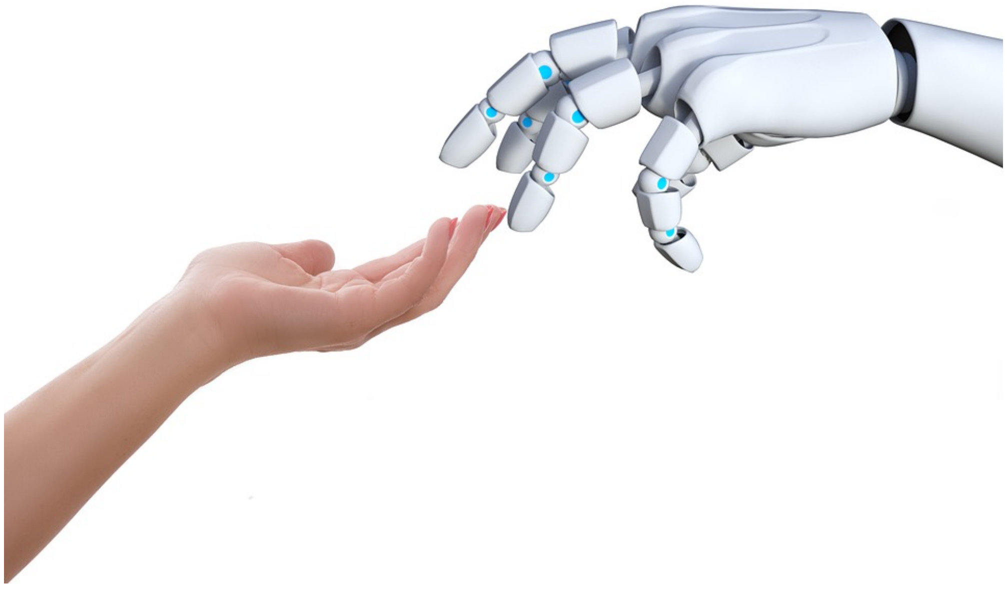
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Human-Machine Interaction, Communication Initiation Probability Estimation

Communication initiation using Computer Vision, Machine Learning and Artificial Intelligence

Master's thesis in Interaction Design

CARL-HENRIK HULT, JOAKIM SCHMIDT

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Master's thesis 2018:NN

Human-Machine Interaction, Communication Initiation Probability Estimation

Communication initiation using Computer Vision, Machine Learning and Artificial Intelligence

CARL-HENRIK HULT, JOAKIM SCHMIDT

Department of Computer Science and Engineering
Division of Interaction Design
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

Human-Machine Interaction, Communication Initiation Probability Estimation
Communication initiation using Computer Vision, Machine Learning and Artificial Intelligence
CARL-HENRIK HULT, JOAKIM SCHMIDT

Supervisor: Olof Torgersson, Department of Computer Science and Engineering
Examiner: Staffan Björk, Department of Computer Science and Engineering

Master's Thesis 2018
Department of Computer Science and Engineering
Division of Interaction Design
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2018

Human-Machine Interaction, Communication Initiation Probability Estimation
Communication initiation using Computer Vision, Machine Learning and Artificial Intelligence
CARL-HENRIK HULT, JOAKIM SCHMIDT
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Robots and digital assistants today typically require the use of key words or phrases to activate them. In this paper we propose and implement a prototype for initiating conversations with robots in a more natural way. Measurements of three key indicators of conversation interest – eye contact, head pose and distance – are captured from a video stream and, using various techniques such as convolutional neural networks and 2d/3d projections, analyzed to estimate who, if any, is most interested in engaging in conversation. Through user tests, we gather and compile data to evaluate the validity of the approach, and the accuracy and performance of the implementation. It is our conclusion that this is a viable approach, with the potential to feel very natural to the users, although the prototype has a number of problems with regards to accuracy and performance.

Keywords: human-robot interaction, eye gaze, communication, initiation, probability, estimation, machine learning.

Acknowledgements

First of all, we want to thank our academic supervisor Prof. Olof Torgersson, both for his feedback on our test setup, as well as his help in keeping the project on track.

Secondly, we would like to express our gratitude to our supervisor Erik Sjödin from Ericsson for his never-ending enthusiasm and encouragement.

Thirdly, we wish to thank Åke Johansson from Ericsson for helping us with our machine learning endeavours.

Finally, we thank Ericsson for giving us the opportunity to do this project and providing all the space and hardware we needed.

Carl-Henrik Hult and Joakim Schmidt, Gothenburg, June 2018

Acronyms

AI Artificial Intelligence. 1, 8, 10
ANN Artificial Neural Network. 7, 8, 12, 18

CIPE Conversation Initiation Probability Estimation. VIII, IX, 40, 41, 54, 55, 70
CNN Convolutional Neural Network. 33, 62
CV Computer Vision. 10

EGE Eye Gaze Estimation. 11, 16, 18, 22, 29, 32, 34, 36, 40, 41, 45, 47, 49, 50, 54, 55, 66–70

HOG Histograms of Oriented Gradients. 33, 36, 64, 70
HPE Head Pose Estimation. 42, 55, 66, 70
HRI Human-Robot Interaction. 1, 4–6, 10

ML Machine Learning. 7, 10, 12

PIS Person Identifier System. 32, 33, 36, 45, 72
PRS Person Registry System. 32, 36, 40, 45

SVM Support Vector Machine. 12

Contents

1
Introduction

1.1 Purpose

Robots and artificial intelligence are two continuously growing fields of research, whose results affect ever more of our lives, in all levels of society. Historically, robots have been highly specialized for specific jobs, and used to replace humans at monotonous jobs, completing them faster, cheaper and more reliably. As computer hardware has improved however, the robots' abilities to complete complex tasks has also improved, and today there are a multitude of different robots, capable of moving at high speeds while avoiding collisions with objects in their path[1], playing the violin[2], or even perform surgery[3]. While these robots are capable of amazing feats, other areas of robotics and Artificial Intelligence (AI) may not seem so impressive, at least at first glance. One such area is Human-Robot Interaction (HRI) which is typically awkward for the human user, requiring the use of specific commands, language and behaviour that feels unnatural. This project aims to take a small part of this large problem space, specifically the very beginning of the interaction, and explore a possible solution to making it feel more natural for the human user.

1.2 Research question

When robots become a part of our everyday lives, their ability to integrate themselves and offer meaningful interaction possibilities become ever more important. To further their perceived usefulness to future would-be consumers and/or users, any interaction with the robot must be smooth and natural, and, as the first impression is usually very important, the initiation of said interactions must also be smooth and natural. Therefore, the question that this work strives to answer is this:

What are some important human behaviour patterns that can be detected by a humanoid robot, and used for initiating a conversation between said robot and a willing human in a, to the human, natural way?

Since this is a very open-ended question, and this work will strive for a working prototype implementation (see section 1.3 for the aims of this work), the following sub-questions naturally arise:

1. What software technologies are useful in solving the individual problems of detecting these patterns in real-time?

2. What hardware is needed to provide the necessary data in real-time?

1.3 Aim

The aim of this work is to make the interaction with robots more natural for humans, specifically the very start of an interaction. Many robots and artificial intelligence programs today require keywords or specific gestures from the user in order to initialize a conversation, such as Apple's Siri, or Microsoft's Cortana. This forces the human to conform to the robot's special needs, even though robots are typically meant to be tools for humans. Social psychology reveals that humans typically use a multitude of subtle tells to communicate interest in starting an interaction with each other. While it may not be feasible to replicate such a complex behaviour in robots (to communicate to a human the robot's interest in starting a conversation), the information communicated from human to robot ought to be possible to interpret and use in order to allow the robot to recognize when a human is interested in engaging in conversation. The goal is to develop a platform for the robot on which human body language interpretation modules can be deployed, and to develop at least one such module. The idea is that the platform can be easily extended with more modules, each interpreting one human social cue, and that the modules will collaborate to determine the likelihood of a nearby human's willingness or intention to engage in conversation with the robot. According to Nummenmaa and Calder[4], eye gaze direction, or eye contact, is among the most important cues in determining a persons object of interest. Therefore, the first module developed will be focused on detecting eye contact with the robot, and will do so using a machine learning approach. Further modules may be developed if time allows, in order to test the validity of the platform architecture.

