A framework to quantify airport display interaction using machine vision

By Yacob Ben Youb
Amsterdam University of Applied Sciences

1. Preface

Dear reader,

This research is written as a thesis for the Technical Computing graduation program at the Amsterdam University of Applied Sciences. It covers the research and development of different ways to gather data on airport display interaction using machine learning and the visualisation of this data into comprehensible graphs.

This study was conducted at Infologic, a provider of digital information displays which is primarily associated with airports, but also offers commercial store display software. The study was conducted under supervision of Ernst Rijsdijk (Infologic) and Willem Brouwer (Amsterdam University of Applied Sciences).

2. Abstract

Infologic is a company specialized in digital information displays. They write proprietary software for displays and install these along with the necessary hardware for their customers. These customers are primarily airports. Infologic currently provides display technology for several big airports such as Schiphol and Singapore Changi Airport. While Infologic does have some commercial stores using their display software, those are currently not their primary target audience.

Because Airports are limited in space and manpower, they rely on technology to aid them in automating their processes and collecting data. Infologic aims to explore this new field of data collection and discover new methods to gather data at airports. Machine learning can offer new technological solutions to satisfy these demands.

This study focuses on the research of usable technologies and creating a practical prototype of a framework which quantifies interaction with airport displays. The primary goal of this research is to gather physical data without having any sensors present aside from a camera. The study kicks off by researching gaze tracking which has sadly proven to still be too inaccurate for the use case. After this the decision was made to move on to feature recognition. By combining multiple machine learning algorithms such as object tracking and feature detection a framework has been built to gather data about display interaction. The data then gets put through a filtering algorithm to clean out dirty data.

When the filtered information is ready it still needs to be visualized. An interface needs to be built to display the filtered information in a way that is compliant with Infologic's current display technology. This is done with the Dash framework and Pandas library. The entire process results in an easily readable interface with charts which contain information on what the algorithms have observed from the recorded footage and provide insight into how humans have interacted with the displays.

3. Abstract (Dutch)

Infologic is een bedrijf dat zich specialiseert in digitale informatie displays. Ze schrijven voornamelijk software voor displays en leveren dit tezamen met benodigde hardware aan vliegvelden. Momenteel levert Infologic display technologie aan een aantal grote vliegvelden zoals Schiphol en Singapore Changi Airport. Infologics display software wordt ook voor enkele andere commerciële doeleinden wordt gebruikt, zoals in winkels, maar vliegvelden zijn nog hun primaire focus groep.

Omdat vliegvelden gelimiteerd zijn in ruimte en mankracht, helpt technologie zeer bij het automatiseren van processen en het verzamelen van data. Infologic doelt om zich ook te verkennen bij nieuwe technologieën, en machine learning lijkt een interessante sector te zijn.

Deze studie gaat over het maken van een prototype voor het onderzoeken van voor het verzamelen van bepaalde fysieke data. Dit wordt gedaan door middel van praktische applicaties van machine learning algoritmes te onderzoeken. Het doel hiervan is om inzicht te krijgen in het gebruik van display layouts.

Door het combineren van meerdere machine vision algoritmes zoals object tracking en feature recognition is het mogelijk om de van personen voor een camera de leeftijd, gezichtsuitdrukking en het geslacht af te lezen. Na het verzamelen van deze data wordt dit opgeslagen in een soort matrix, welke vervolgens uitgelezen kan worden om statistieken van de data weer te geven.

4. About Infologic

Infologic specializes in hardware and software solutions for Information visualization. With over 30 years of experience and a proven uptime of 99,99% they excel in visualizing real time, rule or event based information in a reliable way. Their goal is to minimize information overload, especially in crowded locations like airports or public transport stations.

People need to be guided in an efficient way to prevent big crowds and possible chaos. When presented with the exact right amount of information, it will bring clarity and peace of mind for their journey. They will feel at ease to spend their time money in shops, restaurants, and other commercial locations. Not to mention that when people are able to better process display information, they will be able to find out where to go faster. This leads to a higher throughput and thus a more profitable usage of the same amount of space.

Infologic aims to improve the efficiency of transportation through the use of information displays. However recently they have become interested in finding new creative solutions to improve the passenger experience. The company is actively experimenting with new technologies to improve their products and to find new solutions for their customer's problems.

Infologic currently provides display technology for several big airports such as Schiphol which had 68.5 million passengers in 2017[1], and Singapore Changi Airport which had 62 million passengers in 2017[2].

Infologic is located in De Goorn, their dedicated software development team consists of around seven employees on site and three who live abroad.

[1] "Schiphol | Jaarlijkse Traffic Reviews." https://www.schiphol.nl/nl/schiphol-group/pagina/traffic-review/. Accessed 2 Aug. 2018.
[2] "Traffic Statistics - Changi Airport." http://www.changiairport.com/corporate/about-us/traffic-statistics.html. Accessed 2 Aug. 2018.

5. Table of Contents

6. Introduction

There are many buzzwords floating around in the world of computer science. “Data science”, “machine vision”, “big data”, “machine learning” and more. There is a lot of hype surrounding these upcoming fields. This is because they provide a new approach to solve existing problems in ways that were previously impossible .

A big misconception which comes with this hype is that some people assume that these fields are a one-size fits all solution. While they can potentially solve previously unsolvable problems, there’s also a possibility that they cannot provide more insight than solutions before them.

Infologic has always had a difficult time quantifying the effectiveness of layouts because they were not able to manually gather metrics on how passengers viewed their display layouts because sight is an intangible metric. They are actively looking for ways to gather this information, but since much of the human visual process happens subcounsciously, methods like surveys have proven ineffective, and lab experiments are often not time efficient for single layouts.

With this study Infologic aims to discover whether they can gather these intangible metrics for display interaction using recent developments in technology.

7. Structure

This chapter describes the general structure of the study. The purpose is to provide a generalised overview of the different sections which are covered throughout the study and to briefly explain what they are for.

The study kicks off by defining the problem. What is the problem that needs to be solved, and what are the constraints of this problem? Multiple aspects of heuristics and testing for existing layout rules and display design are compared in order to decide which metrics should be measured. With this information requirements can be set for a possible solution.

After deciding on the metrics which need to be gathered, experiments are conducted to test which of these different methods can best gather these metrics. Then a prototype is built which combines different methods to gather metrics.

After this, another framework needs to be built to process these metrics and visualize them with statistical tools. This results in an interface which conveys information about airport display interaction.

7.1. Approach

This chapter contains the types of research methods and skills which will be utilised throughout the study.

7.1.1. Research methods

This study is based on a very experimental process akin to a new startup. Most our research revolves around determining whether a solution is a compatible solution for our target environment. Airports are an environment with rather strict rules, and have clear boundaries and restrictions as to which solutions are permissible. The requirements to our solution will be defined in the Problem chapter and they will be used to evaluate our possible solutions. If a solution does not meet these requirements it will most likely not suffice for our problem.

6.2.1.1. Qualitative research methods:

Expert interviews: Our product is a tool designed for a select few users (namely the people who design custom airport display layouts). The main source of validation for our MVP's are expert interviews. Infologic and Schiphol will decide the requirements and validate whether an MVP passes these requirements.

Desk research: While expert interviews are used to validate our MVP's, most of the information gathered to create the MVP's comes from desk research.

6.2.1.2. Quantitative research methods:

Experiments: Because of the requirements, an experimental process which allows for quick iterations would be the optimal approach for our research. The research will adhere to the concepts of the UAAS Lean Startups minor, as well as the testing methods from their courses. The Lean methodology[3] is based on making a Minimum Viable Product (MVP) and validating the viability of the product. If the project manager deems an MVP to have enough potential it can be further refined. Otherwise it is discarded in order to research other possible solutions.

[3] "Validated learning about customers - Startup Lessons Learned." 14 Apr. 2009, http://www.startuplessonslearned.com/2009/04/validated-learning-about-customers.html. Accessed 2 Aug. 2018.

7.2. Problem

This section contains the overview of the problem that needs to be solved. It kicks off with the chapter *Incentive* which explains what Infologic's motivation for this study is, and why the study is being conducted. The chapter *Background information* provides some more info on how this problem has affected airports in the past.

7.2.1. Incentive

To define the problem which needs to be tackled, the study started off with three meetings with Infologic's project manager and their head of technical direction. In these meetings multiple possible ideas with a focus on machine learning have been discussed. These can be found in the attachments 1 and 2. After these meetings Schiphol has also been contacted to discuss which of the ideas they would find most interesting. Together with Infologic they have come to an agreement that they wish to acquire dynamical data on how passengers interact with airport displays. With this they can acquire information on each custom display layout and better understand how certain changes in their layouts affects interaction.

7.2.2. Defining the problem

For many years Infologic has focused on providing a content management system for display systems. Because Infologic has primarily associated themselves with airports they have observed many new experiments and innovations in their field over the last years. With a constant push for efficiency, automation has become a key factor to improving the amount of work that can be done in a limited space.

Most big airports want to create their own custom layout to display information. To realize this Infologic supplies a toolkit to give their customers the ability to create heavily customizable layouts for their displays. This is called their Flight Information and Display Software (or "FIDS" as in the rest of this paper). In the last years Infologic has improved their FIDS by adding new logic such as rule and event based functionality. This makes it easier to manage displays because they can react to external factors, and are not dependent on manual updates to display real-time information for the end user. Most of the standard display information however, is still dependent on the layout created by the customer.

A big problem that comes with giving the product user freedom to decide how to implement their solution is that they can also make mistakes. Besides just giving the customer freedom to implement a product however they want, it is important to give advice on what they should and should not do regarding their layouts. This however comes with the big question: **how** does Infologic advise what airports should and should not do with their layouts?

It can be difficult for airports to predict the consequences of their solutions to problems.
A large part of the problem stems from the fact that the way cultures and people work can be radically different all around the world. This can largely impact the way people perceive and understand information. A good example of this is the Western countries which read from left to right, while Arabic countries read from right to left. Some Asian countries even read from top to bottom. Another good example is the side of a hallway which people tend to walk in. In certain countries people tend to walk forward on the left side, while in others they walk on the right side.

These and a multitude of other factors can cause a lot of confusion for passengers when not taken into account and this is why it is of utmost importance for airports to customize their solutions to fit their own audience instead of simply copying what other airports already do. There is no one-size fits all solution for airports.

Even in the design teams at Schiphol as well as at Infologic there is a lot of internal debate about which layouts are cleaner and more efficient. Not to mention the design aspect of these layouts which needs to look modern and fitting with the airport themes. The big tradeoff between design and efficiency is very much an unsolved equation while they are not able to compare data.

7.2.3. Defining the requirements

Now that the goal has been decided, requirements need to be set in order to decide what the MVP should be judged on. The requirements that have been set by Infologic's project manager and Schiphol for the MVP are as follows:

- Must be space efficient and portable

The solution needs to be able to be installed on almost any airport display, meaning it should not take up more than a small amount of extra space on the display, and should possibly be able to expand to be able to utilise a video feed of a built-in TV camera in the future.

- Provides data about the usage of display software by passengers

The solution needs to gather data about how passenger interact with the display.

- Any heavy computing must happen off-site. An ethernet connection is available.

There is no certainty a monitor allows for the connection of powerful computing hardware on site. It is safe to assume every monitor is able to connect to a fast internet connection which can be used to transfer data.

- Requires minimal upkeep

The solution should run fully automatically and require minimal upkeep. After installation the user should be able to remotely start gathering and access the data without needing manually filter out false positives or restart experiments because they crash.

- Filters out dirty data (will be explained more in-depth in the chapter Data Science)

Whenever data is gathered, this will almost certainly have false-positives included. When obvious false positives are found in the data they need to be filtered out before visualisation.

- Is able to deliver the raw data so that it can be viewed with custom software.

After the data is extracted there should be a possibility for third parties to create their own tools to visualise and analyse this data.

- Is able to display the data and give insight in a comprehensible way

Aside from allowing third parties to analyse the data, an interface should be built which visualises the metrics in an insightful way, such as with charts. Preferably this interface is compliant with Infologic's current technology which is browser based, but this is not required.

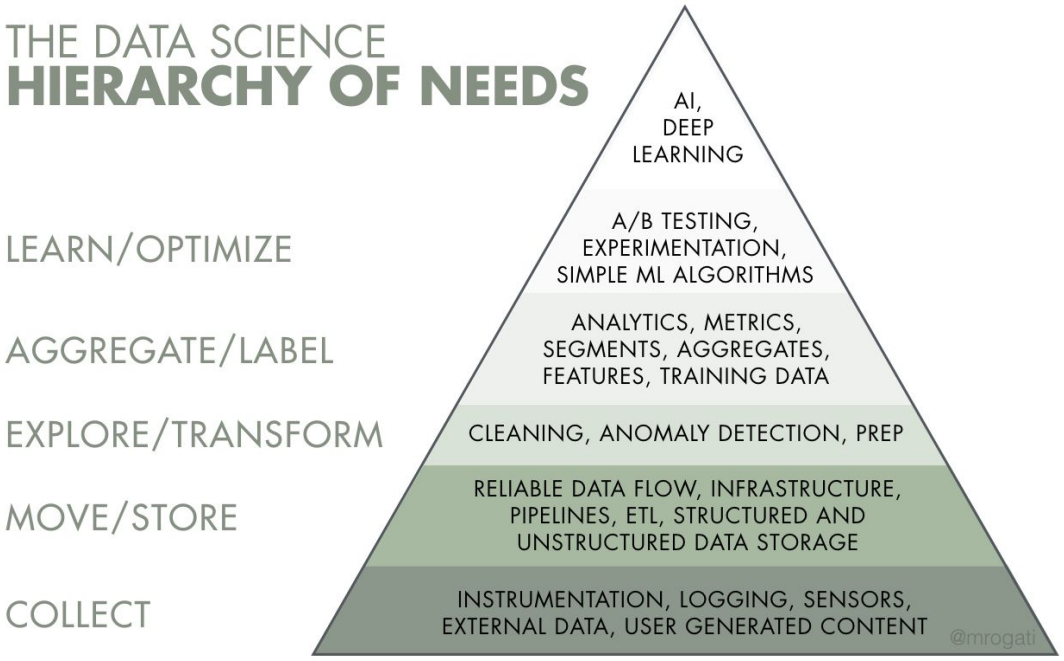
When combining all these demands, it can be summed up as the following: *Data science for airport display analysis*. A major part of the problem we are trying to solve is based in the field of data science. This will be further explained in the next chapter Background Research.

