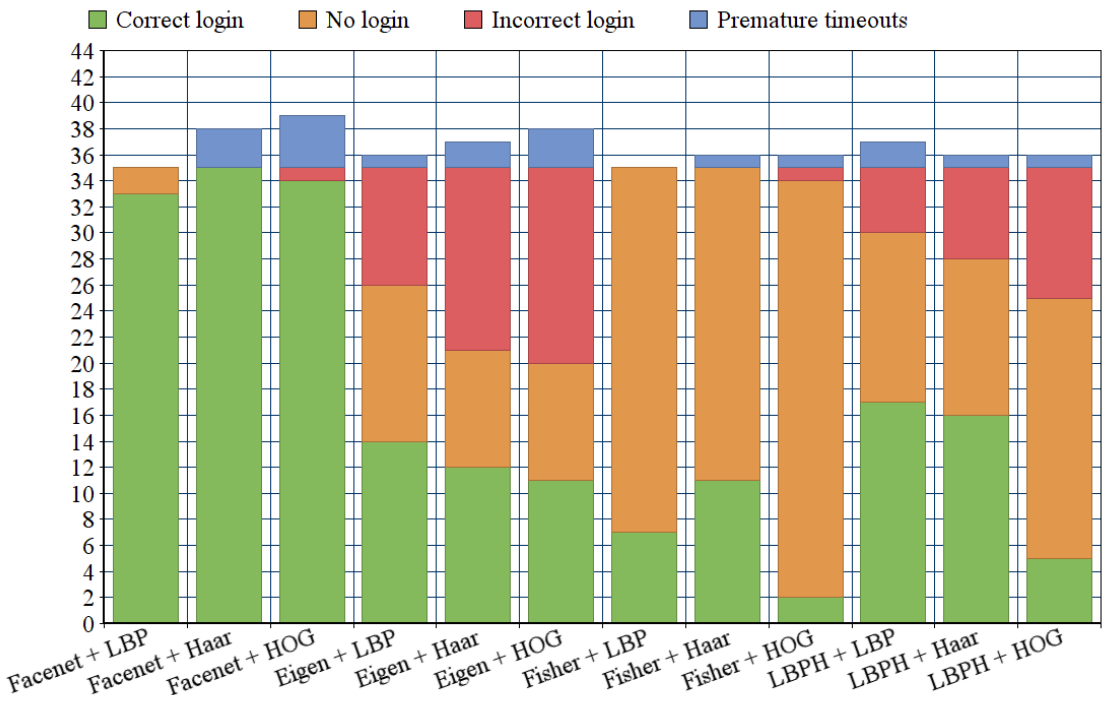


Figure 6.1: Accuracy of all combinations of detection and recognition algorithms

the Facenet method to be superior given how vastly it outperforms the other methods on the LFW dataset. It is however more surprising that the HOG face detector performs so poorly in combination with the other face recognition methods, given that dlib's HOG face classifier implementation is less susceptible to slight occlusions and handles sideway faces better.

As table 6.6 shows, the HOG face detector outperforms both *haar-cascades* and *lbp-cascades* at detecting less false positives. So why come the results are so poor on all methods but Facenet? The most probable reason is that the haar-cascade face detector was used to extract the faces that the other recognition algorithms were trained on. This causes a mismatch between the face boundaries so that the face extracted during testing has completely different dimensions and aspect ratio. The image is scaled to correspond to the dimension used during training, but the facial area that is extracted is more zoomed in with the HOG detector compared to the cascades. The end result is that the face parts do not align and face recognition becomes difficult for the older methods.

The data in table 6.6 is obtained by counting the number of faces detected over the whole test data and compare it with the number of those faces that was given a positive recognition. The number of faces detected that did not return a positive recognition gives the ratio of false positives. The slightly improved Facenet approach (discussed soon) was used to automate the recognition process, rather than manually checking if suggested face was actually a face or image noise. As a consequence the data in table

Algorithm/Results	Detection	Faces Recognized	False positives ratio
HOG detector	5940	5908	0.539%
Haar-Cascade	5942	5721	3.720%
LBP-Cascade	4701	4667	0.724%

Table 6.6: Comparison of total faces detected and false positives ratio.

6.6 are not guaranteed to be 100% valid as even the Facenet approach can make errors. But the overall trend of the recognition algorithms becomes clear. Haar-cascades finds the most false positives. The HOG detector has the lowest false positives ratio, even if it suggests as many faces as the haar-cascade. LBP-cascade has a decent false positives ratio, but finds far less faces than the HOG detector.

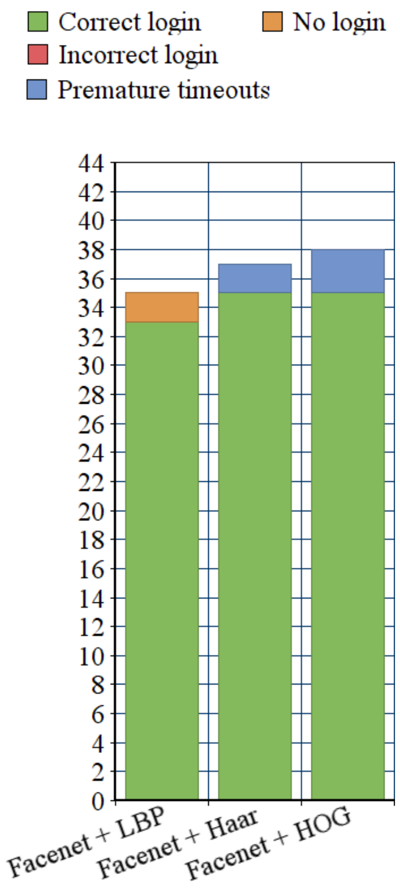


Figure 6.2: Improved Facenet by voting for the most common prediction.

The modern Facenet method does also compare the new faces with the same training set. But because it uses convolution it is able to tolerate translation in the image and is more invariant to dimensions and zoom. To verify that this assumption is true the eigenfaces, fisher-faces and LBPH recognizers could have been trained on another dataset that was created with the HOG face detector. But there are a couple of reasons not to bother do this. Primarily because, as we will see, the HOG face detector is considerably slower than the other face detectors. The only reason *not* to use the Facenet recognizer is if any of the other recognition methods can be used on the RPI. Hence combining any those recognizers with the HOG detector is meaningless as the combination is not suitable for the RPI (too slow) nor suitable on a remote architecture (inferior to Facenet).

Of the older approaches the LBPH recognizer correctly logs in the user on roughly 50% of the test data. Unlike eigenfaces and fisherfaces the LBPH approach compares the new face with each face in the training data independently. That is it only applies the LBPH method to each training image and then votes for the person whose LBPH is overall most similar to the new person's LBPH. Eigenfaces and fisherfaces on the other hand attempts to find a global mathematical descriptor of the training data. The poorer results obtained from eigenfaces and fisherfaces could be the fault of the algorithms simply being old and outdated, but is also very likely that my training data is not varied enough when it comes to different face expressions, lighting and angles. The training images were extracted from video of the subject in an uncontrolled environment. Many of the subjects were taken with a different camera, at a different location, with a different angle, different lighting conditions and different backgrounds. Because those two approaches attempts to find a global descriptor it is difficult to create a training set where all the face images have the same properties when it comes to dimensions and translation in the frame.