1.4 Delimitations

Due to the complexity of this project, and the, in the context of a semester's worth of work time, limitless possibilities of extensions, the focus during this work will be on implementing a working prototype. As this project is mostly about developing a software solution, there will not be a lot of focus on the hardware side of the project. The aim is also not to actually deploy the solution to existing robot hardware, to free up more time working on the software. As such, a low-fidelity prototype, consisting of a micro-computer such as the Raspberry Pi and a camera, will be set up early on, and used as a stand-in for the robot during most of the project. Also, no major user studies will be conducted, though small user tests will be done as part of the evaluation.

Machine learning algorithms typically use a lot of data to develop a working model of a given problem. Such data, in the context of computer vision and neural nets, typically consists of thousands, or even millions, of images manually labeled according to what they show. While gathering data like this is a fundamental part of machine learning, such an endeavour is beyond the scope of this work, as it would likely consume a majority of the time available. Instead, already available data (of which there is quite a lot) will be used and manually labeled where appropriate, and different solutions will be explored where machine learning is impossible due to this limitation.

It is the intent that the platform developed will provide the robot with conversation interest estimations for each person the robot sees. These estimations will be as a percentage between 0-100. However, how this data is then interpreted, and where various thresholds – such as what is enough for the robot to say “hello” – lie will not be defined by this project.

1.5 Ethical issues

To construct the systems that underlie the robot’s intended behaviour in initiating conversation, several different techniques and software solutions will be used. Many of those solutions share the weakness that their performance is heavily dependent on the initial data set used to train them. I.e. they will only ever get good at, for example, recognizing people that share traits with the people whose pictures were used to construct the program in the first place. As such, it is a very real possibility to subconsciously exclude certain groups and minorities from the data set, leading to the finished prototype being able to identify some individuals as humans, but not others. Such groups may be formed based on appearance or disabilities or some other subtle trait that they share, or don’t share with others.

Using peoples’ pictures in any research project, it is important to be wary of privacy issues. The images in the data sets must be handled with care and chosen carefully. All images must be of people who agreed to being photographed, and any person testing the prototype will be well informed of what data is gathered, and how that data is used before agreeing to help us with testing.

2
Background

2.1 Brief history of robotics

The idea of creating life in a artificial way has been around for a very long time. There is an example from ancient Chinese legends about a craftsman called Yanshi[5], who created an artificial human that acted and looked like a human. When he presented this to his king he had to cut it open to prove that it in fact was not a real person but a shell with organs made up of leather, wood and glue.

Another more recent example would be the mechanical man that Leonardo Da Vinci sketched in 1495[6] (see Fig. 2.1). Both of these are examples of humanoid robots that imitate humans both in appearance and in function. This is, however, only one part in the field of robotics as the difference in robot function and looks is as varied as peoples imagination. Today, robots are filling a growing number of roles in society that uses different aesthetics and techniques often tailored to the task at hand. Factory automation, service application and space exploration are only a few of the roles that robots have been used for. NASA has for example placed robots on Mars to act as geologists, examining the planet by doing different scientific measurements using its many instruments mounted on the robot.

Depending on their intended application, robots can have very different appearances and behaviours. Assembly line robots, such as the ABB IRB-series[7], are designed with very specific tasks in mind, and their appearance reflect that, looking far from human. Other robots are designed to mimic animal behaviours or mannerisms, such as the Dragonfly robot[8]. These types of robots may be capable of more general tasks, such as traversing different types of terrain or carrying various pieces of equipment. Still others are built to look and act human, doing tasks that have historically been reserved for humans, such as carrying conversations and caring for the elderly[9]. Interacting with humans in a human way, it's in the robot's best interest if they appear and act human, to leverage peoples tendencies of anthropomorphism, in order to build trust with their user. Most people would probably not allow themselves to be cared for by a typical assembly line robot, but might do so if the robot looks human. In this thesis, the word 'robot' will be used to primarily refer to the last type, the human-looking machine with eyes and ears, legs and arms, since these types of robots are the intended target of the work described herein.

The different applications for robots demands different types of interaction with them as well. This is what the field of HRI is tackling and is also closely connected to what will be the focus of this thesis. How do humans interact with robots and what can you do to make this interaction better?

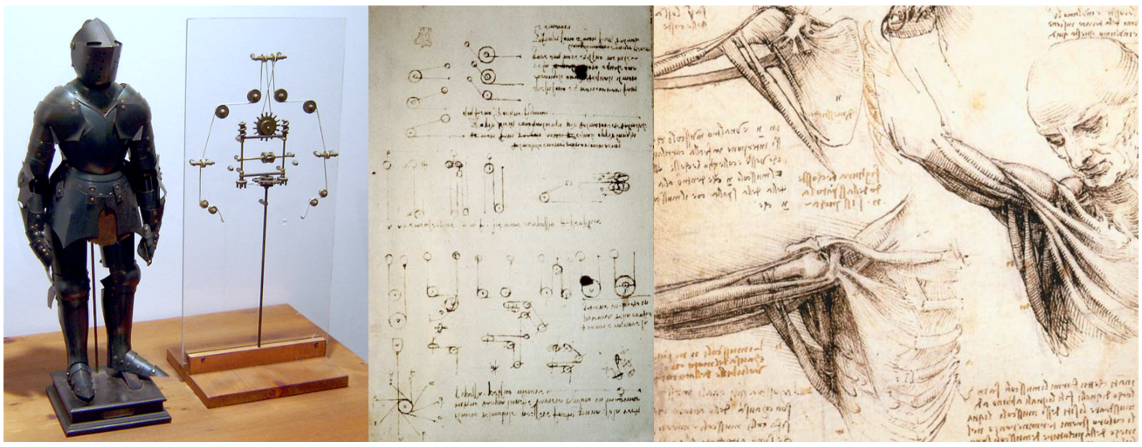


Figure 2.1: Leonardo's mechanical man recreation displayed on the left with Leonardo's actual sketches on the right.

2.2 HRI

HRI is described as a multi-disciplinary field of research[10] that emerged in the mid 1990s and its approach was to involve people from many different fields; researchers from cognitive science, psychology, natural language, human–computer interaction and robotics gathered at events to exchange ideas on the topic. The IEEE International Symposium on Robot & Human Interactive Communication (RoMan) was first organized in 1992 and has since then been recurring annually. Since then, many more of these multi-disciplinary conferences and meetups have been organized focused around HRI, one example being the IEEE/Robotics Society of Japan International Conference on Intelligent Robot and Systems.

2.2.1 HRI in literature

In Ĉapeks play "Rossum's Universal Robots" in 1921[11] he coined the word "robot" and also described a scenario of humans interacting with robots that later rebelled and tried to destroy the human race. This theme of robots rebelling against humans has been a popular theme in pop culture since, and one science-fiction writer in particular set out to actually detailing human robot interaction. Isaac Asimov coined the term "robotics" and laid out groundwork for HRI in the 1950s in his now famous *Robot series* perhaps most prominently in the book *I, Robot*[12] where he laid out what is now known as the Asimov laws of robotics:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey orders given it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

This was an early attempt to describe how robots should interact with humans by detailing what they can and can not do. Not all works in popular culture has an impact on the field of robotics, but there are instances where they do, and Asimov's laws is one example. Even though no robot has been created that is capable of the type of autonomous reasoning that is needed to obey those laws, it has forced a debate among researchers[13] in the area about how to handle it if that technology is created.