8. Background research

In the *Problem* section the problem has been defined. Now it is time to dive more in depth to examine what a possible solution might look like. For this a closer inspection is taken at the field of data science

8.1. Data Science

Data science can be defined as "An interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from data in various forms, both structured and unstructured"[4] The image below provides a good representation of the different aspects of data science that are covered in this research



Different aspects of Data Science visualised[5]

According to the requirements, almost all aspects of the Data Science process need to be covered in the MVP. This starts at the bottom with the collection of big data, and ends with the visualisation of the data into comprehensible analytics.

[4] "The key word in "Data Science" is not Data, it is ... - Simply Statistics." https://simplystatistics.org/2013/12/12/the-key-word-in-data-science-is-not-data-it-is-science/. Accessed 3 Aug. 2018.

[5] "The AI Hierarchy of Needs – Hacker Noon." 1 Aug. 2017, https://hackernoon.com/the-ai-hierarchy-of-needs-18f111fcc007. Accessed 4 Aug. 2018.

8.2. Data science for airports

In general there are two ways to evaluate the usability of a product: heuristic evaluation and usability testing[6]. A heuristic evaluation is an evaluation done by experts in a field who apply design rules to inspect and evaluate layouts based on existing industry standards. Usability testing however, focuses purely on how users interact with certain layouts, and tests which effects layouts have on the user instead of making assumptions based on previously gathered information. But how can the interaction of users with certain layouts be measured when this data is not available?

Many digital platforms like websites have also dealt with this problem. This is where the concept of heuristics and testing originated. In the infant stages of the web, people designed their own solution and hoped for the best. Sometimes in-house testing was done to gather data and define heuristics which other layouts could be based on. In the last decade however a new approach has emerged: usability testing. This method focuses on gathering data on how real users actually use the system. In web-development this is primarily done with AB testing[7].

What AB testing means in the context of a FIDS is that in a test different types of layouts are tested on users under the same circumstances in order to determine differences between them. The users need to be tested under the same circumstances with as many of the variables in the scenario as equal as possible aside from what needs to be tested.

8.2.1. Current situation

As has been explained earlier, airports deal with many different problems from many different sources. The large amount of different factors at play in every scenario make it incredibly difficult to analyse problems and draw conclusions in a controlled way. In order to find correlations between different factors huge amounts of data are needed to analyse them.

Gathering this amount of data would be impossible for humans because it would consume too much space and too many resources, and could still result in many inaccuracies. A human cannot accurately keep track of many variables at the same time, such as: how long each person in a large crowd remains in front of a monitor, said person's gender as well as their age category.

In many other cases where physical data needs to be gathered, analog sensors can be used to gather input, however those can take up a lot of space and this would not fit the requirements of the project. Not to mention it would be incredibly difficult to gather data about display usage with regular analog sensors. Because of this a more modern approach to the problem might provide a solution.

---

[6] "Usability testing vs. heuristic evaluation - ACM Digital Library." https://dl.acm.org/citation.cfm?id=142179. Accessed 6 Aug. 2018.
[7] "To appear in the Encyclopedia of Machine Learning ... - ExP Platform." Online Controlled Experiments and A/B Tests. Accessed 10 Aug. 2018.

8.3. Display heuristic evaluation

Before starting on the development of an MVP to analyse big data, the focus must be defined. Which factors are relevant to this research? Which factors can feasibly be analysed? By using existing testing principles it is possible to narrow down our focus and decide which factors are most important to airport display development. After this it must be decided which of these factors can be analysed.

According to Christopher Wickens et al. in their book *An Introduction to Human Factors Engineering*[8] there are 13 different important principles for effective display design. These are designed to reduce error rates, improve efficiency and increase user-satisfaction. From these, the most relevant design heuristics for airport display software have been selected.

- *Make displays legible:* Readability and airport layouts are a difficult combination for designers to realise in a correct manner. Because display space is limited, and displays need to show a certain amount of information, airports sometimes choose to lower the font size of their information making it difficult to comprehend for people without optimal eyesight.

Top-down processing: Top-down processing is one of the most important factors of information displays. A simplified explanation of this is how the information is displayed in comparison to the users expectations. We expect earlier flights to be displayed more to the top of our display.

Redundancy gain: Redundancy gain is achieved by conveying the same information with different methods. For FIDS this could for example be changing the color of a delayed flight to red as well as writing it is delayed.

Similarity causes confusion: Showing many similar elements next to each other can make them indistinguishable and cause confusion. Elements like planned departure time and actual departure time could seem like redundant elements

Minimizing information access: When a user has to search in multiple locations to find the information they are looking for, there is a feeling of time and effort associated with it. To minimize this it is important to put frequently associated elements together.

Principle of consistency: Old habits die hard. Especially when an entire ecosystem isn't forced to adapt. Because there are no universal rules of FIDS design and airports deal with people who travel a lot between many different airports, it is important to have a design that does not differ too much from other airports. it is difficult to make drastic layout changes without confusing and upsetting many travellers who habitually navigate through airports. When moving to a new style we have to iterate in small steps for people to gradually adapt.

[8] "An Introduction to Human Factors Engineering - Semantic Scholar." https://pdfs.semanticscholar.org/78e4/3d0b0cad62892b5c9eb17871fcff7cdbd01e.pdf. Accessed 17 Oct. 2018.

8.4. Testing heuristics

Now that it is clear which factors need to be analysed, it is important to find out which of these factors can feasibly be tested, and how data for this testing can be gathered. Traditional display studies are mostly based on web-interaction metrics[9]. The big difference between web-design and airport layouts is that in web-design there are inputs which provide insight into the actions and decisions of the user. By analysing metrics such as where the user clicks, time spent on site, and tracking the mouse, it is possible to analyse factors like interaction rate, visual hotspots and areas of interest. Contrary to websites FIDS displays are not able to gather these types of metrics in the same way, because users do not interact with the FIDS with any input. They only observe the screen and are not able to control it. The traditional way to convert analog data to digital data is with the use of sensors. However, because humans are an extremely irregular factor it is hard to track them with sensors. Furthermore it is not a feasible solution to attach sensors to all passengers to gather data from them because there are far too many passengers to track. So how can this data be gathered without attaching sensors to all the passengers?

When looking at the requirements for the solution at hand, the rising field of machine vision could offer a potent solution. Machine vision can provide solutions to many unanswered problems because it can support methods which were impossible to practically implement otherwise. In traditional sensory methods, a physical signal had to be sent out to process input. Recent developments in the field of machine vision however, have enabled the extraction of certain types of data in real time by simply analysing 2d images[10], even without any additional sensory inputs.

Instead of manually analyzing every aspect of every culture and performing massive studies, it is much faster and more efficient to use automation to provide feedback on what works, and what doesn't. Technology can offer many new ways to gather data from the real world. The goal is to gather analog data and convert this to digital data without obstructing passengers or using spacious equipment. This data can then be used to analyse the current situation. When gathering large amounts of data and finding correlations between it, it is possible to come to conclusions which could previously not be made because there was only a small sample size to analyse. Many of Infologic's design principles are based off of studies which were not conducted in an environment that is remotely equal to an airport. Nor were they conducted using many types of different people from many different cultures to analyse whether there's a difference in reaction between them. A one-size-fits-all solution for airports simply doesn't exist, so it is important to find the right middle ground.

[9] "Web Site Usability, Design, and Performance Metrics - INFORMS ...." https://pubsonline.informs.org/doi/10.1287/isre.13.2.151.88. Accessed 20 Aug. 2018.
[10] "You Only Look Once: Unified, Real-Time Object Detection." 8 Jun. 2015, https://arxiv.org/abs/1506.02640. Accessed 22 Aug. 2018.

8.5. Computer vision frameworks

In the field of computer vision there are many closed source frameworks which can compute in the cloud such as Google Cloud Vision[11], Microsoft Azure Computer Vision[12] and Amazon Rekognition[13]. The problem with these platforms is that their functionality is extremely limited. They seem to mostly focus on object detection (which will be explained later on). While this is part of the process, the entire process needs to encompass much more than just this small part of machine vision. There is only one big framework which allows many computer vision algorithms to be implemented and combined, namely OpenCV. The downside of openCV is that algorithms often need to be implemented manually instead of using premade tools. This costs development time and requires coding knowledge.

8.6. Machine vision languages

At the start of the project it is important to decide the programming language that the project will be built with. In machine learning, and especially machine vision, there are two languages which are very popular: Python and C++. While we personally have more experience with C++, this might not be the best choice for the project. The primary tradeoffs between Python and C++ are that Python is far easier to implement and test, while C++ offers better performance.

With the introduction of GPU accelerated computing however, most of the computing work which runs slower in Python is offloaded to the GPU using libraries such as CUDA or OpenCL [14]. This means that the performance advantages from C++ are mostly nullified.

This leaves the choice only between usability. Python is cross-platform which means binaries don't have to be compiled for different operating systems. This heavily favors Infologic's already cross-platform infrastructure. Python also offers far easier installation of libraries, and has simpler syntax which allows for faster development.

Combining all these factors, the decision has been made to use Python instead of C++ for machine vision programming. As such it is time to start development on the MVP.

[11] "Vision - Google Cloud." https://cloud.google.com/vision/. Accessed 5 Aug. 2018.
[12] "Computer Vision API - Cognitive Services APIs Reference - Microsoft." https://westus.dev.cognitive.microsoft.com/docs/services/5adf991815e1060e6355ad44. Accessed 5 Aug. 2018.
[13] "Amazon Rekognition – Video and Image ...." https://aws.amazon.com/rekognition/. Accessed 7 Aug. 2018.
[14] "Python Vs. C++ for Machine Learning – Language Comparison ...." 24 Sep. 2018, https://www.netguru.com/blog/python-vs.-c-for-machine-learning-language-comparison. Accessed 5 Aug. 2018.

8.6.1. Notes

Because the target audience of the analysis is airports, it can be hard to perform tests on a preferable audience. Airports do not easily allow people to take their own experimental test setups with them, and even a simple recording or the placement of a webcam can require many permissions when used for data analysis. This is mostly because of legal reasons. Even our contact at Schiphol could not easily get us the required permissions for recording because Infologic supplies them with FIDS , but has no permissions for data collection. For this reason the experiments will be performed internally using other types of footage than direct analysis of FIDS at airports, and we will validate our prototype on them before we want to try them out at the airport.

A big advantage of making the system able to analyse recorded footage is that it enables us to test the prototype on stock footage of airport passengers. While this is not entirely representational of the scenario our product will position itself in, it does allow us to test with environments similar to the preferred final product. With this we are able to analyse factors of our algorithms, such as the ability of a tracker to differentiate between humans and moving objects at airports such as suitcases and trolleys.

Because many modern publications come in the form of web articles and extensive blog posts. While these sources are not official papers, their authors are actively at work in the fields they post about and information is provided in their articles to support their claims.

9. Gaze-tracking

After discussing the different possible options to gather information using machine vision with the project manager, such as feature detection and object tracking, gaze tracking seems to provide the most potent solution to test the heuristics. As such it has been chosen as the first experiment.

Gaze tracking is a field which estimates where a person is looking. By looking at many common problems people have at airports, it was decided that analysing how people look at flight information could yield important results for feasibly improving the interface design infologic can offer. Desk research shows that most information on airports is gathered visually. Therefore if the visual process of gathering information is proven, this should provide a better experience for passengers. By correctly allocating the information to places of interest on screen, people are faster to comprehend it, and they can be guided faster and more efficiently to their destination.

This experiment is not only designed to find results pertaining the hypothesis, but also to analyse the general behavior of people looking at information displays and find out whether there are interesting types of behavioral interactions which we might not have noticed before. Based on the results of this experiment we will decide whether gaze-tracking is a feasible method to improve the passenger experience, whether we can discover another interesting interaction which could be improved upon, or whether we need to pivot[15] (switch) to an entirely different idea.

[15] "Pivot, Patch, or Persevere (I Patched the Lean Startup) - Medium." 18 Jan. 2018, https://medium.com/swlh/pivot-patch-or-persevere-i-patched-the-lean-startup-206d63023b9c. Accessed 20 Aug. 2018.

9.1. Tracking types

Following the LEAN method it is important to test the hypothesis as quickly and easily as possible before continuing to build upon it. To find out which method that is, background research has been done concerning the different existing technologies for gaze-tracking to choose the most feasible one.

Five commonly implemented methods of gaze-tracking have been found on which information needed to be searched and compared. The results hereof have been compiled in attachment 3.