Perhaps tweaking the training set, whether it be increasing the number of images of each subject or aligning all the images perfectly, could improve its performance, but it is important to remember: *it should be easy for the average user to consistently update, add or delete the set of recognized faces!* Every time a change is to be made to the eigenface or fisherface model the new subject's training images should match the same conditions as the existing images. Also it requires training the model anew for every change. The Facenet method is far simpler to configure as it does not make any assumptions about the training set. Every time the system is booted it propagates the training data through the network and saves the encodings. Then for every new face it simply compares that face's encoding among the set of known encodings.

In figure 6.1 we see that even if Facenet performs extremely well it does get some incorrect logins and timeouts. As mentioned Facenet takes a list of image encodings, compares those with the unknown encoding, and return a list of true or false for every known encoding. True if it believes the unknown face is this person, false if not. This means that when the test performed in 6.1 just used one image of each test subject, it would have to chose randomly which person to select if multiple indices were true. Simply increasing the number of images of each test subject enables the system to make more qualified predictions in the cases where the returned list has conflicting truth values. A simple scheme was implemented where the system used 9 images per user. The system then counts the number of images who matched the unknown face and predicts the user with the most matches as correct. Should it still be a tie the system will choose randomly as before.

Figure 6.2 shows the improvement to the system when using Facenet with 9 images per user. The premature logouts that occur when user with HOG and haar-cascades are due to, as we will see next section, that HOG detector and haar-cascades are significantly slower than the LBP-cascade face detector. Since the logout time used in these experiments are just 5 seconds, the slower methods suffer more. Alternatively the logout time could be measured in frames rather than seconds. Or extending the logout margins. This metric is just a matter of preference: faster logout versus more robustness.

6.4.3 Execution Speed

The other important aspect to the user recognition system is its execution speed. In this section the execution speed of the various parts - background subtraction, detection and recognition - will be analyzed. A comparison is made between the run times when the system is executed either locally on the RPI or remotely on my laptop. The hardware specifications are listed in table 6.7. However it is the relative difference in execution speed between the various methods that are of importance.

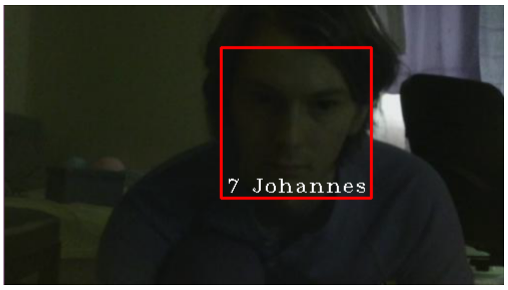
Hardware/System	Acer Aspire V3-574G	Raspberry Pi 3 Model B
CPU	Intel Core i7 (5. generation) 5500U / 2.4 GHz dual-core	1.2GHz 64-bit quad-core ARMv8 CPU
GPU	NVIDIA GeForce 940M - 2 GB DDR3 VRAM	Dual Core VideoCore IV® Multimedia Co-Processor
RAM	8GB DDR3L SDRAM	1GB LPDDR2

Table 6.7: List of hardware specification for the tested devices.

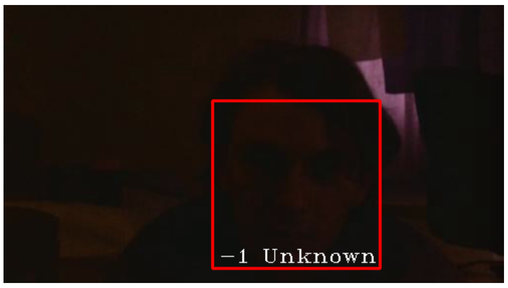
Background Subtraction and Face Detection

Figure 6.4[3] compares the execution times of the face detection methods with and without background subtraction - shown in light and dark colors respectively. The execution times in the figure are acquired by logging the time spent executing the relevant functions averaged by the total number of times the function was called over the testing set. Of course the time spent subtracting the moving regions must be accounted for as well.

The speed increase gained by background subtraction indicated in figure 6.4 does not necessarily reflect the effect it would have in a real application setting. The training data is collected from multiple various location with minimal time to learn the background model. It is therefore natural to believe that the speed increase in a real application setting could exceed those of test data because it will remain in a stationary location for a longer period of time. Also in a real application setting the face detected to no face detected ratio would, over the lifetime of the program, be very small. In the test data however, this ratio is rather high. This is only a hypothesis and no tests have been conducted as to measure the effectiveness of the system as the running time of the program increases [4].



(a) 9:30 am



(b) 9:30 pm

Figure 6.3: Images taken by the mirror in the morning and in the evening during summer.

As seen in figure 6.3 the images become very dark because the mirror only allows part

[3]I noticed just before the deadline that all the following figure have wrongly labelled y-axes. It should have been labelled *seconds* not *milliseconds*

[4]I have had it running for some hours without noticing any issues. I leave proper testing of the effectiveness of the user recognition system to future work

of the light to flow through. The images during the day are already dark, and during the evening it eventually gets so dark facial recognition becomes impossible. This poses a problem because with the current background subtraction system as a movement threshold must be set manually. But the same value is unlikely to work optimally at the different times of the day. Figure 6.4 demonstrates that background subtraction can have substantial speed improvement on some of the methods. And as demonstrated in figure 6.5, face detection is the most time consuming step in the pipeline.

We see from figure 6.4 that in both cases HOG face detector is the most time consuming detection method. But comparing figure 6.4a and 6.4b we can see that the difference is far more drastic on the RPI. This probably has to do with the dlib library lack of support for RPI's SIMD instruction set. More in that in the next section. Further it seems the ratio between haar-cascades and lbp-cascades remain about the same. The lbp-cascade with background subtraction runs at 0.119 seconds per frame on the RPI, while on the laptop it uses 0.014 seconds. About a tenth of the time required. By comparison the HOG detector on the laptop runs at 0.079 seconds with background subtraction on the laptop - faster than either detection method when executed at the RPI. The drastic increase of the HOG detector's speed on the RPI makes the ratio between haar-cascades and lbp-cascades a bit tough to see. But haar cascades runs at 0.321 seconds per frame - nearly three times that of the haar cascades.

Face Recognition

Table 6.8 presents the execution time of the various face recognition methods both on the RPI and the Acer Aspire V3-574G. The laptop results will be discussed first. We observe that the Facenet method is the most computationally heavy, about twice the time required as LBPH. On the other hand Eigenfaces is the least computationally heavy. According to the table however, there is not much of a time difference between using 1 image for Facenet or 9 images. That is because the image of the unknown person is only propagated through the neural network *once*. Propagation is the time consuming step and is only done once regardless of the number of images per person. But classification is fast. As mentioned the classification step is simply the euclidean distance between the unknown encoding and a known encoding followed by thresholding against a tolerance value to receive a binary classification. When using an optimized mathematics library such as *numpy* calculating the euclidean norm between arrays is fast.