2.2.2 HRI – this thesis

Goodrich and Schultz[10] categorizes human-computer interaction into the following:

- Remote interaction — The human and the robot are not colocated and are separated spatially or even temporally (for example, the Mars Rovers are separated from earth both in space and time).
- Proximate interaction — The humans and the robots are colocated (for example, service robots may be in the same room as humans).

In this thesis the focus will be the proximate type of interaction that also humans engage in. Human-human interaction could also be divided into these two categories, as an email to a person would classify as remote interaction whilst waving at them would be of the proximate type. As the interaction between the human and the robot in this thesis is supposed to in some way mimic how humans communicate in the proximate sense using computer vision, the remote applications can be discarded for the purpose of this project. Similarly to the approach of the whole field of HRI, this project will focus on understanding research done in cognitive psychology, neuroscience and social science, in order to, on a deeper level, understand human behavioural patterns, before looking at the robotics side of the issue. The mixing of ideas and knowledge between fields aids in fully understanding the challenges regarding the interaction between humans and robots.

Robots that mimic humans is an area of research that has been explored by many researchers. Turing detailed a test[14] of artificial intelligence. This was described by Feil-Seifer and Matarić[15] as follows: "Alan Turing proposed a test of artificial intelligence (AI), whereby a system is evaluated by whether it could fool a human user communicating with it through teletype[14]. This test was later elaborated to the Total Turing Test[16], where a system communicating in human-like ways (text, speech, facial expressions) tries to fool a human user into believing it is human. Since that time, one of the benchmarks for success in AI and HRI has been how well the system can imitate human behavior." To succeed at the Total Turing test, the conversation initiation must surely feel natural as well. Though it is beyond the scope of this project to try and fully succeed at even such a small part of the Total Turing test, it is nevertheless a very interesting goal at which to aim.

3
Theory

3.1 Machine intelligence

The field of machine intelligence is very broad in its use of technologies and concepts. It typically incorporates all technology that make machines, be they robots or software programs on a mobile phone, look smart or intelligent to the user. Even so, a number of sub-fields exist and are usually able to be kept separate.

3.1.1 Machine learning

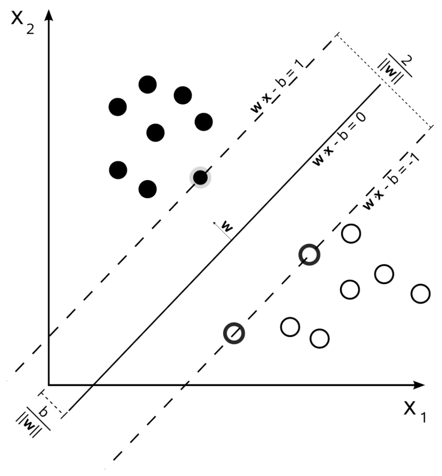
In Machine Learning (ML), a multitude of technologies are used to train computers to do specific tasks, such as identifying hand-drawn digits, predict fluctuations in stock values, or identify trash on a beach from aerial photos. Several different techniques exists within this field that each have their pros and cons. What they all have in common is that they require some preexisting data to train on, in order to learn what they are supposed to do, and that the resulting algorithms and equations are typically very difficult, if not impossible, to decipher after the training is complete.

3.1.1.1 Support vector machines

Classification is important in the field of ML and a way of classifying new example data is to use a support vector machine classifier. This aims to find a line in a labeled training set that with the largest margin to the closest data point separates the two classes as seen in fig. 3.3 You can then use that line as the separation line, classifying new examples according to on what side of the line they fall. The line with the largest margin to the closest data point on both sides of the line is called the optimal hyperplane[17].

3.1.1.2 Artificial neural networks

One of the most complex areas of ML, Artificial Neural Networks (ANNs) is loosely modeled after the human brain to replicate its network of neurons as software[18][19]. The human brain is capable of amazing achievements but the primary building block, the neuron, is a relatively simple structure, with very little power of its own. Instead, the astounding capabilities of the brain comes from the enormous collection of neurons and neural connections. By emulating the core functionality of the neuron – one or more inputs results in one output through some internal calculations and weighting – and neural connections – the outputs from one neuron



X_2
X_1
$\vec{w}\cdot\vec{x}-b=1$
$\vec{w}\cdot\vec{x}-b=0$
$\vec{w}\cdot\vec{x}-b=-1$
$\frac{2}{\|\vec{w}\|}$
$\frac{b}{\|\vec{w}\|}$
$\vec{w}$

Figure 3.1: Support vector machine classifier separating two classes with the largest margin

is one of the inputs to another neuron – ANNs (see fig. 3.2) become capable of understanding complex patterns that humans subconsciously understand, but that are too complex to be programmed into a computer by hand. Facial recognition or handwriting comprehension are two such areas, where the huge but subtle variance of the data makes it impossible to program all the rules manually.

ANNs require large amounts of data to train and test[18]. A large part of the work involved in their use is the gathering and labelling of this data. The data and its labels are various examples of what is and isn't an instance of whatever the ANN is supposed to learn to identify, and typically range from a couple of thousand examples, to several million. As an example, the MNIST database[20] contains 60,000 images of handwritten digits, that can be used to train a neural network to distinguish between the ten digits (0-9) with roughly 99.2 percent accuracy[21]. While it can be difficult and time-consuming to find and label such an amount of data, or even more in some cases, the variance in the data is what makes ANNs so powerful.

3.1.1.3 Machine learning frameworks

To create neural networks there are several different frameworks to use, for example PyTorch[22], Caffe[23] or TensorFlow[24] to name a few. For this project TensorFlow was chosen from these due to close access to competence with this framework through our supervisor. There are also many well documented examples of usage on the website.

Another tool used was dlib[25] which is a library containing many tools for machine learning and computer vision. This was used due to its many working examples available online, as well as its ease of use.

3.1.2 Artificial intelligence

AI has no one agreed upon definition. Instead, it is commonly used to describe any seemingly smart machine or program, with no regards to the underlying technology.

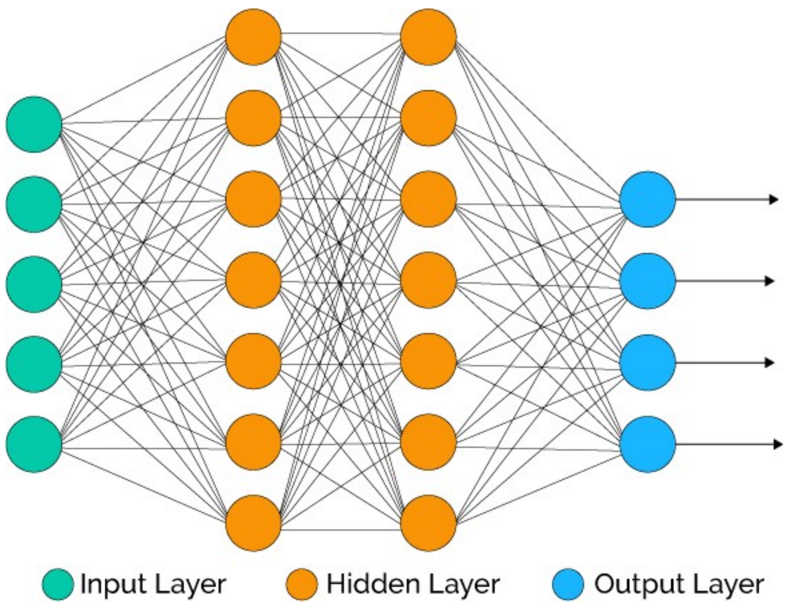


Figure 3.2: Typical rendering of a Convolutional Neural Network, a subtype of Articial Neural Networks that differs in that they have multiple hidden layers.