From the different factors pertaining to each type of tracking it is safe to conclude that optical gaze-tracking is the most affordable and relatively easy to implement in comparison to other gaze-tracking methods. Optical gaze-tracking is done by analysing normal camera footage such as from a webcam or DSLR camera using only software. Another big factor that played part in the decision to choose for optical tracking is that it allows the tracker to run on recorded footage. This makes it possible analyse the video externally instead of having to run the program in a live environment, and it also allows for a smaller on-site setup.

The goal is to test concepts in the easiest and most affordable way possible. For this, open source and freely available gaze-tracking implementations will be used to validate the concept before deciding whether it is viable. If gaze-tracking seems to be a viable option to gather information about how passengers interact with FIDS layouts, this could be a good reason to dive more in-depth into the subject, and perform more tests with more advanced hardware and/or software. Another big advantage of image based gaze-tracking is that the captured video can be analysed with other machine vision algorithms than gaze-tracking.

9.2. Background research

Now that a possible solution has been found, it is time to analyse how well this solution performs for our use-case. For this, desk research has been done to discover more background information, and comprehend what kind of data needs to be gathered.

9.3. Machine vision and human vision:

To start off it is important to understand the basics of human vision. Human vision currently still far surpasses machine vision in many ways. Research on how the human cognitive system works uncovers extremely interesting concepts, many of which have been applied to machine vision as well. The reason this is important, is because the human brain uses many clever tricks to process data, and filter out the important from the unimportant information. By understanding these tricks it is possible to improve our understanding of what the human sight responds to, and what data correlates with factors like attention and interest. It is even possible to use the tricks human sight cognition uses, to guide someone's view to information hotspots .

A common misconception is that visual cognition is a smooth process full of gentle movements from place to place. There are two types of eye movements[16]: The first is called saccadic eye movement. This is rapid movement of the eye from place to place. Because during saccadic the eye only observes a blurry image, the brain tricks us into thinking the movement was instantaneous.

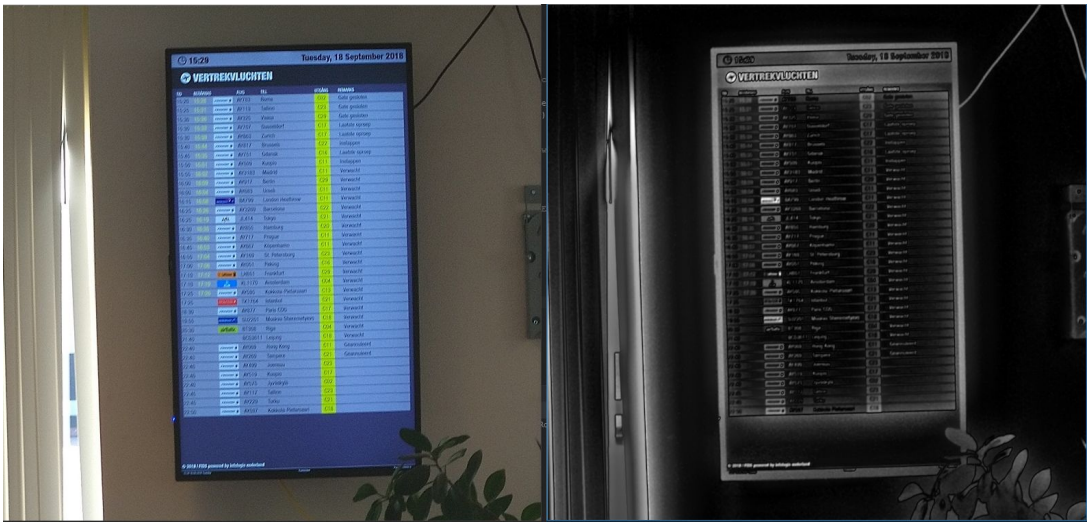
The second action is Fixation. This concerns points of interest that the eye focuses on. Fixation points indicate points of attention.

[16] "PyGaze Analyser – PyGaze." 2 Jun. 2015, http://www.pygaze.org/2015/06/pygaze-analyser/. Accessed 15 Sep. 2018.

9.4. Saliency mapping

Visual saliency[17] are factors which the human visual cognitive system is drawn to. Many eye tracking tests have been performed to understand what draws human attention. Saliency can consist of many different factors such as bright colors and motion, but also the contrast of colors to their surrounding colors. Visual saliency holds a close connection with the earlier discussed heuristic rule "top-down processing"[18]. In this chapter research is done towards predicting visual hotspots in order to compare these with the outcomes of other data, and whether saliency mapping could be used as a heuristic design tool to predict the effectiveness of layouts.

Saliency mapping[19] gives us a visual representation of the visual highlights of our layout. By integrating saliency mapping into the toolkit an image can be tested on its hypothetical layout efficiency.



Saliency mapping applied to a photo of an Infologic display layout to display visual hotspots

Saliency mapping can be useful to predict visual hotspots and points of attraction in FIDS layouts. By placing information in more or less salient positions, combined with eye tracking, it might become possible to discover what information users are looking for. By then placing that information in more visually salient areas it is possible to test whether this leads to increased display efficiency. To realise this an algorithm from the paper Predicting Gaze in Egocentric Video by Learning Task-dependent Attention Transition" (ECCV2018) [20] is used.

[17] "arXiv:1803.09125v2 [cs.CV] 20 Jul 2018." 20 Jul. 2018, https://arxiv.org/pdf/1803.09125. Accessed 10 October. 2018.
[18] "Visual attention: the where, what, how and why of ... - Science Direct." https://www.sciencedirect.com/science/article/pii/S0959438803001053. Accessed 17 October. 2018.
[19] "Computer vision: Taylor Swift saliency mapping – PyGaze." 14 October. 2018, https://www.pygaze.org/2018/05/saliency-mapping-taylor-swift/. Accessed 11 November. 2018
[20] (n.d.). GitHub - hyf015/egocentric-gaze-prediction: Code for the paper .... Retrieved October 10, 2018, from https://github.com/hyf015/egocentric-gaze-prediction

9.5. Gaze-trackers.

Because it would cost over half a year and is most likely worthy of its own thesis, the decision was easily made against developing a new eye-tracker. Contrary to that, the objective is to discover an easy to implement, accurate eye-tracking method which gives us the results that need to be gathered in the easiest way possible. This way it is possible to test whether machine vision based eye-tracking is viable for our use-case or not.

The experiment will be conducted by comparing multiple open and closed source webcam-based gaze-tracking implementations in order to find the most suitable one for display analysis. During this experiment a lot of background knowledge about the different types of gaze trackers and their implementations has been gathered, as well as many other machine vision applications. The following gaze-trackers will be tested in this experiment

Open-source implementations:	**Closed source implementations:**
PyGaze/Opensesame:[21] [22]	Gazepointer[23]
RunwayML-gazecapture[24] [25]	Gazecapture[26]
Ogama[27]	Gazerecorder[28]
WebgazerJS[29]	
Xlabsgaze[30]	

9.6. Hardware setup:

For the first experiment, a Logitech C922 will be used as capture device. This is a high quality webcam which has a better focus than most of its competitors. There was also the possibility to choose the 4K logitech brio, but the lower resolution C922 was chosen because 1080p allows for more rapidly computable frames. As computing device for this test the Latitude E5470 with an Intel 6300U CPU and 16Gb RAM was used.

[21] "PyGaze | Open source eye-tracking software and more.." http://www.pygaze.org/. Accessed 2 Aug . 2018.
[22] "Download // OpenSesame documentation - Cogsci.nl." https://osdoc.cogsci.nl/3.2/download/. Accessed 16 Sep. 2018.
[23] "GazePointer beta – GazeRecorder." http://gazerecorder.christiaanboersma.com/gazepointer-beta/. Accessed 17 Sep. 2018.
[24] "GitHub - oveddan/runwayml-gazecapture: This repository brings the ...." https://github.com/oveddan/runwayml-gazecapture. Accessed 20 Sep. 2018.
[25] "Eye Tracking for Everyone." http://gazecapture.csail.mit.edu/. Accessed 7 Dec. 2018.
[26] "GitHub - CSAILVision/GazeCapture: Eye Tracking for Everyone." https://github.com/CSAILVision/GazeCapture. Accessed 22 Sep. 2018.
[27] "OGAMA." http://www.ogama.net/. Accessed 5 Oct. 2018.
[28] "GazeRecorder download | SourceForge.net." 31 Aug. 2018, https://sourceforge.net/projects/gazerecorder/. Accessed 10 Sep. 2018.
[29] "WebGazer.js: Democratizing Webcam Eye Tracking on the Browser." https://webgazer.cs.brown.edu/. Accessed 26 Sep. 2018.
[30] "xLabs eye, gaze and head tracking via webcam." https://xlabsgaze.com/. Accessed 24 Sep. 2018.

In order to perform the experiment in an airport-like scenario, a stand and monitor provided by Infologic have been used to display our experiment. The experiment has been conducted in the Infologic test room using the Infologic staff as well as some volunteers from a nearby company.

experimental setup for the gaze-tracking experiment.

9.7. Experiment gaze-tracking:

The experiment will focus on testing the accuracy for gaze trackers in multiple use-cases. A test layout has been designed to divide the screen into multiple boxes. Each of these boxes represents a region of the screen. The user stared at each region of the testing screen in a pre-decided order: 1-9-7-3-4-8-6-2-5.

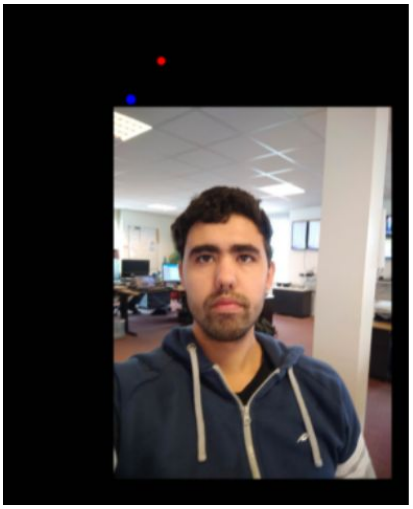
1	2	3
4	5	6
7	8	9

Layout of Gaze-Tracking experimentation test

The first four observations are to test how well the eye-tracker deals with saccadic eye movement from corner to corner, which is usually easier for the tracker to observe, but not always as precise.

Multiple gaze-trackers have been used at multiple distances. The test is performed twice per gaze-tracker at each distance. When an option is available for calibration it has been tested with and without. If an uncalibrated option did not perform accurate for the tests it will be classified as non-functional, and the tracker will be classified as *calibration only* in the results. A few other (mainly older) gaze-trackers have also been tested but because it was obvious in initial testing those did not provide accurate results, they have not been subject to the more extensive test.

RunwayML/Gazecapture
Uncalibrated gaze prediction in action:



9.8. Gaze-tracking results

This chapter contains the condensed results of our experiment with a description of how the trackers performed during testing. All percentages have been rounded off to the nearest fifth.

Open-source implemen-tations	**Gaze-tracking accuracy**	**Special Hard ware**	**Calibrat ion required**	**Description:**
PyGaze/ Opensesame	80% accurate: Depends on hardware used. For a standard webcam no more than 1 metre	Option al	Yes	PyGaze is a python based gaze-tracking toolbox. It seamlessly integrates with OpenSesame:a program to create experiments for psychology, neuroscience, and experimental economics.
RunwayML /gazecapture	60% Accurate up to 1.5 metres directly in front of camera	No	No	This uses a model trained with the MIT CSAIL Vision Eye Tracking for every dataset. What makes this implementation special is that it is able to calculate a gaze point without any calibration, even in static images.The downside of this is that the angle of the camera compared to the monitor and the distance of the tracked target both need to be known. While the distance can be measured with a depth camera, it is too difficult to find the angle the camera is analysing. The tracked target also needs to be positioned directly in front of the camera because it deals very poorly with facial tilt. This tracker is written as an IPython Notebook and has the advantage of being easily executable with cloud services such as Google Collab which outsources computation. It should be noted that for the CSAIL dataset contains images of people standing close to the camera were used, which is most likely why it does not work at a distance.
OGAMA (OpenGaze AndMouse Analyzer)	50% Accurate up to 1 metre in front camera	Yes	Yes	OGAMA is a testsuite for psychological eye-tracking experiments. it is similar to Opensesame but less customizable. This implementation mainly only works with cameras with a depth sensor and without one it requires heavy calibration in advance. While they claim to also have a working webcam based tracker, this tracker has not performed accurately in our limited testing.
WebgazerJS	80% Accurate up to 1.5 metres in a 20 degree angle from the centre.	No	Yes	Webgazer is a Javascript based gaze-tracking implementation, with as primary use case to provide gaze-tracking for websites. There is a calibrated and uncalibrated mode. The uncalibrated mode can be extremely inaccurate, mostly dependent on the angle the head is turned at when looking at the display. Dependent on the tester this can frequently shift the prediction to the left of the screen, no matter the head angle. With an uncalibrated accuracy of only 15% this can not be considered functional
Closed-source	-	-	-	-
Gazepointer	90% Accurate up to 2 metres in a 30 degree angle from the centre	No	Yes	Gazepointer is used by the OptiKey project to gather typing input with a generic webcam for paralysed people who can only move their eyes. This was a fairly accurate capture mechanism even when using only limited calibration, however with good calibration the tracker was extremely accurate as long as the subjects head position and tilt remained stationary.
Gazecapture	90% Accurate up to 1 metre directly in front of	No	Yes	Gazecapture is the engine behind Gazerecorder and Gazeboard. While it does require heavy calibration, it is only functional at a very limited range.

	camera			
Xlabsgaze	70% Accurate even when calibrated at a distance of 1 metre directly in the centre of the camera.Predictions deviate too much to the left.	Yes	Yes	Xlabs is a paid product which offers a demo application for people to test out their product. After having tested their demo it is safe to conclude that they are not any more accurate than most of our open source gaze trackers. This tracker tends to deviate its estimations to the left side of the screen too much.