Algorithm	LBPH	Fisherfaces	Eigenfaces	Facenet-1 image	Facenet-9 images
Time (RPI)	0.023	0.002	0.008	3.590	no data
Time (Laptop)	0.0029	0.0003	0.0008	0.0060	0.0069
RPI/Laptop ratio	7.93	6.25	9.87	598	no data

Table 6.8: Comparison of recognition algorithms execution time

So far it seems the Facenet method is the best option; it has superior accuracy and only slightly longer execution time, and when looking at figure 6.5 it is apparent that the small increase in recognition time is dwarfed compared to the detection time of either detection algorithm. *However* when testing the execution speed of the Facenet method on the RPI the RPI's limitation becomes apparent. *Encoding one single face on the RPI takes roughly 3.6 seconds. About 600 times more than the time it takes on a modern laptop!*

That means the startup process alone will take

$$boot\_time = e \cdot u \cdot n$$

, where e is encoding time, u is number of users, and n is number of images per user. For a system with 10 users and 10 images per user, simply starting the system would take 360 seconds. Sadly this means using Facenet on a local architecture on the RPI is out of the question. Looking at figure 6.5b we see that the other methods does not have such a drastic time increase on the RPI, but rather roughly 10 times longer execution times. So why is it the Facenet method ends up having more than 600 times longer execution speed? It is because running a feed-forward pass on the deep neural network is a huge operation that is basically all matrix multiplications. The laptop used in the remote architecture has a nvidia GPU with CUDA[5] installed which means the feed-forward pass will be run on the GPU in parallel. Even if the device used in a remote architecture does not use CUDA, any modern intel CPU has support for SSE4/AVX SIMD[6] instructions that still parallelizes the math in the CPU for a huge speed boost. As shown in table 6.7 the RPI has a ARM CPU that does not support SSE instructions, rather it has its own set of SIMD instructions called NEON. And unfortunately for us, the dlib implementation of Facenet that we use does not support NEON.

It is also worth discussing the number of frames each method has to process before a login occurs. This does implicitly tell us something about the consistency of each methods predictions.

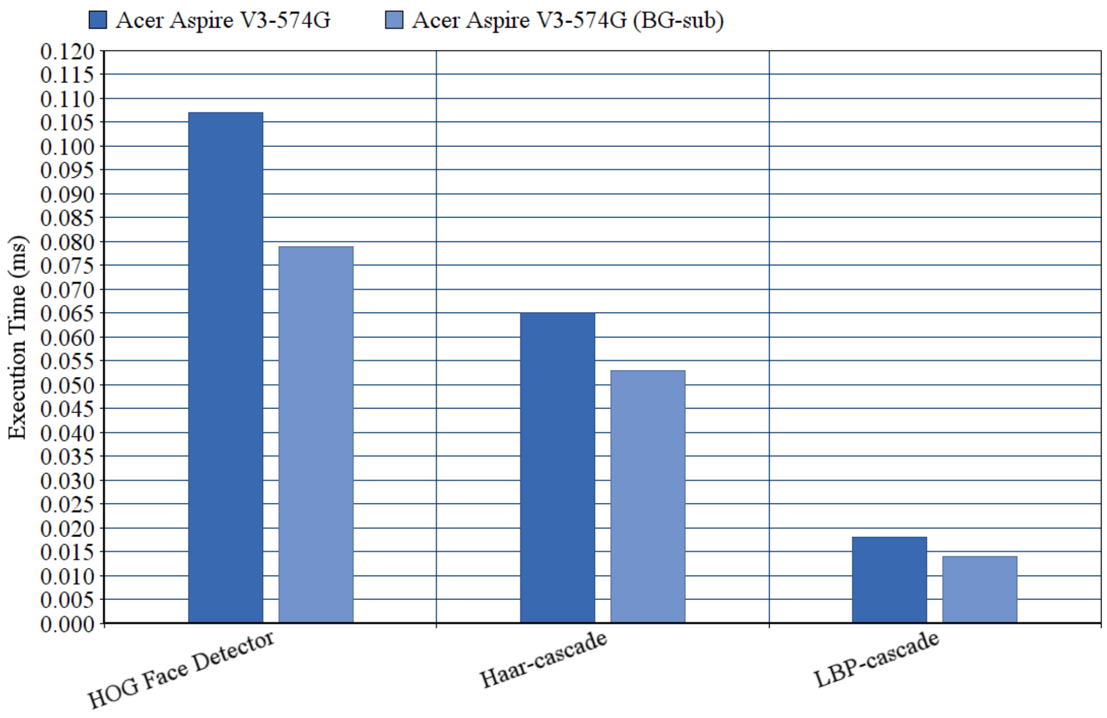
Algorithm/Results	Mean	Median	standard deviation	variance
Facenet	15.6	14.0	9.0	81.9
Eigenfaces	23.3	19.5	18.4	337.0
Fisherfaces	82.6	66.0	67.6	4567.0
LBPH	40.8	28.0	36.1	1306.9

Table 6.9: Statistical overview of the number of frames required per login for each recognition method

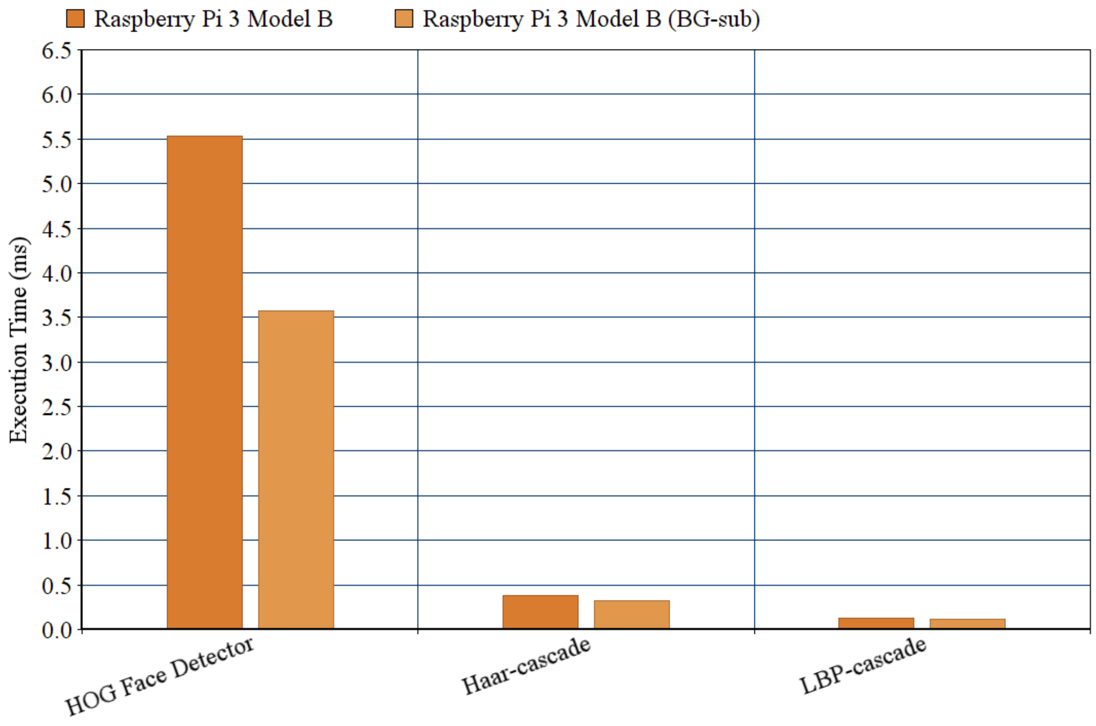
As we see from table 6.9 Facenet is the method with the smallest consistent number of frames required for a login.

[5]Cuda is a deep neural network library (cuDNN) for nvidia GPU's that is purely optimized for neural network tasks

[6]SIMD(Single instruction, multiple data) is a type of parallel architecture. All parallel units share the same instruction, but they each operate on different data elements. As DNN's feed-forward is in simple terms a single instruction (multiply) and multiple data (neuron values to multiply).