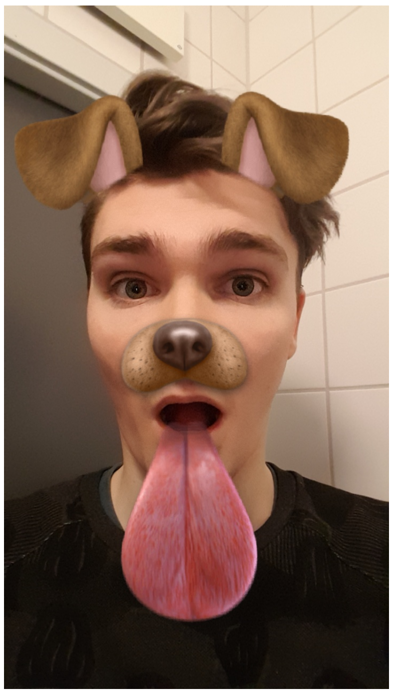


Figure 3.3: Filters applied to facial features to make you into a dog

It is common to divide AI into two groups: strong and weak AI, though this is not always the case either. Strong AI refers to that which is commonly depicted in pop culture; sentient or self-aware machines that can think creatively and are more or less indistinguishable from humans except for their significantly higher processing speed (e.g. the Terminator or HAL9000). Weak AI on the other hand, typically means that which we have in our society today which, while it may seem clever or smart, usually consists of a very limited set of functions with a, to humans, natural interface, such as computer-steered opponents in most games[26]. For this thesis, AI will be used to refer specifically to typically weak AI that does not use ML, so as to limit the terms scope, making it more useful when describing specific things. Any perceived intelligence in the AI system is designed by a programmer to work that way, not generated by any sort of training.

3.1.3 Computer vision

Computer Vision (CV) is the open-ended issue of having computers interpret the real world around them through vision (using cameras usually), much like humans do[27]. This includes problems such as identifying and classifying objects, people and animals in images or videos. This acquiring of high-dimensional data of images from the real world, and processing to be able to make decisions from it, is the cornerstone of what is called CV. It is used in many fields of both research and also entertainment. The popular smartphone app SnapChat uses computer vision to find different facial features of a person in order to add layers and filters on proper locations in the image(see Fig. 3.3). So using CV they are able to turn your eyes into those of a cat in the app, as they identified where your eyes are in the picture. The actual processing of the image can be done in several ways using for example ML or AI.

3.2 Related studies

3.2.1 Human-robot interaction

Human-robot interaction is a large field with a large number of diverse problems, encompassing many different fields of research such as social psychology and interaction design. A lot of research has been done with regards to the connection-forming behaviour of humans, in attempts to translate them to the area of HRI. For example, Chao et al.[28] analyzed human interactions to establish a model for a robot to initiate conversations with humans in such a way as to feel natural to the human. They modeled a scenario in which a robot, acting as an assistant in a store, was tasked to determine when and how to suitably approach customers. Much focus was put on the spatial relationship between two participants in a conversation and the knowledge of how that spatial relationship affects the initiation of the conversation. Similarly, and in a comparable situation, Satake et al.[29], deployed a robot to a shopping mall with the intent of determining how the robot can approach people and offer help or advertising services. Calculating where people are walking, how to intercept them and start a conversation was a major part of the focus of their study and, while not entirely related to this project, their use of computer vision to recognize people in motion might prove useful.

Rich et al.[30] made a study on the connection process between two humans, where four behavioural parts were identified:

- Directed gaze – looking at an object with the intent of having the other person also look at that object
- Mutual facial gaze – what is typically known as eye contact (though it includes looking at other facial features such as the mouth), when both persons look at each other
- Adjacency pair – pieces of conversation where one person's statement is followed by an immediate answer from the other (such as questions and answers)
- Backchannel – includes all the subtle movements and sounds people make, like nods and "mm" sounds, that indicate their interest in, and attention to, the conversation

The study was done with the intention of transplanting some or all of these behaviours to a robot, to make the human-robot interaction better. The study also recorded and presents some data showing how long those connections take to establish, both for human-human and human-robot interaction. The concept of turn-taking in an interaction is also touched upon, which may be relevant for the initial connection establishment process, since who takes the first turn may depend heavily on who was the primary initiator.

The rotation or pose of someone's head is also indicative of their interest in starting or maintaining an interaction. It is often used in combination with Eye Gaze Estimation (EGE), as the two are heavily correlated but use different facial features for their evaluation. Asteriadis et al.[31] attempted to estimate the relative importance of head rotation and eye gaze direction by having test subjects annotate images of people's faces with regards to how attentive towards the camera these faces appeared to be. Their findings indicate that head rotation and eye gaze estimation, when used together, give significantly better agreement among test subjects than when either one is used on its own.

Sorokowska et al.[32] analyzed the average distance two people keep to each other when conversing, showing a rather large variance between countries. The relationship between the participants was also shown to influence the distance significantly. While their result is not directly applicable to this project, the maximum distance recorded for social encounters with strangers (about 140cm) may be of use to declare a maximum distance to which the robot will attempt to initiate or engage in a conversation. This is useful because a majority of the people that the robot may see will be well out of this range and can therefore be quickly discarded as non-interesting, resulting in less processing power being wasted. It may be that the distance intervals recorded can also be used to help determine the overall conversation initiation probability, though this must be tested.

3.2.2 Eye gaze estimation

Indicating interest in starting or maintaining a conversation by attempting to establish eye contact is very common in humans. Robots could imitate this behaviour, or analyze the gaze of humans around it, to initiate conversations. Eye-gaze estimation (EGE) is the concept of determining where someone is looking, which can be implemented through several different approaches. Strupczewski[33] describes two different categories for the possible approaches and their previous use:

1. Appearance based approaches are those that, using different forms of ML such as neural networks or linear regression, establishes a mapping between the point of the gaze and the image of the eyes
2. Model based approaches involves constructing and using 3D models of the eye, with known geometric properties, to estimate eye gaze direction

Strupczewski's work is focused on finding the gaze point (where the user is looking) on a screen in the best way possible. In contrast, this thesis will focus on the rather more binary question of whether eye contact has been established, and what importance, or weight, this has for conversation initiation. Appearance based approaches have been tried in a number of different ways previously and while some of the proposed solutions are not usable in real-time (which is important for this work) or outside of very specific settings, they may still contain relevant insights. Baluja and Pomerleau[34] tested several different ANN architectures to create a non-intrusive (i.e. without equipment that has to be installed or mounted on the subject) eye gaze estimator, with an accuracy of 1.5 degrees, as early as 1994. In contrast to their use of ANNs, Zhang et al.[35] made their own eye gaze estimator using a weighted Support Vector Machine (SVM) classifier on a dataset containing about 5000 images, binarily labelled as either "eye contact" or "no eye contact". Their results significantly indicates the validity of such an approach, reporting Matthews Correlation Coefficient[36] values that are well above chance.

4
Methods

4.1 Design process

Our process will adhere to four basic activities of interaction design[37] (see fig. 4.1).