9.9. Gaze-tracking conclusions:

This chapter discusses the results of the gaze tracking experiments.

9.9.1. Conclusions

In recent years gaze tracking has made much progress, however for our use case there does not yet seem to be an accurate solution available. When gathering big data it is impossible to ask every user to calibrate their gaze in a public space, so using any type of infrared tracking which requires calibration is impossible. An even bigger hindrance is that people do not all stand at the same distance from the camera, which means that the amount of eye movement in relation to what they perceive on the screen can be drastically different. A person standing close to a monitor needs to move their eyes to the side of the monitor to observe what they want to see, while a person who stands further away only needs to slightly drift their eyes to the side. Along with this, gaze-tracking is dependant on the width of your eye sockets which is very hard to measure from a distance.

Another big problem is that it is impossible to use an existing algorithm for airport terminals without first calibrating them specifically to those airport monitors. Generally airport monitors hang high up in the air, which means that the head, as well as the eyes are pivoted upwards far more than when watching any normal monitor. Calculations which are made for cameras positioned right above a monitor become completely incorrect. This was noticed in the experiment when we positioned the camera below the monitor and accuracy drastically improved on a few calibrated algorithms (it did not change on the uncalibrated ones).

There were some very interesting observations made about the CSail dataset which allowed for uncalibrated eye tracking. At a close distance to the monitor this dataset provided us with surprisingly accurate results. The downside of it was when testing from further away (less eye movement) and at an angle off centre.

A possible way to implement uncalibrated gaze-prediction for our use-case would most likely have to come from a deep-learning algorithm with a lot of accurate data input which takes multiple factors into account such as watching distance, head tilt and eye-socket width. This would require an extremely large dataset as initial input for the algorithm. And even with this there is still no guarantee it would result in an accurate uncalibrated remote gaze-tracking algorithm. Leading to the end of this chapter.

9.9.2. Future prospects

Even in the last four years there has been a lot of progress made in machine-learning based tracking technologies. Remote crowd eye-tracking however is one of the few fields which has not accurately been solved yet. In many other fields such as pedestrian tracking which is discussed further in the paper, machine learning has tremendously improved in accuracy within only 3 years. It developed from an inaccurate gimmick which would lose its target as soon as a small part disappeared, to some incredibly robust algorithms which can with some algorithms accurately recognise targets hours after they reappear on screen. There is a very good possibility that remote gaze-tracking technologies will also make tremendous improvements over the coming years, and as such it would be wise to take a look at this technology again in a couple of years.

9.9.3. Pivot

After the experiment, a meeting was held with Infologic's project manager to discuss the viability of gaze-tracking for our use case. From the experiment has been concluded that currently there might currently only be one way to create a working gaze tracker: creating our own dataset.

Gathering a huge dataset from thousands (if not more) people to create a custom gaze-tracker requires a lot of effort and resources, and will with no certainty result in a working gaze-tracker. Together with the project manager the decision was made against continuing in the field of gaze-tracking for now. The complications are too high and the technology is currently still too experimental. It simply delivers too many false-positives to provide us with reliable and usable data for our use-case. Sadly this also means that we will not be able to test the display heuristics for now. There is however a high possibility that it might very well be worth it to revisit this topic in the near future.

10. Feature recognition

Even though it was not yet possible to gather precise information using gaze-tracking, machine learning might still allow for gathering different metrics from passenger interaction with displays, albeit different types of information.

10.1. Background research

There are many other factors which can be analysed other than just gaze-tracking. Factors such as how many people observe a display simultaneously could give insight to the crowdedness in front of a display and could provide incentive to place multiple displays with recurring information. With the ability to track targets on screen individually it becomes possible to track how much time each target has spent in front of a display. If a target can be tracked, there can possibly also be other indirect correlations which can be extracted from facial features

10.2. Setting the requirements

First the discussion was held concerning which data is currently of interest to Infologic. After discussion with the project manager it was decided the following points of interest should be tested for viability. These are not all hard requirements for the MVP but rather guidelines to test what is possible.

- Tracking each individual person in frame and giving them a unique ID
- Starting time of each ID entering the frame
- The last time they were detected in frame.
- Total time a target has been tracked
- Optional: The average emotion of a target
- Optional: The gender of a target
- Optional: The age of a target
- Optional: How many people are in frame
- Saves data in a small format
- Framework must be expandable and allow for additional algorithms to be added.

10.3. Multi Object Tracking

The next experiment is to test whether it is possible to track all passengers on a screen. The aim of this is to extract information from multiple passengers at the same time. A few internal tests were performed to find the best working tracker, and after some more research a condensed group of highly functional MOT (Multi Object Tracking) test databases were found. These contain extensive tests of many different MOT algorithms and comparisons such as quality vs speed. There are several databases such as the CVLibs[31] and the UA-DETRAC [32] database, however the best information on trackers was found in the MOTChallenge.net MOT17 database[33]. This helped decide which object tracker is best for our use-case. The database contains more recent algorithms and performs more extensive testing than other MOT databases. It provides objective results from many different multi object trackers and states what each excels at.

An object tracker mainly consists of two parts, namely the detector and the tracker. In the case of our tracker we first need to detect all humans in a frame. After detecting all possible humans we want to find out whether they have appeared in our video before. In order to do this different detectors need to be compared.

10.4. Hardware setup:

Initially testing was done with an Intel 3770K processor with 16GB RAM, however this proved to be extremely slow (0.4FPS) and hardware acceleration proved necessary in order for Tensorflow to run at feasible speed. One possibility is to write all software to run with cloud computing software such as Google Colab[34], but this would cost a lot of extra time to rewrite each algorithm that needs to be tested. For this purpose an Nvidia 1060 6GB was purchased. This GPU provides entry level CUDA computing capabilities for a reasonable price.

[31] "Tracking - The KITTI Vision Benchmark Suite." http://www.cvlibs.net/datasets/kitti/eval_tracking.php. Accessed 10 Dec. 2018.
[32] "Tracking - The UA-DETRAC Benchmark Suite." http://detrac-db.rit.albany.edu/Tracking. Accessed 13 Jan. 2019.
[33] "MOT Challenge." https://motchallenge.net/. Accessed 22 Oct. 2018.
[34] "TensorFlow with GPU - Colaboratory - Google Colab." https://colab.research.google.com/notebooks/gpu.ipynb. Accessed 14 Jan. 2019.

10.5. Object detection

In order to detect all humans in a frame, an object detector is required. An object detector can detect objects and classify them. For this use-case only humans need to be detected, but the detector still needs to be able to tell the difference between for example a human, and a suitcase. By using a classifier which is able to detect other objects than solely humans it will provide fewer false-positives.

Because of time-constraints it was not possible to extensively test many existing options against each other. Instead only a few of the top detectors have been tested. The most interesting of these detectors were Facebook's Detectron[35], YOLO9000[36], Faster RCNN and SSD Mobilenet[37]. A big advantage in comparison to gaze-tracking, is that the field of object-tracking is far more developed and has some of the most extensive testing datasets. This means that accuracy of algorithms is often extensively tested by third parties, along with their respective speeds. Such a benchmark was also available for human detection. SSD mobilenet was the fastest out of the four but rather imprecise (~18FPS) . Yolo9000 had a far better precision at the cost of some performance (14FPS). Detectron and especially Faster RCNN had minor increase in accuracy on top of YOLO9000, but took a big hit in performance (5FPS and 3FPS respectively).

10.6. Detector of choice

The options were shown and discussed with the project manager and the decision was made to use the YOLO9000 object detector. The accuracy of this detector suffices for the development of the MVP and the speed provides a big computing time bonus. At the time of publishing an improved version of this tracker, namely YOLO V3[38] has been released, which mostly provides small optimisations in speed. In future development this might be taken into consideration as an area of improvement.

[35] "facebookresearch/Detectron - GitHub." https://github.com/facebookresearch/Detectron. Accessed 12 Dec. 2018.
[36] "GitHub - philipperemy/yolo-9000: YOLO9000: Better, Faster, Stronger ...." https://github.com/philipperemy/yolo-9000. Accessed 28 Nov. 2018.
[37] "chuanqi305/MobileNet-SSD - GitHub." https://github.com/chuanqi305/MobileNet-SSD. Accessed 12 Dec. 2018.
[38] "YOLOv3: An Incremental Improvement - Joseph Redmon." https://pjreddie.com/media/files/papers/YOLOv3.pdf. Accessed 28 Nov. 2018.

10.7. Object Recognition

While seemingly similar, object recognition in this use-case fulfills an entirely different purpose. A simplified explanation is that the object recognition algorithm is applied to all the detected objects in frame, and a type of cascade (concatenation of several characteristic classifiers from an object) is made from those objects. With these cascades it can decide whether an object has been detected before. Aside from comparing the frames the recognition algorithm also compares the location of the target, the target's speed and its direction of movement in previous frames. With this it can more accurately predict where the target will be in the next frame. When an object "disappears" from a frame, either by walking out of the frame, or being blocked by something inside of the frame, the object's cascade is temporarily saved. If a similar cascade is detected within a few seconds after, the algorithm will be able to recognise the object instead.

10.7.1. Tracker of choice

Finding a functional tracker proved all but difficult. With a plethora of trackers in the MOT database as well as in public repositories the SORT[39] tracker came out as a quick tracker with good accuracy which fits the requirements perfectly and interfaces well with the YOLO detector. Like the YOLO detector the object tracker suffices for the MVP which means extensive testing was not necessary. A possible alternative that could be implemented in the future is a dual camera tracker[40] which can track far more accurately because it has two dimensions in which it can capture information, but this would limit the project to very specific camera setups.

10.8. MOT Conclusions

After comparing multiple of the highest ranking object trackers, the conclusion was made that the YOLO9000 object detector using the Yolo.h5 model, combined with the SORT tracking algorithm delivers great tracking results for our use-case. This has a combination of good speed while still being very accurate.

[39] "abewley/sort - GitHub." https://github.com/abewley/sort. Accessed 13 Jan. 2019.
[40] "GitHub - mvondracek/VUT-FIT-POVa-2018-Pedestrian-Tracking ...." 2 Nov. 2018, https://github.com/mvondracek/VUT-FIT-POVa-2018-Pedestrian-Tracking. Accessed 2 Nov. 2018.

10.9. Tracker extraction

In order to get a better understanding of our passengers, we also want to be able to link their data together. The goal of tracking passengers is to extract metrics from each individual person and save them separately. Meaning that for each individual passenger we can track for example their emotional state, gender and other metrics which can be detected from the tracked target.

10.9.1. Extracting individual targets

In order to extract every tracked target individually code was needed to extract the sub-image coordinates of the original image where tracking is detected, and putting those in a numpy array. A numpy array can write pixels into an array and so it can be used to extract a sub-region from our image. This region is extracted by using the pixel locations of the tracker detection. Multiple functions have been concatenated into only a few lines of code to manipulate the bounding boxes (note: this caused errors with detected targets which were estimated outside of the image boundaries, so a few additional lines have been added elsewhere to cap the bounding box to the maximum pixel range of the image)

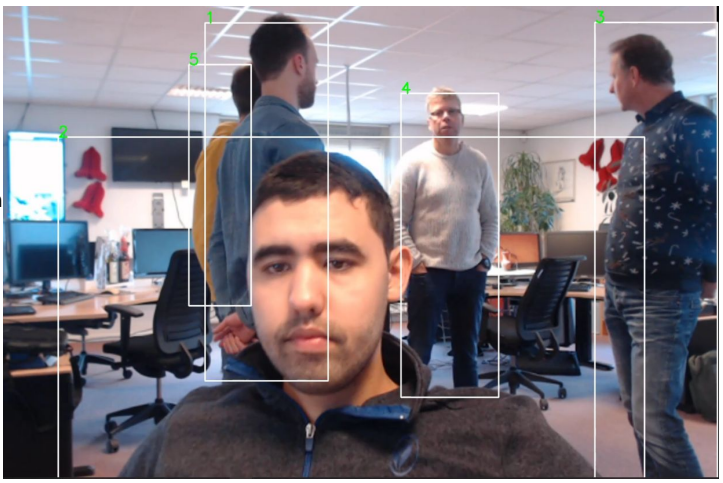
```
#Imagelist is a list of all the images within the tracked bounding boxes of the tracker.
imageList = []
#bbox are the coordinates of a bounding box around the target.
numpArr = np.array(frame[int((bbox[1])):int(bbox[1] + bbox[3]), int(bbox[0]):int(bbox[0]
+ bbox[2])])
```

This numpy array can be appended to an array of extracted images, and allows for individual calculations on each tracked target.

```
imageList.append(numpArr)
for item in (imageList):
        #Run feature analysing code here
```

By analysing only select parts of an image where people are being tracked are extracted. The speed of feature recognition algorithms which will be discussed later on, is vastly improved. Now that we have filtered our tracking targets from the main image, we can apply feature recognition algorithms to each item in our image list. We can even retain the tracking number of each individual, so that we can track which person had which attributes during their tracking period.