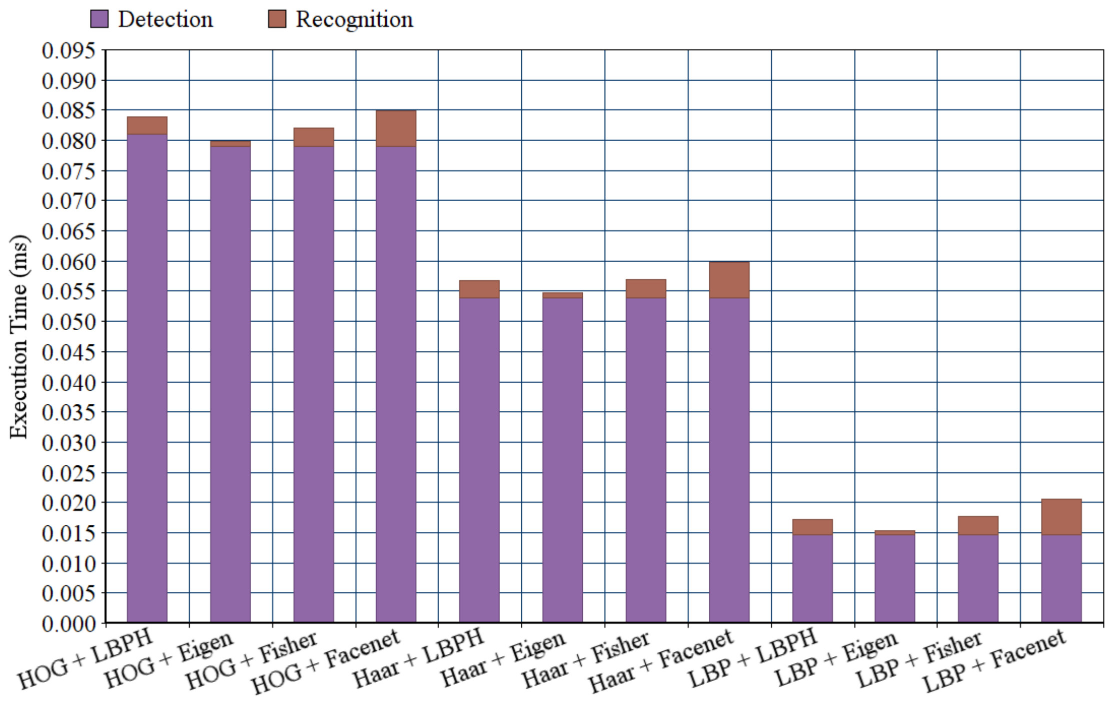


(a) Comparison of execution times of the detection algorithms on Acer Aspire V3-574G

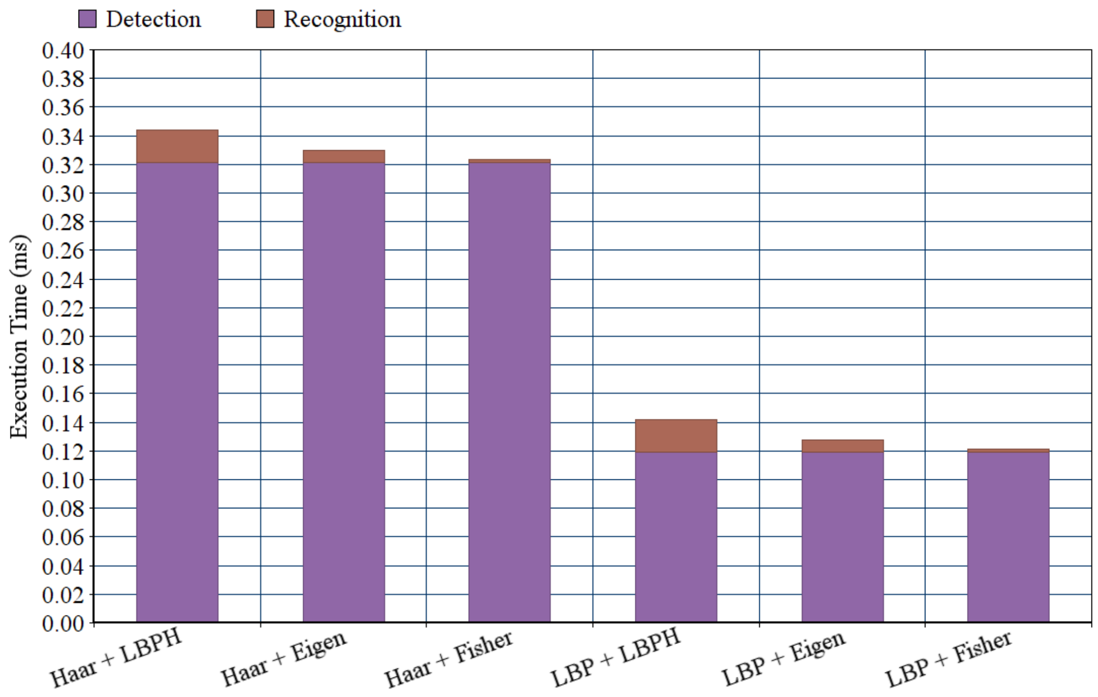


(b) Comparison of execution times of the detection algorithms on RPI.

Figure 6.4: Execution times of detection algorithms on RPI and Acer-Aspire V3-574G



(a) Comparison of execution times of the combined system on the remote architecture with background subtraction enabled.



(b) Comparison of execution times of the combined system locally on the RPI with background subtraction enabled.

Figure 6.5: Total execution times for every combination of methods for both local and remote architecture

7 Evaluation and Conclusion

This final chapter will wrap up this thesis by concluding the results from the previous chapters and discuss whether the local architecture or the remote architecture is recommended for applying the user recognition system. Finally ideas and improvements that can be utilized in furthering the project is summarized in section 7.4

7.1 Evaluation

Chapter 6 highlighted many properties of the detection and recognition algorithms that were evaluated. The two most key conclusions we can draw from those results are that: 1) face detection algorithms has the highest impact on speed, and 2) face recognition algorithms has the highest impact on accuracy on the testing data. This is altogether not that surprising. It is only natural that the method to extract relevant features from a face matters most when it comes to accuracy.

Further it was made clear that one cannot expect the speed of the user recognition system to be linearly comparable across the tested devices. Due to ineffective parallelization on the RPI, the HOG face detector and Facenet recognition suffered a 4500% and 598000%, respectively, increase in execution time when transferred to the RPI. This tremendous increase renders these methods unusable if the system is to run on the RPI. By comparison the remaining methods had a roughly 600-1000% time increase when transferred to the RPI.

That means the possible combination of methods in the pipeline is halved for the RPI. The combination that received the highest accuracy on the test data was using LBP-cascade for face detection and LBPH for face recognition, with 17 correct logins, 13 no-logins and 5 incorrect logins. The total execution time for this configuration (given in figure 6.5b) was 0.14 seconds for each frame. By using the statistics regarding the number of frames required for a login event to occur for each of the recognition algorithms given in table 6.9, the average login time on the RPI with this configuration would be $0.14s \cdot 28.0 = 3.92s$. This is the best case scenario. In other words, every time you wished to use the mirror, you would wait roughly 4 seconds for the system to make a login request that, according to the test data, will only log you in correctly 50% of the time. *However* one very important thing to note is that it did fail 50% to recognize different persons. If the system was able to recognize you once, it would recognize you again. It would be more accuracy to say that it will fail to recognize 50% of the users.