4.1.1 Requirements gathering

The goal of the project, initiating conversations between robots and humans, may be easily stated, but its requirements are very loose and abstract. What do humans typically do when initiating a conversation? Eye contact, body language, distance and head posing are a few examples of subconscious behaviours that people use to communicate interest in a conversation with another, but there may be (and probably are) several others, that are not so apparent. What role does each of those known and unknown factors play in the other persons evaluation process? Which are more important, and how does it work on a mechanical level? All these questions need answering to build a set of requirements for achieving the project's aim. Literature studies will be the main method used to find answers to these and further questions, since much of it has been studied and published by psychologists and cognitive scientists. Neuroscience, the study of the nervous system, might also provide some insight towards what happens on a biological level when people interact. Further, the human-robot interaction domain has done many studies with related goals, and their requirements may be of help in constructing our own. The stakeholders

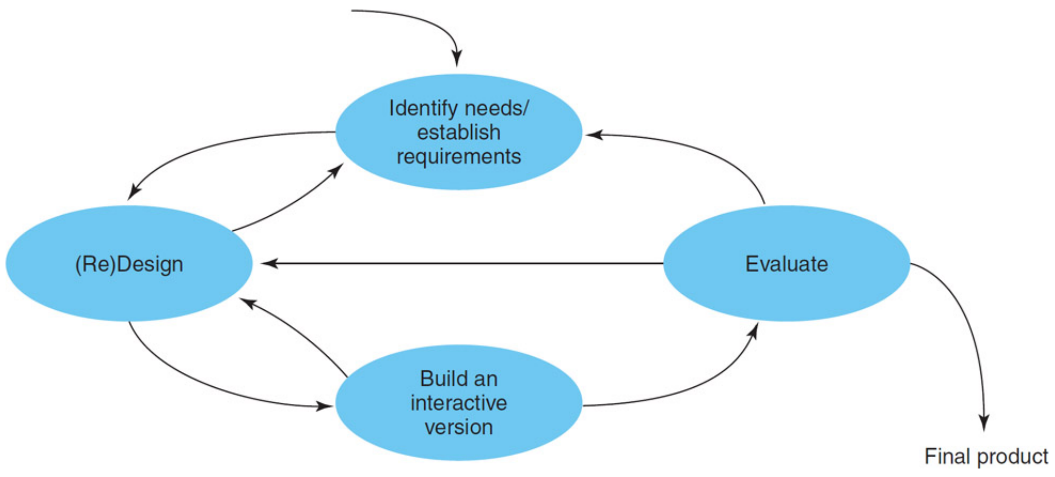

Identify needs/ establish requirements
(Re)Design
Evaluate
Build an interactive version
Final product

Figure 4.1: Model of the design process

may also have requirements on the architecture of the prototype, which will have to be taken into consideration during the development process. Most of this work will be completed during the first four weeks of the project, and the requirements established (though not set in stone), in order to brainstorm alternative designs.

4.1.2 Alternative designs

Two types of designs will be developed during this stage of the project, both being developed without input from users. Firstly some conceptual designs are devised, that aims to describe the context and the idea without focusing on details. This will consist of flow charts, diagrams and other types of both visual and textual descriptions of how the software architecture could help us to reach the set requirements. The flowcharts shows the flow of information in the program, and can be used to explain the role of the different parts to the stakeholders. Another part of the conceptual design is scenarios describing both typical user interactions with the prototype as well as edge cases. This helps both in the evaluation stage, in that you have an actual scenario to test out, but also in the conceptualization by thinking of eventual problems with edge cases early and think of that when designing the program.

Normally, the second step would be to create physical prototypes which can be used to determine details of look and feel[37]. However, since this project aims to create something which will actually never be seen by the user, it makes little sense to do this step on its own. Instead, it will be merged with the highly iterative prototype development process described in section 4.1.3.

4.1.3 Prototype development

The development of the prototype will be using some key agile practices throughout. We will be in constant contact with the stakeholders and letting them have insight into our progress and will be taking their feedback and new requirements into the process. This is a big part of being agile[38]. The process of training a neural network is iterative by nature, a key focus of agile development. As the program evolves there will be many short iterations both improving the algorithms and tweaking the training data. The evaluation in between these tweaks along with also iterating back and changing things if data from evaluating the product with users says so is very important.

4.1.3.1 Pair programming

The concept of pair programming is an agile practice that was one of the prominent parts of Extreme Programming[38]. The idea was originally that all programs should be written by two people sitting at one machine, one operating the keyboard and the other checking all the code. This type of extreme case is not how it will work during this thesis. We will, however, throughout the project, work very close to one another, trying to keep both in the loop of what is going on in all parts of the program. The idea that two eyes are better than one is often true, but for efficiency we will not follow that type of extreme requirement. Large parts of the programming will however definitely be constructed as a joint effort solving the problems at each step. The problems with high complexity can definitely use this type of co-programming.

4.1.3.2 Coding standards

As a natural consequence of trying to make it easier to help one another we will also adhere to creating some joint coding standards before the programming. As switching tasks and getting feedback on code is important when working with complex problems, this is something we will strive to follow. This hopefully lowers the threshold of analyzing code and problems someone else is wrestling with. Since most of the code will be written in Python we will, for the most part, be following the Python coding standards laid out in PEP-8[39].

4.1.4 Evaluation

The evaluation of such a narrow goal – what human behaviour patterns can be detected by a robot, and used for initiating a conversation between said robot and a willing human in a, to the human, natural way – will be difficult. The actual conversation initiation process, the way humans do it, is very short and subtle and the established conversation quickly moves on to the dialogue phase. However, with carefully constructed scenarios and user testing it might be possible to objectively evaluate the correctness of the prototype and the validity of the suggested approach, which is what we will attempt to do. Said scenarios will include a few different people with different spatial relationships to the prototype, with varying degrees of 'willingness' to engage in conversation. There might also be a variable number of people in front of the robot (or camera), which the final prototype should also be able to handle. To reduce problems with the test subjects' immersion with regards to seeking eye contact with a camera, the camera will either be mounted on a human-looking robot, though the robot will not be active during the tests, or behind a human face mask. No personal data will be saved (i.e. no images of the test subjects).

These user studies will be quick and dirty, and primarily used to inform the direction of the next iteration of the development process, though they may uncover some deeper flaws in the requirements, which will then need adjustment. To start, they will be restricted to the development team, until such a time that the prototype is good enough that bringing in more test subjects can help uncover new failing situations. It is the intent to implement the prototype as a number of highly decoupled systems that can be built and tested mostly independent from each other. The evaluation process will therefore be employed separately on each of these systems, and only later, when each of the system has reached some definition of done, be evaluated together. This last evaluation will be near the end of the project and the result will hopefully be of use in determining the validity of this project's proposed approach to human-robot conversation initiation.

4.1.4.1 Privacy issues

The user tests that will take place outside the development team will inevitably involve users being filmed by camera, and having some of their facial features analyzed. Since this can be a sensitive matter to people, they will be informed beforehand that no images taken by the camera will be saved after the test concludes, and all data gathered will be completely anonymous. Should the test subject wish to withdraw from the test at any point during the test they are free to do so without any repercussions, and any data gathered will be immediately and irrevocably destroyed.

4.2 Scientific method

Wadsworth outlines a certain ruleset[40] or methodology that applies to the research phase of any design or programming project. These rules will be used as guiding principles during the project. The sixth rule, for example, states that you should "be rigorous and skeptical about your observations and existing assumptions", which will probably prove very useful in a project like this, as confirmation bias is always a problem. An example in this project would be the assumption that eye gaze is the most important social cue for a upcoming conversation. This assumption would have you start searching for eye gaze, finding articles about that and just reinforcing your initial idea of this being the most important. To instead ask the question "What social cues are there and what are the prioritization of those?" not assuming you already know the answer would be a better approach.