10.9.2. Example of the finished object tracker in action:



10.9.3. Facial extraction

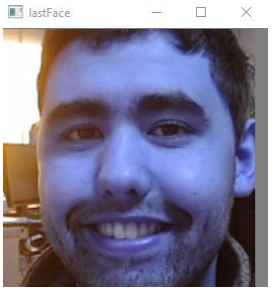
After tracking the tracked targets yet another filter needs to be applied to the tracked targets to detect whether the target is facing their head towards the camera. This is important because in order to extract facial features from the target, it is necessary to detect a frontal image of their face for the feature detectors to work well. The feature detectors will be explained later on. For frontal face recognition there are currently two commonly used options: Haar cascade and HOG (Histogram of oriented gradients). While HOG is slightly faster, there is a distinct difference in "precision and recall"[41].

In machine learning "Precision" is the accuracy of an algorithm. In this case it would mean that whenever a face is detected, it truly is a face and not for example the back of a person. The less mistakes the better. "Recall" in this case is that all faces that can be detected, are detected. Or simply said "it doesn't matter a few backs are detected, as long as all faces are found". While seemingly similar, there is a big difference between the two.

Between the two, Haar cascade had a higher precision, while HOG had a better recall. In total HOG would find faces which the Haar cascade did not detect around 50% of the time, however it was also recognising around the same amount of objects which were not faces as actual faces, thus making it less precise.

Because the algorithm needs to make certain a frontal face has been detected for the other recognition algorithms to work, precision is far more important for this use-case than recall. When an object which is not a face is scanned for emotions and other factors, this could incorporate very strange results in the data, but if a face is temporarily missed for a few frames this would likely not affect the total accuracy by much because no false positives are being added. This is also the reason solutions such as the YOLO face detector[42] have not been used for facial detection. After testing it was found they would recognise faces so accurately that the feature recognition algorithms which will later use them as input, could often not extract useful information from them and gave extremely inaccurate results. It would for example detect faces which were mostly covered up, or even side-views of faces. No deep learning algorithm which would only detect full frontal faces was found .The Haar cascade remains the most accurate option for now.

The result of this Haar cascade is a cascaded image of detected Frontal faces. An example of this can be seen on the right



note, the image looks grey because it is extracted in grey-scale. The feature extractors work with these grey-scale images as well.

[41] "Precision vs Recall – Towards Data Science." 11 May. 2018, https://towardsdatascience.com/precision-vs-recall-386cf9f89488. Accessed 6 Nov. 2018.
[42] "GitHub - dannyblueliu/YOLO-Face-detection." https://github.com/dannyblueliu/YOLO-Face-detection. Accessed 7 Nov. 2018.

10.10. Feature recognition Classifiers

Now the point has been reached where each target is individually tracked and detected frontal faces are extracted from their respective targets. With this a feature extractor can be applied on the image of the frontal face to extract information about its characteristics.

10.10.1. Classifiers

There is far less development ongoing in the field of feature recognition than bigger fields such as pose estimation and multi object tracking, which means it is like gaze-tracking more difficult to find reliable results for testing, as well as there being a more limited selections of algorithms which can be used.

After testing multiple available feature detection classifiers, in small tests, the widely popular Yu4u age-gender prediction model, and the Orriaga gender-emotion detection model have been selected for the MVP because they performed fairly well in small sample tests. These two are in the top modern deep-learning classifiers for their respective fields

Both classifiers claim an accuracy above 95% for gender recognition according to their respective papers but in testing these numbers varied . Because of time constraints a complete experiment for each classifier was not feasible but from quick testing the following results were concluded (note: these tests were both performed tests with live runs of the MVP, and on still stock images)

Type	Accuracy in paper	Testing results	Speed:
Orriaga Gender[43]	96%.	65%	~100ms
Yu4u Gender[44]	-	75%	~150ms
Orriaga Emotion	66%	75%	~100ms
Yu4u Age	96.6% ~10 years	40%	~150ms

[43] "face_classification/report.pdf at master · oarriaga ... - GitHub." https://github.com/oarriaga/face_classification/blob/master/report.pdf. Accessed 13 Jan. 2019.
[44] "Deep expectation of real and apparent age from a single image ...." 20 Jul. 2016, https://www.vision.ee.ethz.ch/publications/papers/articles/eth_biwi_01299.pdf. Accessed 13 Jan. 2019.

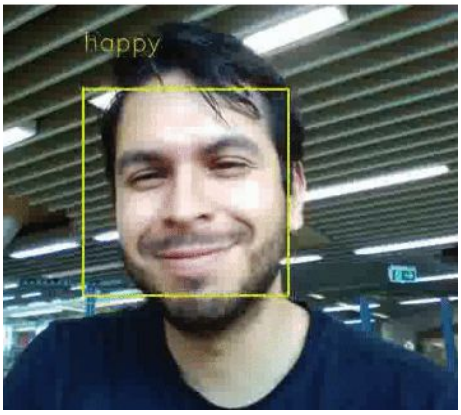
10.10.2. Gender recognition

For the gender recognition algorithm the yu4u CNN network for gender and age detection has been combined with the Orriaga face classification model. Each of these algorithms scan the face, predict facial features and state the certainty of its prediction. The Yu4u model is accurate when providing results with high certainty (70-100%) but when certainty reaches less than 65%, the result holds little merit. This usually happens when the input image does not contain certain features which the algorithm uses to base its decision on. Because of this, when certainty of yu4u gender prediction drops below 65%, the result from the Orriaga gender detection model is used.

10.10.3. Emotion recognition

For emotion recognition the Orriaga face classification model is used. This is one of the top emotion recognition algorithms with almost 1000 forks on github[45]. Many public emotion prediction algorithms are implementations of this model. With an accuracy of 75% the project manager has decided it suffices for the MVP.

Example of live demo of the Orriaga emotion classification



10.10.4. Age recognition

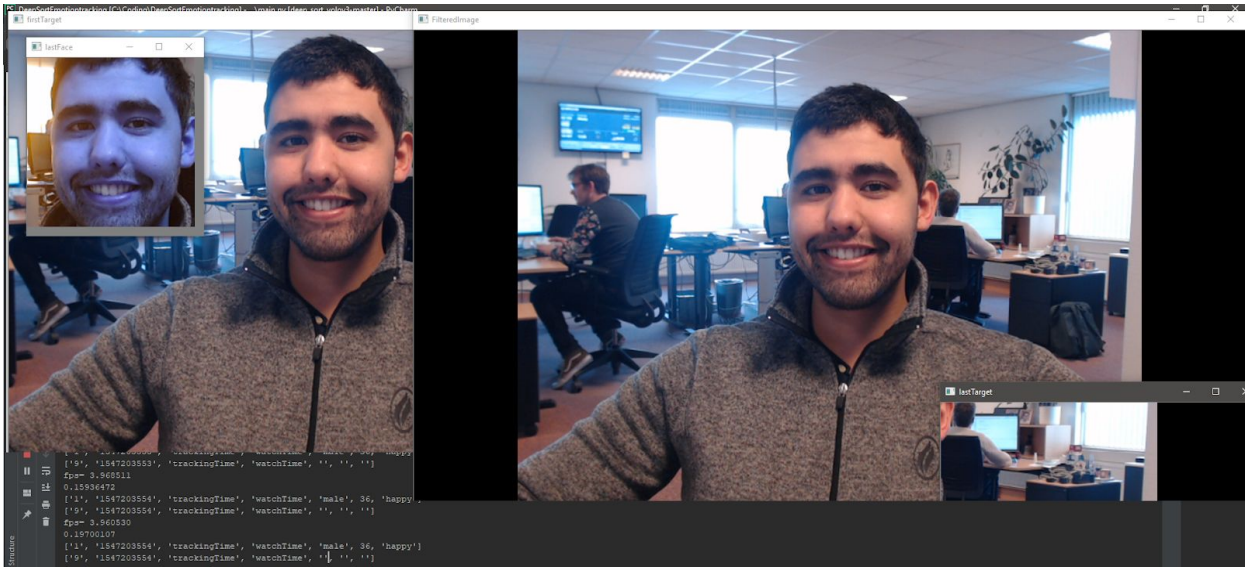
The Yu4u is a widely popular age-recognition algorithm, and with over 200 forks on github[46] currently provides the base for many modern age recognition programs. A problem that was found after integration was that this recognition album was trained on the IMDB Dataset, consists for 86% of actors from ages 30-50. This causes it to perform rather inaccurately and when it is uncertain it always predicts within those age ranges. This was rather strange because the algorithm is revered for its age recognition, while in limited testing it performed far better in gender recognition.

[45] "oarriaga/face_classification - GitHub." https://github.com/oarriaga/face_classification. Accessed 8 Aug. 2018.
[46] "yu4u/age-gender-estimation - GitHub." https://github.com/yu4u/age-gender-estimation. Accessed 10 Aug. 2018.

10.11. Results

After combining all these different algorithms and models, it results in the following:

- A list of the location of each tracked human in the frame
- A list of images of the extracted locations
- If a face is found, their emotion, gender and age are scanned.



Output of a real-time test run with extracted target frames and metrics. (note that the bounding boxes are not rendered because they would cause inaccuracy in feature extraction when two targets overlapped)

10.12. Conclusions

The goal of this undertaking was to create a minimum viable product which shows enough potential to be deemed viable. Because of time constraints it is not possible to optimize these algorithms during the study as the data still needs to be filtered and visualised for the MVP.

After discussing the results of the MVP with the Infologic product manager it has been decided that the current level of accuracy suffices for now.

10.13. Future prospects

For future prospects manually fine tuning the model on a "cleaner" dataset could be an option to increase accuracy. Especially the rather inaccurate age-recognition could be manually trained and fine-tuned on the ChaLearn[47] dataset. Also if a deep-learning model for frontal-face detection is released this could greatly improve accuracy.

[47] "Apparent age V1 (ICCV '15) - ChaLearn." http://chalearnlap.cvc.uab.es/dataset/18/description/. Accessed 19 Aug. 2018.

11. Data processing

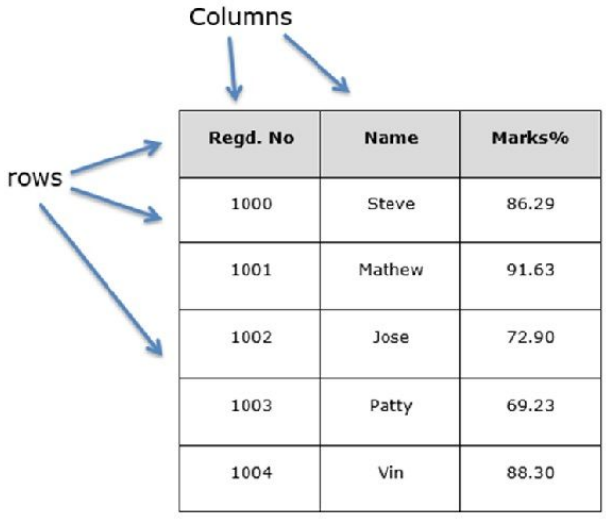
Now that the MVP is able to gather different types of data with the aid of machine vision, the data needs to be processed into usable information[48].

11.1. Data processing language

In data science there are two commonly used languages: Python and R. From desk research we have concluded from multiple sources that aside from a few exceptions these languages have almost all of each others data science functionality. The differences in these languages are rather trivial, they can both perform almost all of the same functionality aside from a few specialised libraries which are not relevant to the project. Because the entire project up till now is written in Python, and we are far more familiar with Python, the choice for Python for this task was easily made.

11.2. The Pandas Library

The Pandas[49] library is the most widely used library for data science in Python. This is a library which provides a python interface with a back-end mostly written in C. It contains heavily optimized functionality for data science such as Dataframes and Series. These contain a matrix-like functionality which is conceptually similar to an SQL table as shown in the image below. It also contains functions to load, manipulate and visualise data, and is interfaceable with many other Python libraries. This can allows for manipulation of the data without having to interface it with another language.



Visualisation of a DataFrame[50]

[48] "A review on coarse warranty data and analysis - ScienceDirect." https://www.sciencedirect.com/science/article/pii/S0951832013000100. Accessed 7 Oct. 2018.
[49] "Pandas." https://pandas.pydata.org/. Accessed 12 Oct. 2018.
[50] "Python Pandas Tutorial - Tutorialspoint." https://www.tutorialspoint.com/python_pandas. Accessed 12 Oct. 2018.

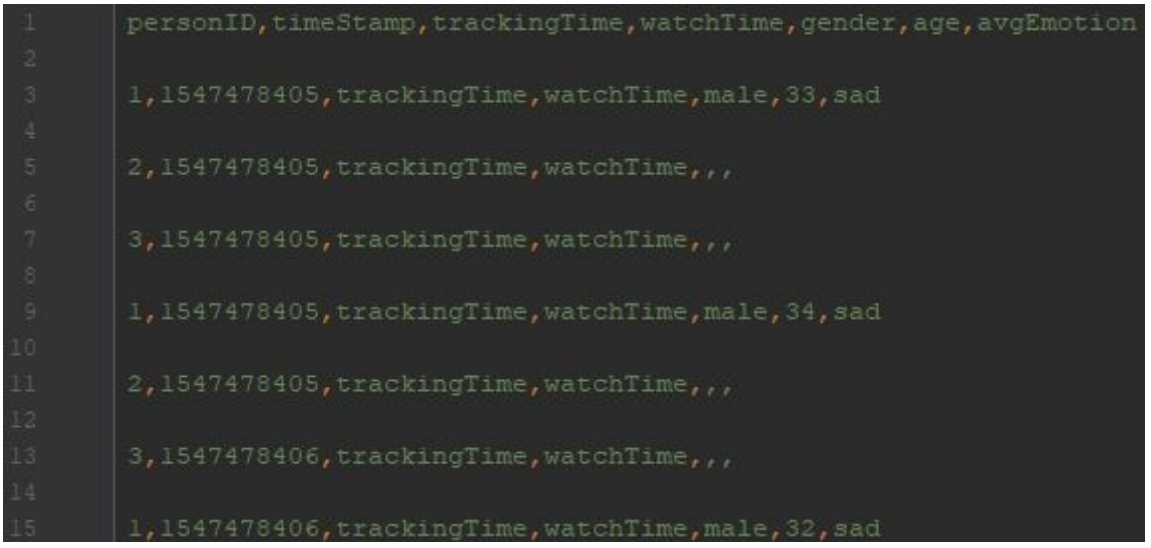
11.3. Data storage

When an experiment is done the user needs to be able to visualise the data from the experiment, and be able to view the data in an insightful and comprehensible way. We need to specify which parts of the data we want to visualize, and/or give the user the option to demand their own types of data correlations.