In the case where a remote architecture is applied there is no question the Facenet recognition method should be applied. By doing a similar login delay analysis the system could theoretically login a user in 0.28 seconds at a maximum of 50 fps. Though this metric is not the single deciding factor of the fps when using the remote architecture. Remember in the remote architecture the video feed is streamed as an IP camera. There the fps is also limited by the fps the IP camera can provide. Currently it is limited at 25 fps. A higher fps will create more bandwidth and there is little point in processing any faster than this. There is limited movement between each frame.

The speed metrics alone might suggest that LBP-cascade is the obvious choice, but when referring back to figure 6.2 we see that LBP-cascade failed to login a user in two of the videos. Inspecting the test data showed that it was the same person in both clips. 10 subjects are too few to draw any grand conclusion if the haar-cascade is worth using over lbp-cascade based on its one error. However, considering the haar-cascade is still able to login the average user in 0.82 seconds, it might be worth prioritizing this extra accuracy at the cost of slower login speed. Further, the difference between a 0.28 seconds and 0.82 seconds login delay is so small it can be hard for humans to perceive the difference. Consider the impact of denying one user access to the system (because LBP-cascade did not find a face) versus the small login delay increase of 0.54 seconds, I believe the first issue is far more critical than the second.

Detection method/Metric	login delay	fps
HOG	$0.084s \cdot 14 = 1.176s$	11.9 fps
HAAR	$0.059s \cdot 14 = 0.826s$	16.9 fps
LBP	$0.02s \cdot 14 = 0.28s$	50 fps

Table 7.1: Comparison of login delay using the Facenet method combined with different detection methods. The 14 constant is the median from table 6.9 for Facenet

Also, the data set used only had 10 subjects spread across 35 test instances. In machine learning this is a very small dataset. Though, if a larger data set is made, more powerful hardware would be required for running the tests. On the Acer Aspire V3-574G running all the tests would take roughly 15-25 minutes depending on the configurations.

7.2 Discussion

The first part of this section will re-examine the research questions from the introduction and discuss whether we were able to answer these questions and what the consequences of our findings has been.

The research question most of this thesis has been devoted to is *Can we build a user recognition system that can run on the RPI at an acceptable level?* As of now the conclusion is: **no it cannot.** As already thoroughly explained, the RPI's lack of parallelization support is the limiting factor of enabling the user recognition system to run locally. Deep learning has come to stay; solving most complex recognition tasks. As the evaluation of the old methods show; their performance on our collected test data does not even come close to the performance of Facenet. I think any further focus should be directed at finding neural network implementations that has been optimized for the

NEON SIMD instructions of the ARM CPU the RPI uses, *if* a local architecture is to be pursued.

The follow-up research question was *Propose an alternative system architecture if the previous research question is unsuccessful.* A simple remote architecture was proposed where the RPI simply streams the recorded video over the local network, acting as an IP camera. A remote computer will continuously read the IP camera and send login or logout request to the RPI. There are two primary reasons as to why I think using a remote architecture is the way to go: 1) the speed and accuracy of most modern laptops will by far exceed the performance of the system running locally on the RPI, and 2) it is a solution that will be able to scale when expanding the functionality of the intelligent mirror. If the system is implemented on the RPI, its computational resources are already exhausted, leaving no possibilities for expanding the system.

The second part of the discussion relate to the research questions *Explore the existing implementations of intelligent mirrors. If any, what types of artificial intelligence do they apply?* and *Build a physical prototype of the intelligent mirror.* To answer the first research question we found examples of intelligent mirrors such as the **wize mirror** whose goal was to create an unobtrusive intelligent health monitoring mirror. The project was a success and the development team aim to commercialize the mirror in the future. However most of the projects in the realm of intelligent mirrors had their focus on creating *smart* mirrors rather than *intelligent* mirrors. Microsoft released their take on the intelligent mirror that included facial recognition and customized interface profiles for each user - the exact same topic that has been the major focus of this thesis. However I discovered this late when I was far a long in developing the user recognition system. Microsoft's library uses their cloud service for matching users, so it is a remote architecture akin to the remote architecture proposed in this thesis. Had I discovered this service earlier it would be natural to compare the system developed here to that system. That will have to be postponed for future work.

The final research question was *Build a physical prototype of the intelligent mirror.* To this end a minimalist design was created for a frame-less mirror that was as thin as possible. A thorough step-by-step guide to the assembly of the prototype can be found in appendix A. There are a few important lessons learned I would like to detail. The properties of both the mirror and screen are very important to create a bright and clear GUI. When the two-way mirror was ordered I failed to ask for the percentage of light that was reflected versus what would be able to pass through. This is a very important property that must be considered for two reasons: a too high reflected light percentage will make the GUI hard to see, and it will also make the camera footage very dark. On the other side, a too low reflected light percentage will make reflection poor and too much of the screen will bleed through the mirror. I have not been in contact again with the mirror manufacturer, but the mirror used in the prototype had a higher reflection percentage than ideal.

The reflected light percentage alone does not determine how clear the GUI will be. A high quality screen is able to work well with more reflective mirrors. While lower quality monitors need a mirror with low reflection percentage to make the GUI clear. To obtain the best effect, or in other words, the best illusion of the mirror itself being the screen, the dark background of the GUI should be totally reflected, so only the white colors pass through. However as mentioned most LCD monitors uses background lighting to light up the diodes. This means the black pixels will be lit as well. The screen used in the prototype is such a LCD screen. Unfortunately the GUI will be near impossible to see when viewed at a side angle. Any viewing angle that is not perpendicular to the surface will be poor. However the mirror is designed to be highly modular, so that any 16" screen can be replaced at ease. For a better GUI the screen should be upgraded, but this will of course increase the overall price of the mirror.

With our mirror being highly reflective, the original PiCamera had large issues giving bright enough images as our mirror reflects most of the light. Switching to the NoIR camera partly solved this issue, but the user recognition system is still rendered unusable during the times of the day where natural light is sparse.

7.3 Contributions

The contributions made in this thesis are as follows. First a simple minimalist design for the physical part of the intelligent mirror was created and subsequently a corresponding prototype was made. I created a user recognition system that was highly modular so various pre-existing libraries of famous algorithms could be thoroughly compared. A simple login/logout system was made by keeping track of the user recognition system's user predictions over a certain amount of past frames. This simple paradigm proved to be a quite fast and responsive system when coupled with the Facenet network. Finally a test data set consisting of 35 videos was collected and ideal login and logout frame numbers was manually annotated in order to perform extensive accuracy testing of various face detection and face recognition algorithms.

7.4 Future Work

The topic of intelligent mirrors is a huge topic. Future work could aim to implement any of the unused scenarios listed in chapter 2, such as adding gesture control, communicating with the mirror by voice or integrating the intelligent mirror into an already existing smart home economy.

The face detection and face recognition methods tested in this thesis are only a small sub-set of all research done in the field. Future work could explore if other methods are able to provide satisfactory results on an embedded device. Additionally other embedded device options beside the RPI should be explored. Until the RPI is upgraded to include a basic GPU able to run a feed-forward on deep neural networks, or neural network implementations has been optimized for the NEON SIMD instructions used by RPI's ARM processor, we should look to other embedded device possibilities.

The current login system is naive in the sense that if a user was logged in, it would never revert this decision until the defined logout interval has passed. If later predictions suggests that the initial login decision was wrong, the system should be able to revert the original decision and login what is later believed to be the true user, without having to wait for the originally believed user to be logged out before the correct one can be logged in.