To make assumptions however is not inherently wrong but can be a powerful way of finding more information about the subject. Rule seven states: "Use your imagination to get deeper, richer and wider understandings and explanations– and check these out as part of the research" highlighting that using your imagination, your guesses and thoughts might lead you into deeper understanding of the subject at hand, and your assumption that eye gaze is important might have lead you in the right path from the beginning. So listening to yourself is important, but be critical of your solutions and assumptions.

Another rule that resonated very well with our methodology of this project was the ninth rule stating that "Good research interprets and analyses the findings (takes them to pieces) and then synthesizes (puts them together) in new ways. It makes explicit links between theory and evidence, explanation and description, and tells a story of how things are, or were, or could be." This is something to be aware of, and something that is easy to forget when for example looking at similar studies on a subject. To solve the problem of EGE it is easy to end up looking for other studies doing just that. But the answer you are looking for does not have to come from that type of source. If you break down the problem, what you are really trying to do is to decide if object A is of type 1 or 2. This is a binary classification problem and, by identifying that, you could find work that has nothing to do with eyes, but solves this same type of problem with something else. To be aware that you could take bits and pieces from what could seem to be unrelated studies and put it together in a new way to solve a different problem is important.

5 Implementation

5.1 Requirements

From our initial research, and with feedback from the stakeholders, we managed to establish a number of requirements for the overall solution of this project:

1. Eye contact is the most important social cue when determining interest in initiating a conversation, and must therefore be used in our solution.
2. Machine learning must be used in some capacity, due to interest from the stakeholders.
3. The underlying platform on which all the different parts of the prototype will run, should be modular in its architecture to help future work.
4. The solution must be performant enough to work for multiple people simultaneously, in real-time.

The first two requirements together form the basis for a major part of this project. We train a convolutional neural network to be able to detect eye contact in humans. After research in this area, searching for appropriate data sets to train on, we decided to manually label images of people in the wild. This was due to the lack of data sets containing both accurate labeling for eye contact as well as being diverse in both the environment where the images were taken and in its participants.

The definition of real-time is a little vague, but for this project we will accept a solution that is fast enough that the robot's reactions do not feel awkward or unnatural for the people trying to interact with it. Thus, one update per second is likely too slow, while five updates per second should be good enough.

5.2 Image labeling

Since a lot of research has been done to show that ANNs are capable of determining where, in relation to the camera, a person is looking, it stands to reason that they are equally capable of determining whether a person is looking at the robot (camera) or not (i.e. having eye contact). ANNs requires large amounts of data to train on, and it is usually the case that more data equals better results. Since the data set has to be tailored specifically to what the ANN is supposed to learn, the EGE will require a vast amount of images of human faces, both with and without eye contact. These images should be in different lighting conditions, with different backgrounds, and with a large variation on the humans' appearances. Today, such data is readily available on the internet, through Google for example, and there are multiple large data sets available that have been previously used for similar endeavours. These data sets constitute the bulk of the data that will be used for training, only supplemented with more images from Google or other online image databases if necessary. However, the data had to be manually labeled, as the labels needed for eye contact (i.e. eye contact/no eye contact) often differ from the labels originally put on the data (such as where on a smartphone screen the person is looking).

The eye contact module uses a convolutional neural network to make estimations. Eye contact is, at least for this project, a binary state; you either have eye contact with someone or you do not, there is no in-between. As such, the data required to train the neural network must consist of thousands of images of people either with or without eye contact (with the camera).

5.2.1 Finding data

Finding and collecting enough images to create a whole data set can be a very time-consuming task. Many previous projects in this area have created their own data sets for their special needs, some through the use of Google image search, others by distributing a smartphone app that willing participants can contribute their own images through. Fortunately, many such projects also make their data sets available for use by future research projects. By downloading and labeling several of these data sets, and adjusting them to fit the needs of our neural network, we managed to save a potentially enormous amount of time.

The largest dataset we used was the LFW-dataset[41] from which we used over 9000 images for training our model. This was however not enough to make a decent training, so we used several other datasets as well. Those can be seen in the table below.

	LFW[41]	Mugshots[42]	Bioid[43]	Caltech[44]	Essex[45]	FEI[46]	Stirling[47]	Total
Negative	6368	131	140	72	390	1924	662	9687
Positive	2976	920	792	446	1246	751	895	8026
Total	9344	1051	932	518	1636	2675	1557	17713

These were databases containing images of people and faces that were free of use for research purposes. These images were as training and validation images during the training of the neural network of the project.

5.2.2 Tools development

Labeling has to be done manually, one image at a time. While it may be difficult to make such a job interesting or exciting, certain steps can be taken to at least simplify it. We developed a tool, the *Binary Labeler*, that allows us to efficiently label images with the press of a button. To use the tool, one supplies the name of the data set, a destination path on the hard disk, where the resulting data set will be created, and any number of folders containing images that are to be labeled.

```
binlab.py --label lfw ~/labeled/lfw ~/original/lfw/A*
```

Running this command will launch the program in labeling mode (one of four modes) and present the user with the first of the images with a name starting with 'A'. The user then press one of 'f', 'j', 't' or space to label the image as negative, positive, trash or undecided respectively. The trash label is for images that are either too low quality or have other problems that make them unusable, such as large amounts of graphical artifacts, while the undecided label is for when the user has trouble decided if the image is positive or negative. The main reason to label images as undecided is speed; there are a lot of images to label, and one can not afford to put more than a few seconds on each image. However, images that are difficult for one person to label might be easier for another, and so they are saved as undecided for future evaluation. The tool has a secondary mode, review, which can be used to quickly go through the undecided images again at a later date. Once an image has been labeled, the next image is immediately shown, until all images have been labeled. At that point, all the actions taken are written to file (the manifest), and all images are copied to their respective folders (positive, negative, trash, undecided) under the destination folder (see fig. 5.1 for a flowchart of the process).

The manifest file saves all the decisions made when labeling a data set, allowing future recreation of the labeled data set if something goes wrong.

Listing 5.1: Excerpt from a manifest

```
/Azra_Akin/Azra_Akin_0003.jpg -> /negative
/Azra_Akin/Azra_Akin_0004.jpg -> /positive
/BB_King/BB_King_0001.jpg -> /positive
/BJ_Habibie/BJ_Habibie_0001.jpg -> /negative
/Babe_Ruth/Babe_Ruth_0001.jpg -> /negative
```

Saving the actions taken this way also allows multiple people to work on the same data set, as multiple manifests can be combined into one, which can then be used to recreate the complete labeled data set.

```
binlab.py --combine ~/labeled/lfw/manifest.txt ~/temporary_manifest.txt
binlab.py --recreate ~/original/lfw ~/labeled/lfw/manifest.txt
```

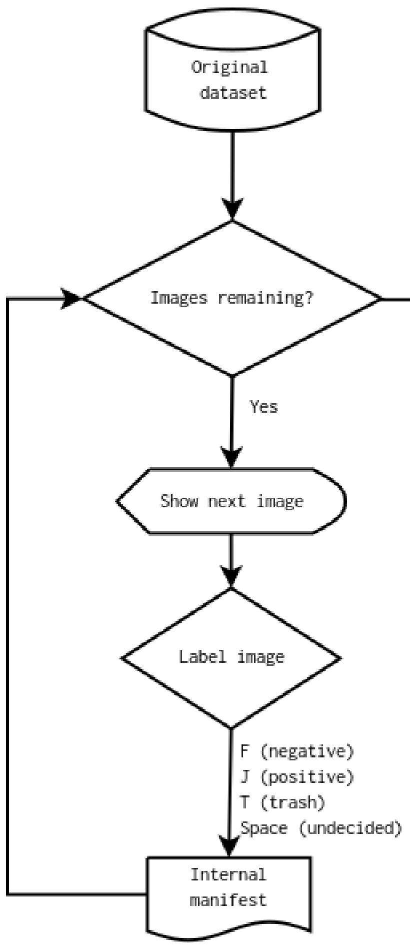


Figure 5.1: Flowchart depicting the labeling of images with the *Binary Labeler* tool.