The experiments also needs to be safe and readable. CSV (Comma Separated Value) files will be used as the standard to save, modify and read experiment data. This format was chosen because CSV files are easy to create and experiment on (only need to put in multiple values separated with commas), quick to manipulate, very versatile, interfaceable among many other languages and data platforms and languages such as Excel, Google Sheets and MYSQL, and the format efficiently stores data.

11.4. Data transformation

For each experiment a new CSV file is created to save the raw experiment data using a custom method which checks the folder for the last experiment and then increments the next experiment number by one.



Sample from a CSV file which contains raw output data.

11.4.1. Optimization

Although an MVP usually doesn't focus on optimization but rather on testing the concepts, when the dataset got big enough (more than ~20.000 results) the filtering code began to run extremely slow and the speed simply wasn't acceptable, even as a prototype, to be called a functional product (half a second of processing time per line cleaned after 15.000). Advanced methods to clean the code needed to be used in order to qualify the MVP as functional. A more technical example of this is given below.

11.4.2. Overwriting the starting time for each target

A good example of a method which would run incrementally slower with a larger dataset, is overwriting the total tracked time for each target. This was done by looping through the entire DataFrame row by row, and each time a new unique ID was found, that ID and its timestamp were saved to a seperate list as an epoch stamp. If an ID was already in the list of uniques the starting time of the first unique ID could easily overwrite its value. However with many unique ID's the list of uniques started to become rather big, and checking each ID in the list manually took a long time.

This is where Pandas functionality comes into play. With Pandas Dataframes there are many different ways to solve a problem. For this case a more advanced method of vectorisation has been used to solve the issue. With this it is possible to execute code hundreds if not thousands of times faster. The results sped up the execution by more than thousandfold when manipulating a large dataset. The code for this looks as follows:

```
for personID in dataframe.personID.unique():
   # First unique and last value it finds, subtract and into trackingTime
   a = dataframe[dataframe.personID == personID].head(1).timeStamp.get_values()
   b = dataframe[dataframe.personID == personID].tail(1).timeStamp.get_values()
   dataframe.loc[dataframe.personID == personID, 'trackingTime'] = b - a
```

For each unique personID, a boolean mask is laid over the dataframe which shows only rows with information from that person. The first timestamp gets subtracted from the last one, leaving the time a person was in frame in total. A vectorised locate method is then used which locates every personID using the same boolean mask, and the "trackingTime" column gets overwritten with the total tracking time.

11.4.3. Gender mode

While it is logical that a single person cannot change their gender in between frames, the gender recognition algorithm does not take previous predictions into account. This means that when a person turns their face or changes their facial expression, they can be recognised as a different gender for a couple of frames. There are also frames in which a gender is not found for a target, but it is possible to attribute them a gender based on previous and later observations of their unique ID.

Using the following code within the personID.unique() loop method above, it is possible to overwrite all gender values with the mode (most occurring) value of that target.

```
modeList = dataframe[dataframe.personID == personID].gender
modeList = modeList.dropna()
if modeList.size > 5:
  a = modeList.mode().values
  dataframe.loc[dataframe.personID == personID, 'gender'] = a
```

An adjustable minimum of 5 detections is required for overwriting, in order to prevent a single false-positive gender detection from overwriting all gender values for that target.

11.4.4. Cleaning data

A big problem with the raw data is that it still contains a lot of "dirty data". These are observations which are obviously incorrect. An example of this is when the tracking algorithm recognizes a new person but isn't able to build a good tracking cascade of them before they switch to another pose. The algorithm will think another new person has been found. In a span of just a few seconds the algorithm can detect many new targets from only one person when it is not able create a good cascade. This creates a slew of false positives which would result in strange result calculating metrics which use the median of the uniqueID's.

In data science such an occurrence is called dirty data[51]. In order to remove these obvious false-positives from the data set a filter needs to be applied to the data. To remove these false positives, an advanced method was written to filter the personID for a certain minimum frequency using multiple boolean masks, a transformation and a group-by function. With this every ID which isn't registered at least x times, is removed from the dataframe.

```
col = 'personID' # Column that needs to be filtered
df = dataFrame[dataFrame.groupby(col)[col].transform('count').ge(minimumCount)]
df.to_csv(self.lastResult(), index=False)
```

11.4.5. Future prospects:

To increase performance of CSV data manipulation it is possible to consider separating data which gets repeated for uniqueID's such as watchtime and trackingtime into separate files containing only the unique ID with its value. This can also be done for other metrics which might contain more global data such as the amount of people in frame which is currently not saved in the CSV even though it is detected on screen. The visualisation interface needs to be changed in order to load three different CSV files, but loading multiple DataFrames simultaneously in the visualisation interface has proven possible already so this should pose no problem.

Another point of interest which has not yet been realised because of time constraints is the calculation of total watchtime. This can indirectly (albeit not completely accurately) be derived from existing values. People generally turn the front of their face to wherever they are looking [52].This could mean that it is possible to predict how long a person has been looking at the screen by checking whether a frontal face is detected. For this code needs to be written which can determine based on how many times in a row a person was looking at the frame, whether a person was staring at the camera or simply turning their head, and then combining the subtraction of the first and last occurence of these epoch timestamps from each other.

[51] "What is dirty data? - Definition from WhatIs.com - SearchCRM." https://searchcrm.techtarget.com/definition/dirty-data. Accessed 3 Dec. 2018.
[52] "Combining Head Pose and Eye Location Information for ... - IEEE Xplore." http://ieeexplore.ieee.org/document/5959981/. Accessed 14 Dec. 2019.

12. Data visualisation

When the MVP is finished filtering the data, it should have clean data which we can call information. a framework needs to be built to allow users to interact with this data. This should consist of a back end which cleans and combines the data, and a front end which visualises it.

12.1. Framework

Because Infologic's FIDS is browser based and cross platform, it would be preferable to show the results of our experiments in the browser as well. This would keep the platform in a consistent cross-platform environment and allow for easy integration.

First we take a look at the different possibilities for displaying our data. We are working with a large sample size of people being analysed, and the results of individual passengers will most likely not contribute to our conclusions. We need to display the data in a way that filters these unimportant factors out, and gives the user freedom to correlate data however they want.

After experimenting with a few different frameworks such as Bokeh and Google Sheets/Graphs, the recently released Dash[53] framework was discovered. Dash is a framework used to create analytical applications for Python, and can be used for many different purposes. For this study it will primarily be used for information visualisation.

12.1.1. Dash

Dash offers a cross-platform browser based solution for data visualisation which will work on all platforms such as Windows, Mac, Linux and Android/IOS. Another big advantage is that most of the Infologic interface is already using a browser interface. This means Dash is compatible with the current Infologic framework, and interfaceable with Python data. Lastly Dash offers almost anything that we want to visualize, which means it is not necessary to combine different frameworks, libraries and API's in order to create the MVP. Dash also offers some unique functionalities to let users select data and dynamically change charts and graphs based on the selected input, which makes Dash an attractive framework.

Dash can visualise data from many different formats such as lists and dicts to make graphs, but it is also optimised to use Pandas DataFrames which provide more advanced filtering methods.

[53] "GitHub - plotly/dash: Analytical Apps for Python. Dash Is Productive™." https://github.com/plotly/dash. Accessed 22 Oct. 2018.

12.2. User Interface.

With the back end in place, it is time to begin visualising the information. The first step in this process is deciding what kind of information needs to be displayed. For this we turn to the field of data analysis.

12.2.1. Analytics

Data analysis focuses on processing input data and converting it into useful and readable information. This can be done manually by giving the user access to the many aspects of the gathered data, or by presenting the user with the proven most relevant and useful types of data. Because it is uncertain which of the factors gathered are important to improving the display visualisation, and there is insufficient time to perform live-testing, the framework needs to be flexible and expandable, with as little hard-coding as possible. Most of the methods used for visualisation have been, or can easily be reworked, to accept parameters from different DataFrame columns, based on what the user wants to visualize.

The following requirements have been set by the project manager for the MVP's user interface:

- The interface can show all the cleaned data.
- Provides comprehensible analytics such as charts and/or graphs.
- Can visualize relations between multiple types of data

12.2.2. Layout

To program the layout of Dash pages , a special type of syntax is used to write HTML inside python. Dash can convert this HTML code to a page layout[54], and fills the elements on the page with the data which is passed to these elements. For example

```
app.layout = html.Div([
   html.Div([
       html.H2('Machine learning interface',
               style={'display': 'inline',
```

along with some CSS in JSON format, results in the rendering of

Machine learning interface

in the Dash interface.

[54] "Dash HTML Components - Dash User Guide - Plotly." https://dash.plot.ly/dash-html-components. Accessed 17 Nov. 2018.

12.2.3. Datatable

The Dash DataTable provides a visualisation of a DataFrame. The concept behind this is comparable to a Mysql table. The DataFrame is the back-end of the DataTable, while the DataTable visualises this. Whenever a value in the DataFrame changes, so will the DataTable.

Filter Rows

trackingTime	gender	personID	▼age	avgEmotion	watchTime	timeStamp	beginTime
24	male	108	40	happy	watchTime	1547563406	2019-01-16T11:27
39	male	95	40	sad	watchTime	1547563424	2019-01-16T11:27
44	male	58	40	sad	watchTime	1547563370	2019-01-16T11:27
18	male	15	40	angry	watchTime	1547563293	2019-01-16T11:27
93	male	3	40	sad	watchTime	1547563296	2019-01-16T11:27
32		127	39	happy	watchTime	1547563457	2019-01-16T11:27
32		127	39	happy	watchTime	1547563457	2019-01-16T11:27
32		127	39	happy	watchTime	1547563456	2019-01-16T11:27
32	female	127	39	happy	watchTime	1547563455	2019-01-16T11:27
24	male	108	39	happy	watchTime	1547563406	2019-01-16T11:27
32	male	98	39	angry	watchTime	1547563424	2019-01-16T11:27

Example of a DataTable containing data from an experiment.

What makes the DataTable special is that like some other Dash elements it allows for custom filtration which can be enabled or disabled in the code. This provides a handy filtering tool on top of the datatable where custom filters can be entered. For example entering >=35 in the age field filters out all ages below 35. This allows the user to only examine specific information

Filter Rows

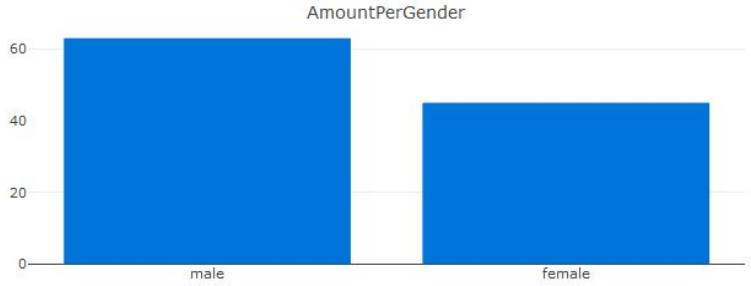
avgEmotion	watchTime	gender	timeStamp	age	beginTime	trackingTime	▲personID
Search	Search	Search	Search	>=35	Search	Search	Search
happy	watchTime		1547563294	37	2019-01-16T11:11:	93	3
sad	watchTime		1547563294	39	2019-01-16T11:11:	93	3
surprise	watchTime	male	1547563295	38	2019-01-16T11:11:	93	3
sad	watchTime	male	1547563296	40	2019-01-16T11:11:	93	3

Example of a filter on the DataTable

While dash allows for easy filtration, one problem which is currently present is that the datatable filtration does not always work as intended. This is because of the Pandas method datatables use for filtration. For example in a datatable between men and women, every woman would also be counted in men because the letters “man” are also present in “woman”. Because of this custom solutions are sometimes needed which filter the DataTable in a more hard-coded way.

12.2.4. Charts

As mentioned earlier Dash provides a lot of functionality to display charts. In order to feed information from the DataTable to these charts they need to read data from the DataFrame in the back end.