As discussed, face detection is the most time consuming step. The background subtraction step was under-prioritized in this thesis. The experiment results suggest that a robust background subtraction system can greatly reduce the execution time of face detection methods. This step should be researched further in the setting of intelligent mirrors. Running the user recognition system on an embedded devices is all about reduce computation time wherever possible.

Future projects should consider replacing the camera with a full-on night vision camera so the camera is able to provide valid footage for the user recognition system regardless of lighting conditions. Current face recognition methods is intended for use with greyscale images, but experiments should be done to examine if night vision images is compatible with current algorithms.

Bibliography

T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, Dec 2006. ISSN 0162-8828. doi: 10.1109/TPAMI.2006.244.

P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, Jul 1997. ISSN 0162-8828. doi: 10.1109/34.598228.

Vijayadharan SuseelaKumari HariKrishnan Chidambaram Sethukkarasi et al. Interactive mirror for smart home. *International Journal on Advances in Intelligent Systems*, 9:148–160, 2016.

Trevor Darrell Christopher Richard Wren, Ali Azarbayejani et al. Pfinder: Real-time tracking of the human body. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 19(7):780 – 785, July 1997.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1:886–893, 2005.

Alexander Nikic Frank Honold Felix Schüssel Daniel Besserer, Johannes Bäurle et al. Fitmirror: A smart mirror for positive affect in everyday user morning routines. November 2016.

Sara Colantonio Giuseppe Coppini Franco Chiarugi, Eirini Christinaki et al. A virtual individual's model based on facial expression analysis: a non-intrusive approach for wellbeing monitoring and selfmanagement. *13th IEEE International Conference on BioInformatics and BioEngineering*, November 2013.

B. Froba and C. Kublbeck. Real-time face detection using edge-orientation matching. *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, pages 342–347, May 2001.

Frischholz Jesorsky, Kirchberg. Robust face detection using the hausdorff distance. *Proceedings of the Third International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 90–95, 2001.

P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for realtime tracking with shadow detection. *In Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems*, September 2001.

V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874, June 2014. doi: 10.1109/CVPR.2014.241.

Shengcai Liao, Xiangxin Zhu, Zhen Lei, Zhang, et al. Learning multi-scale block local binary patterns for face recognition. *Proceedings of the 2007 International Conference on Advances in Biometrics*, pages 828–837, 2007.

F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, June 2015. doi: 10.1109/CVPR.2015.7298682.

Chris Stauffer and W.E.L Grimson. Adaptive background mixture models for real-time tracking. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:246–252, August 1999.

M. Turk and A. Pentland. Learning multi-scale block local binary patterns for face recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, Jan 1991.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*, 1:I–511–I–518, 2001.

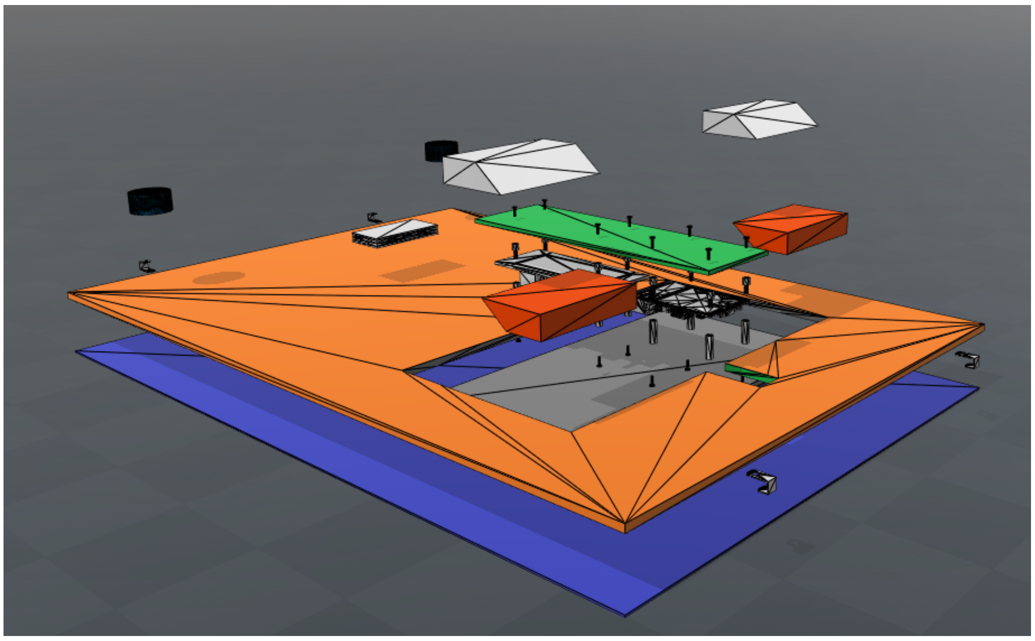
Sara Colantonio Giorgos Giannakakis Yasmina Andreu, Franco Chiarugi et al. Wize mirror - a smart, multisensory cardio-metabolic risk monitoring system. *Computer Vision and Image Understanding*, 148:3–22, July 2016.

Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. *Proceedings of the 17th International Conference on Pattern Recognition*, 2:28–31, August 2004.

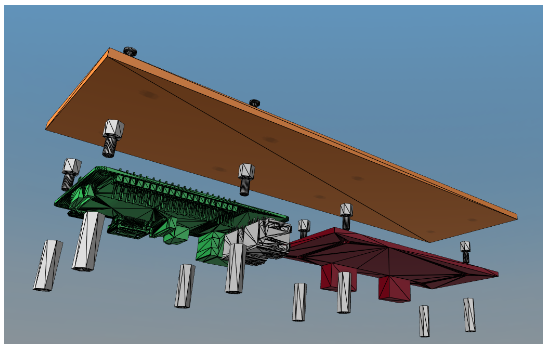
Z. Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. 2006.

Appendix A

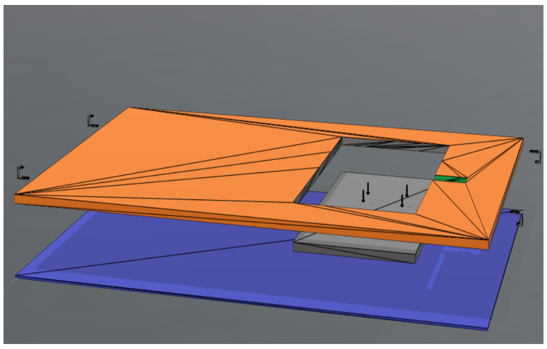
This appendix will detail how the physical intelligent mirror was built, an estimate of the total cost, and a guide for installing all necessary software to get the mirror up and running. Hopefully this appendix will serve as a useful reference for anyone trying to build their own smart mirror.



(a) Blueprint



(b) Back-cover blueprint



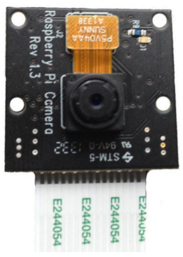
(c) Main frame blueprint

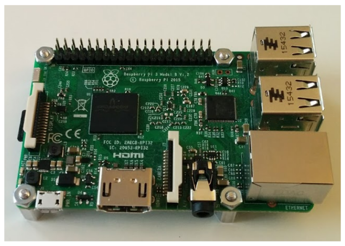
Figure 1: 3D modelled blueprints

1 Inventory list

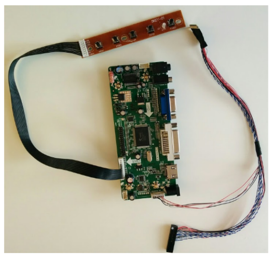
The major components are listed in figure 3. Additionally you are going to need some screws and a lot of black tape. The tools you will need are: hammer, drill, knife, ruler, pencil and a saw.