5.2.3 Dividing the data sets

Instead of combining all data sets together in one folder the data sets origin was preserved as well as keeping all of them separate. This would enable us to try out different data sets, evaluate the results and then only train on those that proved valuable. When starting a training session you could therefore choose on which data sets you wanted to train on, and vary those trying to get different results. It gave us one more parameter to control during training.

This solution rose only due to that all the data sets come from different sources. One data set might be images taken in lab setting, where people were told where too look and where to stand, whilst others were just pulled from image databases on the internet.

5.2.4 The actual labeling

The labeling was done by both members of the development team simultaneously, taking different parts of the data set and manually labeling the images using the labeling tool. After about 5000 images were labeled, we identified a problem. We discovered when going through the already labeled images was that the people labeling had different opinions on what constitutes eye contact and what does not. One person labeling reacted entirely on feeling, meaning that if he did not feel the eye contact through the screen, he labeled the image as negative. The other person labeling looked at the problem a bit differently. He also reacted on feeling when an image was clearly eye contact that he could feel through the screen, but he also evaluated images that perhaps did not feel like eye contact, even though it technically looked like it. These were images where the person might have looked just above or next to the camera.

After this discovery a discussion, determining how labeling was to be conducted in the continuing weeks, was had resulting in us agreeing on not just reacting on feeling, but also including people who looks like they could have eye contact, or that it at least could be misinterpreted as such. This was also a attempt of increasing the number of eye contact images, as the distribution between the two was already getting uneven.

5.3 Eye-gaze estimation - neural network

The eye gaze estimator is the main part of the EGE module, which is one of the primary goals of this project. It includes constructing a convolutional neural network with the Tensorflow library, and training it on the assembled data sets. This part of the work may take a long time, but a lot of that time is spent waiting on the training to complete, a process which may take many hours. Primarily, the training take place during nights, since it is fully automatic, but quite often it also take place during the day, meaning that other work can be done in parallel. Since testing is done together with the training, the training procedure, once it is finished, automatically reports a result of how well it did as a percentage of how many of the final round of testing images it managed to classify correctly. Later the neural net model is deployed to the eye gaze estimation module and used together with the rest of the system for some real-world and real-time testing.

5.3.1 Auto-crop tool

Our goal is to have a prototype that should be able to recognize a face in an image, an then estimate the likelihood of that person being interested in talking. But for the testing and training of the neural net we needed an easy way of both labeling images, see section 5.2.2 but also a quick and easy way of converting those photos of people into smaller images just showing their eyes. We also needed a guarantee that if there were several eyes in an image it would only pick the pair that was labeled.

5.3.1.1 The cropping

This auto-crop feature was inspired from an article published by Adrian Rosebrock[48] detailing how you could use dlib[25] and Python to retrieve different facial landmarks from an image. As we wanted to include both eyebrows and eyes in our training data, this allowed us to pick and choose what facial landmarks we wanted. First using the "get_frontal_face_detector()" function from dlib we get a detector the can be used to find faces in the image. Then using a predictor from dlib, landmarks can be found in a face. The variables received from this "FACIAL_LANDMARKS_IDXS" dictionary can be used as indexes in the array that the predictor returns. In that array coordinates for different landmarks are stored. Using these coordinates we can create a rectangle that we can later use to crop those landmarks out of an image.

```
temp = shape2[i:j]
temp = np.concatenate((temp, shape2[k:l]))
temp = np.concatenate((temp, shape2[m:n]))
temp = np.concatenate((temp, shape2[o:p]))

(x, y, w, h) = cv2.boundingRect(np.array(temp))
```

Thanks to the image being a numpy array its very simple to crop out the rectangle from the image using python slicing. An issue was that the images that were created were of different sizes. If someone was tilting their head, the height of the rectangle could increase rapidly. As we wanted the training images to be the same size, the auto-crop tool also had to resize them. This was done using OpenCV resizing functions along with functions to create a black border around the image, to make it consistent in size.

5.3.1.2 Using the tool

To find and crop out the eyes from folders of images we use the following.

```
eyedetect.py dlib/predictor.dat ~/datasets/lfw/ ~/datasets/lfw_eyes/
```

The first argument is the location of the predictor model used for finding the eyes, the second is the original labeled data set and the third is the location where you store the images. The auto-crop automatically skips images that has either more than one pair of eyes in them or has none. This way the location folder should only be filled with images of eyes, and not include anything else.

5.3.2 Preparing the data set

All labeled images in the data set had to be prepared before being able to train the neural net on them. As we wanted to keep different data sets separate, but still being able to use them together, all data sets were kept in folders named after the specific image data set from where they came. Each folder had the same structure. They contained one folder for images marked positive of containing eyecontact and one for those that were marked as negative.

The value of keeping different data sets separate was so that we could try to exclude and include different sets to get different results on the training, effectively having another parameter to control.

Each data set provided was prepared individually and then put into an array containing all the combined prepared images. The individual preparation was done as follows:

1. Reading the image as grayscale
2. Resizing the image to a specfied size, giving all the training images the same size
3. Reshaping the image into a numpy array with the correct shape
4. Data augmentation
5. Saving data into arrays, one for each class (label)
6. Combining the arrays
7. Shuffling the arrays

To reduce the size and complexity of the neural net and the model for predictions it produces the images were chosen to be used as gray scale images. This could be used since eye contact is easily distinguished in gray images. The reading and the conversion to grey scale was done using OpenCV.

After reading, the image was resized to a predefined image size. As the neural net uses the size of the images to decide input and output to different layers its important that the images are the same size. You can build a neural net that does not use a predefined image size but since that was not really needed in this project

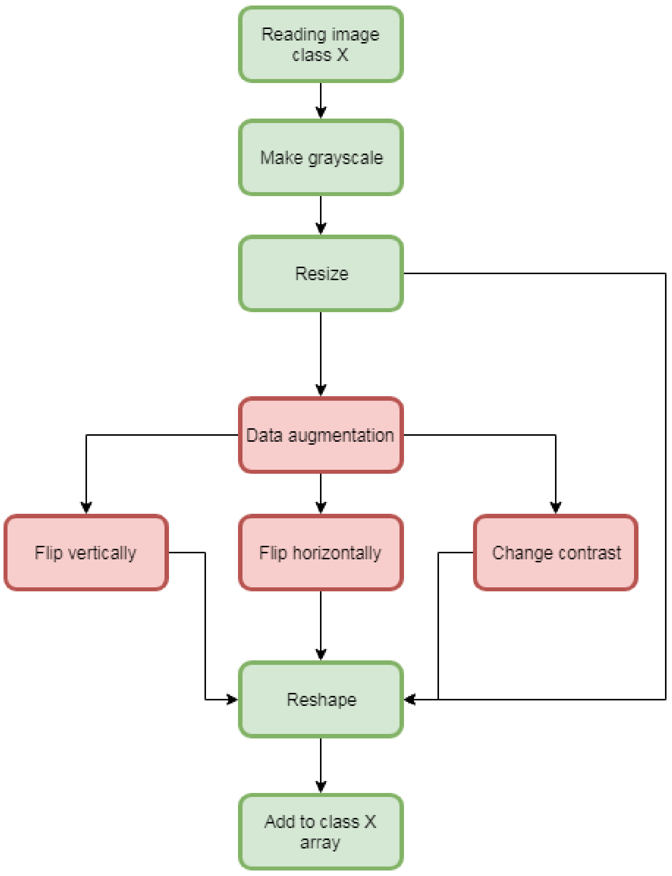


Figure 5.2: Flowchart of data set preparation

it was not explored further. Instead of doing the resizing at run time while training, we decided to do it during the preprocessing stage but both alternatives are valid.