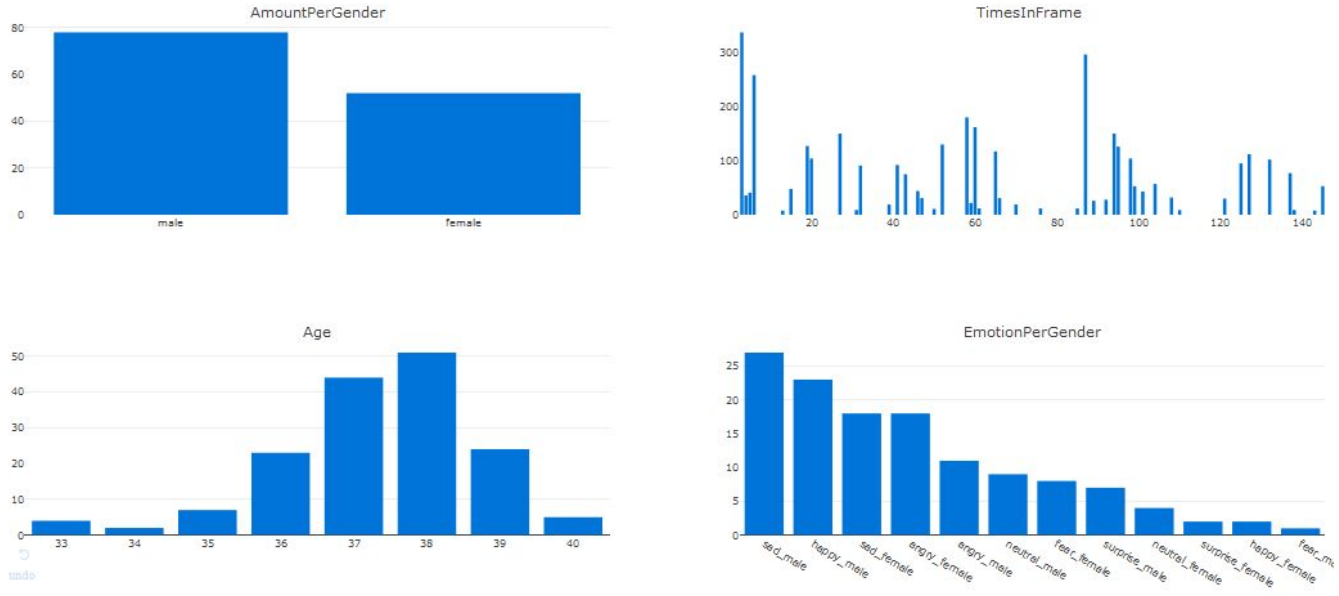
Example of a visualisation of the amount of occurences per gender

What truly makes Dash special is that elements such as the DataTable allow for interactive data visualisation. Charts and graphs can be linked to the DataFrame behind the DataTable, and whenever the DataTable applies a filter on the DataFrame, it is also applied to all the linked charts and graphs. Whenever a filter is applied, a customizable method called the "callback" is executed. All code in the callback is executed whenever a change is detected.

12.2.5. Multiple charts

While it is possible to stack charts beneath each other, this would take up a lot of space and require the user to do a lot of scrolling. For this purpose a special subplot[55] element is used.

Graphs



[55] "Python Subplots | Examples | Plotly." https://plot.ly/python/subplots/. Accessed 16 Jan. 2019.

12.2.6. Combined charts

A direct demand from the project manager was to visualise combined columns. Visualising basic information is very useful, but something which might give even more insight is finding correlations between multiple columns. For example to analyse which emotions were most frequent for each gender.

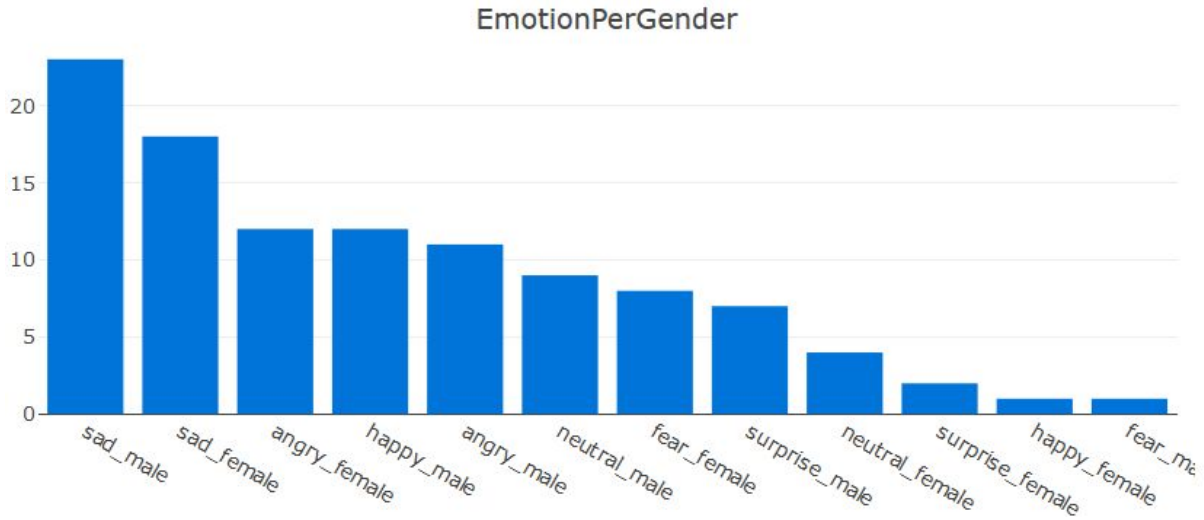
To do this a custom method is created which combines the values of multiple columns and creates a new column from them which is appended to the DataFrame.

```
def createCombinedChartValue(self, dataframe, col1, col2):
  return ((dataframe['%s' % col1] + '_' + dataframe['%s' %col2]).dropna())
```

This method can be used to put data into the charts tool to dynamically create custom charts. The reason this has to be done in a method is so that every time a filter is applied to the DataTable above, this method runs on the filtered version of the DataTable in the callback method, making the chart dynamic instead of static.

```
'x': methods.createCombinedChartValue(dff, 'avgEmotion','gender').value_counts().keys(),
'y': methods.createCombinedChartValue(dff, 'avgEmotion','gender').value_counts().values,
```

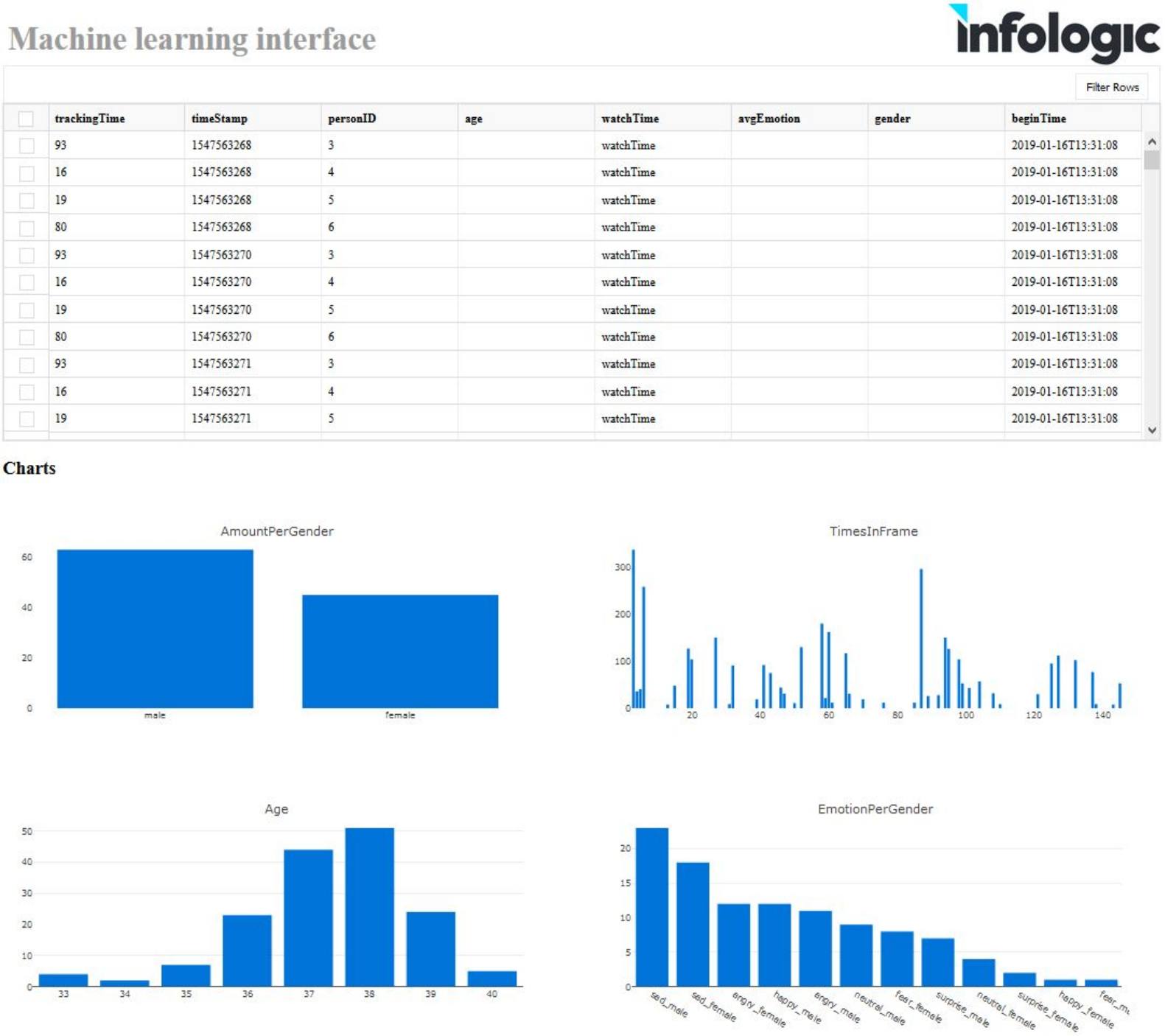
Resulting in the following chart:



Example of a combined chart showing the emotions per gender.

12.3. Interface results

The final interface MVP provides a user interface with many different analytics. These provide insight into the results of experiments performed with the feature recognition MVP. An example of this interface can be seen below.



Example of what the final user interface looks like. Rendered in Firefox 64.0.2

This interface is:

- Compliant with Infologic's existing browser based infrastructure.
- Shows all the cleaned data with the use of a DataFrame.
- Can apply customized filters to the DataFrame as well as its sub elements.
- Provides insight into the product.
- Can visualize relations between multiple types of data.

With all these requirements sufficed, it can be concluded this interface suffices for the MVP

12.4. Future prospects:

There are many development possibilities to expand the functionality of the front end interface.

For usability: improvement, the integration of the data collection framework should be done within the dash interface, so users can start and end experiments from this interface. For this a button[56] needs to be created with callback functionality to start the *new experiment* program in python. Another big functionality which would greatly benefit the interface is the visualisation of a list with all past experiments, and the functionality to allow a user to choose which experiment to visualize.

For analytics: one of the most impactful expansions is to add an input element to the combination graph which can be linked with the combine charts method, so that users are able to create custom combined charts within the layout. On top of that the addition of more advanced formulas for the back end, possibly using machine learning, could also be looked at if the data collection framework is expanded into something bigger. Another possibility is to take a look at the addition of graphs. While most easy information can be displayed with the use of charts, some information is better represented in the form of a graphs, which have not yet been implemented in the interface.

[56] "html.Button - Adding a click event - Dash - Plotly Forum." 25 Jul. 2017, https://community.plot.ly/t/html-button-adding-a-click-event/5073. Accessed 5Jan. 2019.

13. Conclusion

In this chapter a recap is made of the MVP developments in this study, and how the final MVP contrasts with the original requirements.

13.1. Results

Now that the MVP is finished it is time to take a look at how its capabilities fare in comparison to the requirements set at the start of the project.

- Provides data about the usage of display software by passengers

The MVP gathers multiple types of data on passengers interacting with displays such as their emotions and gender, how long they have stood in front of the display and more.

- Any heavy computing must happen off-site. An ethernet connection is available.

The MVP can analyse recorded video footage. Recording can be done with an on-site camera which needs to send this information to a server where the MVP will be running, however this functionality still needs to be coded. While this functionality is missing for now, there is no doubt about the possibility of implementing this functionality as many existing applications can already perform this task, meaning it is confirmed possible.

- Requires minimal upkeep

Besides manually starting and ending experiments, and manually starting the Dash interface, the MVP does not require any hands-on attention. The experiments can run for extended periods of time without crashing. The only current edge-condition is that the hard drive to which the CSV file is written provides enough storage space.

- Filters out dirty data (will be explained more in-depth in the chapter Data Science)

By running multiple algorithms the experiment data is cleaned and certain values are calculated from existing values. There are still some missing data columns which still need to be calculated.

- Is able to deliver the raw data so that it can be viewed with custom software.

The MVP provides raw and filtered CSV files of the experiment results. Because of the versatility of CSV files, they can be converted and viewed with virtually any other data science tool.

- Is able to display the data and give insight in a comprehensible way

A lightweight cross-platform interface compliant with Infologic's current display technology has been created. While graphs have yet to be included in this interface, it allows for custom filtering and the combination of different columns.

13.1.1. Conclusions

In conclusion it was possible to create an MVP that satisfies all the aforementioned requirements. In some cases it was possible to even exceed the requirements for the MVP, although a few options that we would have liked to introduce, but were not yet necessary for the MVP were beyond current capabilities of easily accessible technologies. An example of this is the gaze-tracker, which would have been very helpful if it provided accurate results to evaluate display heuristics.

14. Discussion

14.1. AI/Deep learning

For this research it has been decided to not yet dive into deep learning from the data, as Infologic cannot allow a custom algorithm to modify live FIDS layouts. This might result in extremely confusing layouts which are not comprehensible to passengers, or it might find unintended edge cases, such as the more data it removes from the display, the faster the flow of interaction, because people will quickly move on from an incomprehensible display that does not contain their information. There are too many risky edge cases which are impossible to prevent with a limited development time in order for the product to run in a mission critical environment.