(a) 15.6" monitor


(b) NoIR camera


(c) RPI 3 model B

(d) USB DC 5V to 12V converter


(e) Controller board

(f) USB-hub

(g) Two way mirror

(h) miscellaneous PCB screws

(i) Mirror hinges

(j) Carbon fiber roll

(k) Wooden plate

Figure 2: Collection of the major components necessary

2 Step by step mirror assembly

Before I began building, I made a 3D blueprint of the smart mirror. This way I could prototype several different ideas without wasting resources and money. The blueprint can be seen in figure 1 and I have also broken the blueprint into two sub-parts, shown in figure 1b and figure 1c, for easier reference. The building process consists of building the back-cover (figure 1b) for mounting the hardware, and the main frame (figure 1c).

I will not be providing any specific measurements, as the basic design could work with other hardware. The principles remain the same. Because the back-cover is detachable this also makes it possible to replace the monitor as long as it is not larger than the initial size.

2.1 Building the back-cover

The back cover serves as a mounting stand for the raspberry pi and the controller board. This way the hardware can be easily removed from the rest of the mirror as well as providing a simple and clean look. The reasoning for mounting the hardware on a separate piece of wood is so that the camera can be mounted above the monitor. In order for the PiCamera to be above the monitor, the raspberry pi and controller board would have to be mounted directly behind the monitor, which would not have been possible. By mounting the hardware on a separate piece of wood, the hardware can be elevated to "float" behind the monitor.

Table 1 will list the different types of mounting stands and screws used in the build for easy reference.

- Step 1. Cut a rectangular piece of wood to serve as the mounting stand. Measure accordingly
- Step 2. Measure where the mounting stands must be installed to line up with the hardware holes
- Step 3.1 Drill holes halfway through the plate in each of the four corners. The hole should be about the size as the female end of [MFS1].
- Step 3.2 Drill four holes for the RPI and four holes for the controller board. The hole should be about the size of the male end of [RMF1] for the RPI and the size of the male end of [MFS2] for the controller board. Make sure not to drill all the way through. **NOTE**: most wooden plates are created by gluing smaller layers together. By noting the number of layers drilled through you can use this as a depth measurement to ensure all holes are equally deep. It is vital that all holes are equally deep, otherwise it can be challenging to mount the hardware
- **Optional for step 4-6: Put a few drops of superglue in the hole prior to the mounting stands to make it more rigid.** NOTE: make sure the standoffs are as perfectly perpendicular to the board as possible, a few degrees off will render you unable to mount the hardware. This is especially important if glue is applied

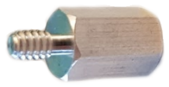
Image	Name	Code
	M3 6mm+6mm Brass Male-Female Spacer	[MFS1]
	M3 10mm+6mm Brass Male-Female Spacer	[MFS2]
	M3 20mm Brass Female-Female Spacer	[FFS4]
	M3 8mm Stainless Steel Screw	[SRW1]
	3/8" Male to Female Aluminum Standoffs (M3 x 9.5mm)	[RMF1]
	1/4" Zinc Screws (M3 x 6.35mm)	[RSRW1]

Table 1: Overview of the mounting stands and screws used. The codes are used in the text as reference

- once it has dried there is no way back. I messed up one of my standoffs so the screw would not fit. However, three out of four standoffs are sufficient for the hardware to be properly mounted

- Step 4. Screw down the four pcb standoffs [RMF1] male-end down in the pre-drilled holes for the RPI
- Step 5. Screw down the four pcb standoffs [MFS2] male-end down in the pre-drilled holes for the controller board
- Step 6. Hammer down the four pcb standoffs [MFS1] female-end down for the back-cover in the pre-drilled holes
- *All the necessary mounting stands has been installed.*
- Step 7. Optional. Cover the wooden plate in carbon fiber.

- Step 8. Attach the RPI and controller board to the mounting stands using the matching screws [RSRW1] and [SRW1] respectively
- Step 9. Extend the corner standoffs using the remaining PCB screws to get the desired height

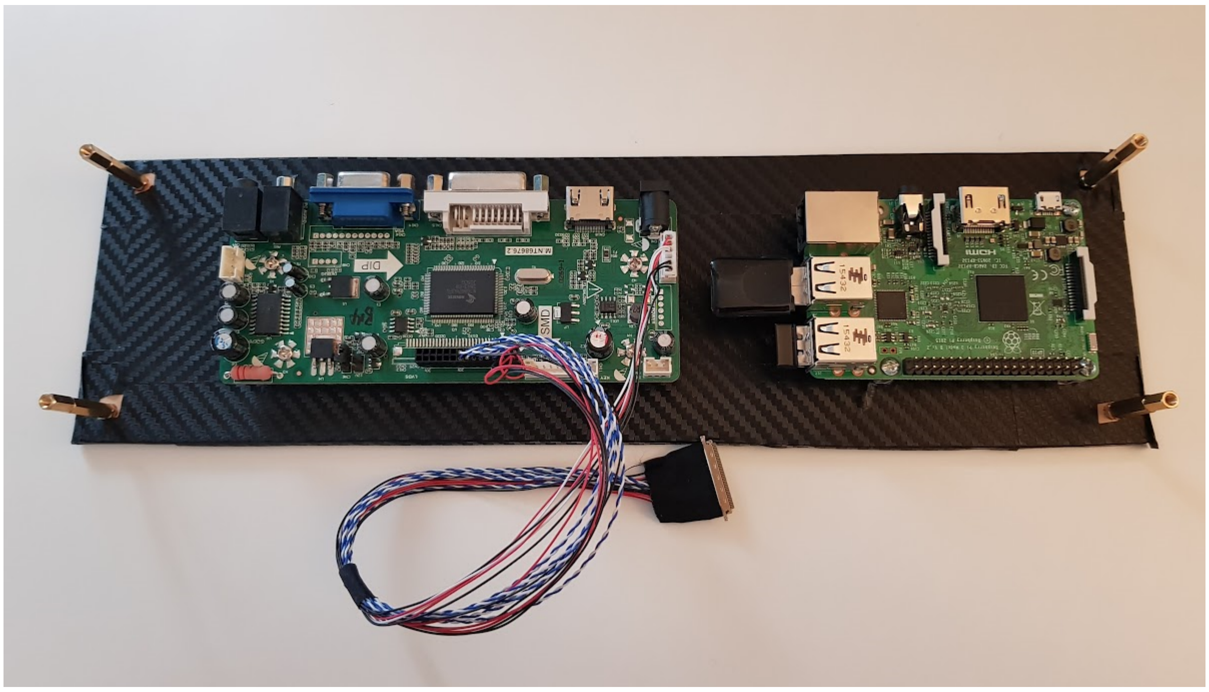


Figure 3: Hardware mounting board

It would be ideal if it was possible to use the same PCB set to mount the RPI as those used to mount the controller board. Even though both were noted to be *M3* the pcb screws used for the controlled board was slightly larger than the RPI pcb holes. I bought my set of pcb standoffs from ebay so the full product details can be vague or incomplete. The RPI pcb standoffs was bought from `modmypi.com`. For future projects the easiest way to reduce the overall depth of the mirror would be to install smaller standoffs for the RPI and controller board. In this installation the hardware is mounted quite high, allowing for good air flow. If future projects attempt to lower the hardware height, the heat generation needs to be accounted for. I have not done any thorough analysis on the temperature of the hardware other than made sure the current air flow does not cause any problems.