15. Reflections

In this chapter we reflect on some of the subjects that have been studied in this thesis, and our general thoughts on these subjects beyond usability in this thesis.

16. General thoughts on machine learning

In some areas machine learning is still far less advanced than originally expected before starting this study. In computer years a relatively long time has passed since algorithms like age and gender detection have been introduced, yet there is still no solution which is even remotely close to being as accurate as they claim. Often algorithms are developed to perform well on a select few tests rather than being accurate in general use. Some are even trained on the dataset they need to evaluate which naturally results in extremely high scores. One of the biggest problems seems to be that almost every single algorithm uses deep-learning. This means that for every use-case which needs to be solved, a giant dataset needs to be gathered with examples the algorithm can learn from. Because there is a lot of interest in solving the field of face-recognition and object detection, many of the problems in those fields have already been solved. But the field of feature recognition still leaves much to be desired. When a new type of machine-learning technology is discovered, it is often applied to as many different fields as possible in hopes of providing

16.1. Reflections on gaze-tracking

A big regret was that we should have probably stopped researching the field of gaze-tracking sooner. There was hope that the uncalibrated gaze-trackers would provide accurate enough results for the use-case, but after diving deeper into the field and finding out gaze-tracking is reliant on a multitude of factors such as knowing the distance of the target to the screen, the distance between their eye sockets and their head rotation, it became clear that this is an incredibly difficult problem to solve.

A few papers and repositories claimed to have solved this problem of uncalibrated gaze-tracking, but the most advanced implementation of uncalibrated gaze tracking with the CSAIL dataset did not fare well when targets were at an unknown distance from the screen.

16.2. A reflection on feature recognition

In feature recognition fields such as emotion and age-detection, many developers are working with the same IMDB-dataset because it is one of the biggest datasets with labeled information which algorithms can be trained on. A big problem is that the majority of that dataset falls in a certain age and race category, which skews its bias to results within that demographic. To create more accurate recognition algorithm, a model should be trained on the combination of different datasets. Especially the age-prediction algorithm has too little accuracy to provide accurate results. Sadly there was not enough time to flesh out these algorithms because else the visualisation interface would not be finished in time.

16.3. Reflecting on Data filtering

Data manipulation was both one of the most interesting and boring parts of the study. There are so many different approaches to the same issue, which have widely varying efficiency. There is a wide assortment of different sorting features, all which have their own purpose. It was a big task to understand the different methods available, and Pandas is by no means comparable to a beginner-friendly querying framework such as MYSQL.

16.4. A look back at Dash

Dash is one of the most interesting frameworks to stumble across. It uses python to combine multiple different big open source frameworks such as PlotlyJS and Flask into a data visualisation framework which actually works surprisingly fast and accurately. With one of the most passionate communities I have ever seen, they create advanced interactive graphs I would have never thought possible. And all that inside of a browser. Like Pandas, Dash is not an easily programmable interface for beginners. Especially with the quirks such as the requirement to code HTML within Python instead of having default functions which generate these HTML structs from pure python code. But the amount of customisation that can be achieved makes Dash an incredibly potent solution, which could reach great heights in the future.

17. Special Thanks

I would like to thank the project manager Ernst Rijsdijk for his guidance on writing this thesis, as well as being able help with technical issues. It was a blessing to be guided by a person with as many technical as social qualities as him. I am also grateful for the entire team of Infologic for providing a pleasant work experience in a joyful environment, as well as their support in every aspect including their proprietary foosball bootcamp.

I would also like to thank the entire open-source community for providing development materials and aiding with technical questions.

Last but not least I would like to thank my friends and family for their support with this thesis, and life in general.

18. Sources:

1: https://www.schiphol.nl/nl/schiphol-group/pagina/traffic-review/
2: http://www.changiairport.com/corporate/about-us/traffic-statistics.html
3: http://www.startuplessonslearned.com/2009/04/validated-learning-about-customers.html
4: https://simplystatistics.org/2013/12/12/the-key-word-in-data-science-is-not-data-it-is-science/
5: https://hackernoon.com/the-ai-hierarchy-of-needs-18f111fcc007
6: https://dl.acm.org/citation.cfm?id=142179
7: https://www.exp-platform.com/Documents/2015%20Online%20Controlled%20Experiments_EncyclopediaOfMLDM.pdf
8: https://pdfs.semanticscholar.org/78e4/3d0b0cad62892b5c9eb17871fcff7cdbd01e.pdf
9: https://pubsonline.informs.org/doi/10.1287/isre.13.2.151.88
10: https://arxiv.org/abs/1506.02640
11: https://cloud.google.com/vision/
12: https://westus.dev.cognitive.microsoft.com/docs/services/5adf991815e1060e6355ad44
13: https://aws.amazon.com/rekognition/
14: https://www.netguru.com/blog/python-vs.-c-for-machine-learning-language-comparison
15: https://medium.com/swlh/pivot-patch-or-persevere-i-patched-the-lean-startup-206d63023b9c
16: http://www.pygaze.org/2015/06/pygaze-analyser/
17: https://arxiv.org/pdf/1803.09125
18: https://www.sciencedirect.com/science/article/pii/S0959438803001053
19: https://www.pygaze.org/2018/05/saliency-mapping-taylor-swift/
20: https://github.com/hyf015/egocentric-gaze-prediction
21: http://www.pygaze.org/
22: https://osdoc.cogsci.nl/3.2/download/
23: http://gazerecorder.christiaanboersma.com/gazepointer-beta/
24: https://github.com/oveddan/runwayml-gazecapture
25: http://gazecapture.csail.mit.edu/
26: https://github.com/CSAILVision/GazeCapture
27: http://www.ogama.net/
28: https://sourceforge.net/projects/gazerecorder/
29: https://webgazer.cs.brown.edu/
30: https://xlabsgaze.com/
31: http://www.cvlibs.net/datasets/kitti/eval_tracking.php
32: http://detrac-db.rit.albany.edu/Tracking
33: https://motchallenge.net/
34: https://colab.research.google.com/notebooks/gpu.ipynb
35: https://github.com/facebookresearch/Detectron
36: https://github.com/philipperemy/yolo-9000
37: https://github.com/chuanqi305/MobileNet-SSD
38: https://pjreddie.com/media/files/papers/YOLOv3.pdf
39: https://github.com/abewley/sort
40: https://github.com/mvondracek/VUT-FIT-POVa-2018-Pedestrian-Tracking
41: https://towardsdatascience.com/precision-vs-recall-386cf9f89488
42: https://github.com/dannyblueliu/YOLO-Face-detection
43: https://github.com/oarriaga/face_classification/blob/master/report.pdf
44: https://www.vision.ee.ethz.ch/publications/papers/articles/eth_biwi_01299.pdf
45: https://github.com/oarriaga/face_classification
46: https://github.com/yu4u/age-gender-estimation
47: http://chalearnlap.cvc.uab.es/dataset/18/description/
48: https://www.sciencedirect.com/science/article/pii/S0951832013000100
49: https://pandas.pydata.org/
50: https://www.tutorialspoint.com/python_pandas
51: https://searchcrm.techtarget.com/definition/dirty-data
52: http://ieeexplore.ieee.org/document/5959981/
53: https://github.com/plotly/dash
54: https://dash.plot.ly/dash-html-components
55: https://plot.ly/python/subplots/
56: https://community.plot.ly/t/html-button-adding-a-click-event/5073

19. Glossary:

FIDS: Flight Information and Display System.
MOT: Multi Object Tracking
CNN: Convoluted Neural Network
CSV: Comma Separated Value

20. Attachments

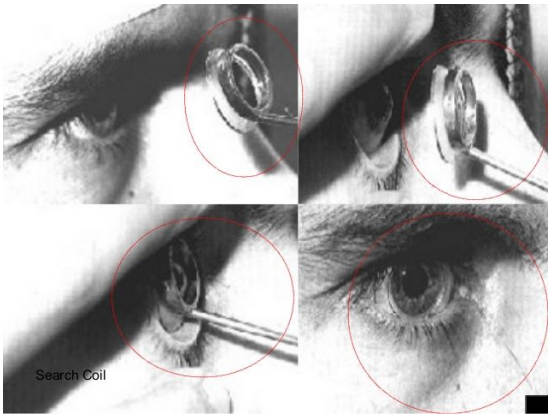
Attachment 1: machine vision application concepts presentation

Attachment 2: machine vision concepts and a machine hearing concept

Eye-attached tracking[57]:
The first accurately implemented method of gaze-tracking. Eye attached tracking uses special hardware attached to the eye to determine its position and the direction in which it is looking. Usually this is a special contact lens with an embedded mirror, or a small magnetic coil[58]. A special sensor for the technology it is based on can track the location of this attachment.

Pros:
- Extremely accurate tracking

Cons:
- Hard to set up
- Requires special hardware
- Very intrusive



Sources: https://ubicomplab.cs.washington.edu/publications/eyecontact & http://www.cs.cmu.edu/~ltrutoiu/pdfs/ISWC_2016_trutoiu.pdf

[57] "Comparing the accuracy of video-oculography and ... - Science Direct." https://www.sciencedirect.com/science/article/pii/S0385814604001774. Accessed 26 Aug. 2018.
[58] "A Method of Measuring Eye Movemnent Using a Scieral Search Coil ...." https://ieeexplore.ieee.org/document/4322822/. Accessed 26 Aug. 2018.

20.0.0.2. Electric Potential measurement

Electric potential measurement is done by attaching electrodes around the eye socket. By measuring the difference in resistance between certain points, the cornea can be tracked. This method is accurate in measuring saccadic eye movement (movement between focus points) and detecting blinking. It doesn't require any lighting, thus making it a great method for tracking eye movement, even while the eyelids are closed.

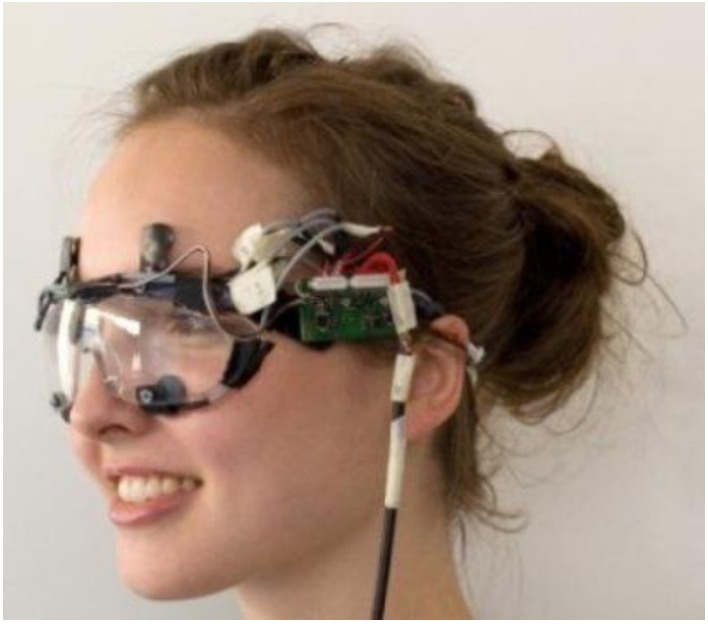
The downside of this method is that it is less accurate in finding exact fixation points and minimal movement. This is because while detecting movement is accurate, when faced with low variations in electric conduct, it becomes hard to differentiate whether an eye moved or not.

Pros:

- Inexpensive special hardware
- Simple to operate

Cons:

- Hard to set up
- Frequent need for recalibration
- Requires specialized hardware



Source:[59]

[59] "Wearable Computing: Special Goggles Analyze Eye ... - ScienceDaily." 28 Apr. 2008, https://www.sciencedaily.com/releases/2008/04/080428083418.htm. Accessed 1 Aug. 2018.

20.0.0.3. Video-oculography (head-mounted)

Gaze tracking is often confused with eye-tracking, however there is a big difference. Eye tracking focuses solely on the position of the eyes without taking other variables into the equation, because video-oculography is very commonly used in the lab-based eye tracking, Often this is done by attaching a static device close to the sides of their eye pupil. By attaching a camera which looks from the center of the nose it is possible to combine the eye position with the recording camera to see where the user is looking in that camera image.

Pros:

- Accurate results
- Allows for free movement
- Usable in non-controlled environments

Cons:

- Expensive
- Requires specialized hardware

Figure 3: a video-oculography setup

Source: [60]

[60] "Haslwanter: Video-Oculography - Thomas Haslwanter." 24 Jun. 2012, http://work.thaslwanter.at/Projects/Proj_VOG.htm. Accessed 13 Nov. 2019.

Pupil-corneal reflection[61]:
As the name suggests, this method uses a special infrared camera to track the position of the cornea. The camera shoots out an infrared laser which is reflected back from the cornea. With this the position of the eyeball can be determined

Pros
- Easy to implement
- Non-intrusive
- Doesn't require much setup.

Cons
- Requires special hardware (especially an IR camera)
- Non-moveable
- Restricted angle of movement allowed for the face.

SOURCE LINKEN

Machine vision tracking
Pros
- No proprietary hardware required.
- Can be combined with other types of machine vision when running in frameworks such as OpenCV
- Cheap

Cons
- Difficult to implement well
- Mediocre accuracy
- Often requires calibration

[61] "Iris center corneal reflection method for gaze tracking using visible light.." https://www.ncbi.nlm.nih.gov/pubmed/20952326. Accessed 23 Nov. 2018.