Mistakes: The major design flaw I did not uncover while prototyping was the way the back-cover was intended to be attached to the main frame. The idea was to have pcb standoffs with the male end pointing outward from both the back-cover and the main frame, then use a female-female pcb standoff to attach the back-cover to the main frame. However, naturally, this is impossible: the hinges requires rotation to be made clockwise on both ends of the female-female standoff. This is fine for one of the male pcbs, but since the other is rotated 180° it's relative required rotation will be anti-clockwise. Hence my prototype idea to attach the back-cover to the main frame was flawed. **The easy fix for this malfunction would be to 3D-print some custom female-female pcb-standoffs that had reversed hinges on either end.** The current solution was to remove the female-male pcbs attached to the main frame and rather attach them to the back-cover. The back-cover is then simply attached to the main frame by hammering it down in the holes the female-male pcbs used to inhabit. It works, but is not as elegant as the prototyped implementation.

2.2 Building the main frame

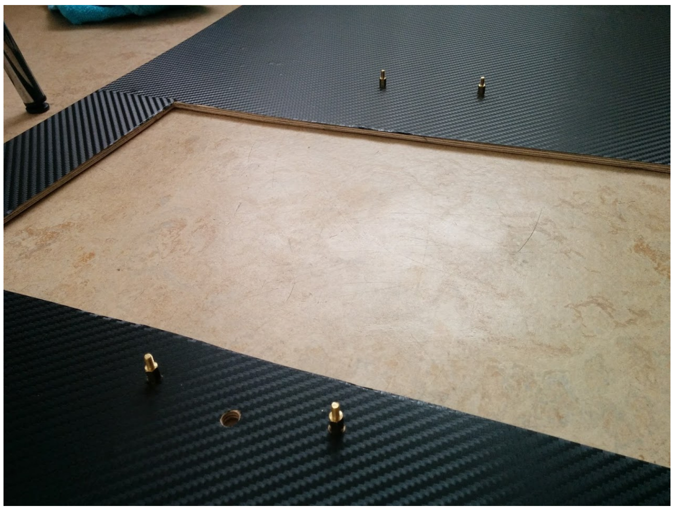
The main frame was built almost identically to the 3D modelled blueprint shown in figure 1c. The idea is to use a thin wooden plate to hold everything together and to strengthen the construction so it can be attached to a wall.

- Step 1. Measure and cut out a wooden plate that matches the dimensions fo your mirror
- Step 2. Carve a hole for the monitor at the height you want it. Try to keep it as tight as possible so the monitor is held in place. However the monitor will be taped around the edges both in the front and the back, so worry not if the hole turned out a bit too large.
- Step 3. Drill a hole above the monitor for the camera to peek through. Try to match the width of the hole to that of the camera lens. This is the only step that varies from the prototyped blueprint. In the 3D modelled I carved out a section for the whole camera to sit in. During construction I figured this was redundant and a simple hole for the camera lens sufficed.
- Step 4. Smooth down all the edges and corners of the wooden plate with sandpaper.
- Step 5. Drill holes on the backside of the wooden plate so the back-cover can be attached.
- step 6. Attach the mirror hinges (**not as shown in the blueprint**). Two evenly spaced at the bottom to carry the weight. And two on both sides at about two-thirds of the height. Attach the side hinges with a few mm margin - it does not need to fit the mirror exactly, gravity will keep it in place. Giving the side hinges some margin will also make it easier to add and remove the mirror. **NOTE:** ideally hinges would be attached with nuts and bolts, but the screws need to fit the hinges perfectly or else it will scratch the mirror. I could not find any nuts and bolts that fit the hinges so I had to use the screws that came with the hinges. The problem is that these will pierce through the wooden plate. These are sharp and hazardous, but with the proper tools the excess parts could be trimmed or cut off.

- Step 7. Coat the backside of the wooden plate in carbon fiber. Cut pieces that are wider than the frame itself, so it can be wrapped around the edge and reach 3-5 cm around to the front. **TIPS:** The tape holding the mirror in place does not look very good. For a future project I would suggest making the carbon fiber cover most of the back side of the whole and just cut a whole for the connector and remove the glue from the carbon fiber. This way it will look a lot more professional from the backside. The monitor can still be taken in and out from the front side of the mirror instead.
- Step 8. Coat the entire front side in black tape. Simply tape over the carbon fiber that was wrapped around the front. This is important because the properties of a two-way mirror requires a dark background to reflect as much light as possible. Tape slightly over the edges of the monitor hole. But you only want to cover the bezels of the monitor, be sure not to tape so it would cover the actual screen.
- Step 9. Insert the monitor and tape around the edges so it is tight and secured.
- Step 10. Locate the camera lens hole and back-cover attachment holes and pierce the carbon-fiber so the holes are accessible.
- Step 11. Slide the mirror in place
- Step 12. Use double-sided tape to place the controller board buttons on the side on the backside.
- Step 13. Push the back-cover in place in the pre-made holes on the back of the mirror.
- Step 14. Power up the smart mirror!

2.3 Unfinished touches

The majority of the smart mirror is built but there are a few touches remaining. Namely putting the mirror on the wall and combine the power of the controller board and the RPI so only one power outlet is required to power the entire mirror.



The original blueprint shown in figure 1a outlines the basic idea for mounting the mirror on the wall. The idea is to use french cleats. These can easily be added as the mirror is detachable so that screws can be inserted from the front into the cleats on the back. To keep the mirror parallel to the wall some extenders can be attached to the bottom as shown in the figure.

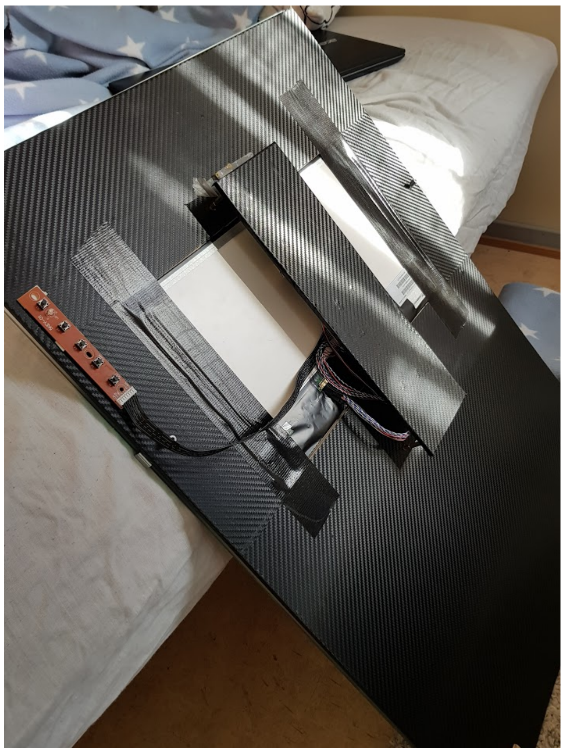


Figure 4: The finished product

In order to power the mirror with a single power outlet, the RPI could be powered from the usb-hub that powers the controller board. The RPI will show an under-voltage warning while running (this can be hidden), but from my experience I do not notice any difference the performance of the RPI. According to other RPI users the under-voltage warning is not severe or poses any danger, but I am not making any statements about that here. I'm simply saying it is fully possible to run the RPI with an under-voltage warning from the usb-hub.