The reshaping stage is about making the image into a numpy array that the neural net can use when training. When doing this reshaping there was one fault that came with using OpenCV to do the gray scale conversion. When making an OpenCV image into a numpy array that array has a shape. For an RGB image it gets the shape of

```
(image_size, image_size, 3)
```

where the three describes how many channels the image has (three channels one for red, one for blue and one for green). When converting a gray image however you get the shape

```
(image_size, image_size)
```

which is problematic, as it does not detail the number of channels (for a gray image this would be 1). This disappears in the conversion, and this axis must be added for the numpy array to have the right shape. If the neural net assumes a certain shape that shape must be guaranteed in the pre-processing stage.

The next step was the data augmentation. Data augmentation during the pre-processing stage allowed us to increase the size of our data set without increasing the amount of space required on the hard drive. This was done between the reshaping and the resizing steps. Three different augmentation techniques were used; vertical flipping, horizontal flipping and changing the contrast of the image. The vertical flipping, although in theory would still render images that has eye contact, is not really mirroring reality, as no one will approach the robot hanging upside down. The horizontal flipping however is a good way to double the size of the data set with images that are mirrored versions of the original images. The last data augmentation technique used was the changing of contrast in the image. This creates an image that is vastly different in look but still keeps the eye contact.

Each augmented image gets reshaped and is saved into the array for the corresponding class. The last step of the pre-processing is combining the arrays. Before combining the arrays of the two classes we even out the size of the two classes. The reason for this is to reduce bias for either class. If you would train the network on twice as many positive images than negative, a result of 67 percent accuracy might trick you to believe that the neural net works and is learning, as its generating an accuracy above 50 percent which is a misrepresentation. This was something that we learned during the early stages of the development of the neural net. Our neural net showed very promising results of 75-80 percent very early in the process, but we could not get it to get any better than that. At first glance it was assumed that it was because of the lack of data, but after observing a big difference in amount of data for the two different classes that basically mimicked our observed results, we tried to even out the data sets. The accuracy dropped to around 50 percent exposing a fault in the neural net as its actually only guessing randomly.

After making the two classes even in size, the arrays were combined and shuffled.

5.3.3 Graph

The convolutional neural network used in this project for the eye gaze estimation follows a common structure, alternating convolutional and pooling layers leading into a flattening layer which then leads to fully connected layers.

In 5.3 you can see the first part of the neural net where each pooling layer follows a corresponding convolutional layer. In implementation each layer is assigned a name, each name is a shortened version of the type of layer plus a number increasing for each new layer of the same type. The creation of each convolutional layer in this implementation consists of 4 steps:

1. Creating the weights
2. Creating the biases
3. Compute the 2D convolution
4. Adding the bias to the convolution

To create the weights and biases the Variable-function in Tensorflow is used. When creating the weights, a shape consisting of the filter size, number of input channels and the number filters specified for the specific layer is used. When creating the biases the only variable in the equation is the size, which here corresponds to the number of filters. Computing the 2D convolution is done using the TensorFlow *conv2d*-function.

This performs the convolution on the output from the previous layer with the previously calculated weight as filter. After the convolution, the biases are added. This way of building the neural net using TensorFlow was inspired by the *Deep MNIST for Experts*-tutorial found on the TensorFlow website[21].

After the steps above there is one more thing to do before giving the output of the convolutional layer to the pooling layer and that is normalization. The normalization used here is through a RELU activation function which normalizes the data, replacing any negative values with zeroes. This activation function is used by calling the *relu*-function in Tensorflow, passing the convolution as the argument. The output from this function is what later is passed to the pooling layer. In this neural network we use max pooling, and the actual pooling is done using the Tensorflow-function *max_pool*. This downsamples the incoming feature map x (which in this case would be the convolution from the previous convolution layer) by 2.

The output from this pooling layer is then used as input to the next convolutional layer, and so on. After three convolutional and pooling layers in a row, the output from the last pooling layer goes into a flattening layer. The flattening layer converts whats a feature map with a shape of 4 to a feature vector with the shape of 2. The flattened feature vector is then passed to the first fully connected layer "fc1" as seen in Fig 5.5. All neurons in this fully connected layer have full connections to the activation layers in the previous layer. The layer can then make a matrix multiplication between all the activation layers. The output from this layer is mapped into a dropout layer. This dropout layer is used to reduce overfitting. This dropout layer sends its output into the last layer, which is a fully connected layer (also called logits layer) which has an output which is as big as the number of classes.

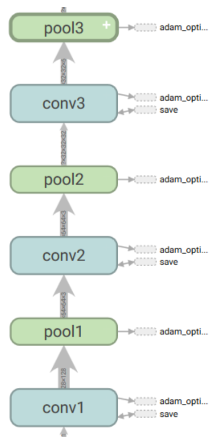


Figure 5.3: Three convolutional layers with a pooling layer between each. The output of one layer acts as input for the next layer and so forth.

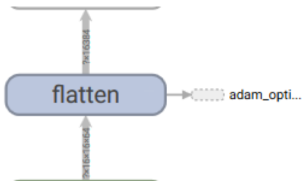


Figure 5.4: The output from the convolutional layers goes into a flattening layer

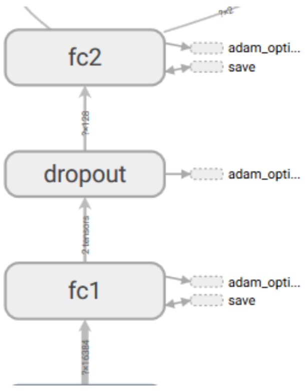


Figure 5.5: After the flattening layer there are two fully connected layers with a dropout-layer in between.

5.3.4 Training - technical

After the convolutional deep neural network, some calculations and some functions are used on the output to estimate accuracy and calculate loss. As an optimizer we use the “AdamOptimizer”[49] provided with Tensorflow, which handles the gradient descent.

When training this neural network there are a few different hyper parameters that you can control to tweak the results. You have the “learning rate” and the “epsilon”, which are both used in the AdamOptimizer. Another parameter is the “batch_size” which defines the number of images that each training step will handle.

5.4 Training

The training took up a significant part of the implementation process and the details of this will be described below.

5.4.1 Tools

5.4.1.1 Queuer

Some projects have data sets containing millions of images. Since the EGE will be binary in nature, only deciding between eye contact or no eye contact, the data set can be a lot smaller, which will also help to keep the project within the time limit. It is not uncommon that training on the more massive data sets can take several days, but with our smaller data set, one training session takes a few hours at most. To maximize the number of training sessions we could run, we needed to be able to queue up sessions with different parameters to run during the night.

The Queuer acts as an interface to the Task Spooler program[50], allowing us to queue training sessions with parameters that get syntactically checked before the job is added to the job queue. It also queues cleanup jobs for each of the training sessions, since they tend to save a lot of data, all of which is not desirable to save. This solution meant we could queue up multiple jobs which would run sequentially through the night, without any fear of crashing due to syntactically incorrect parameters or similar problems.

5.4.1.2 Auto-queuer

Despite the usefulness of the Queuer, manually specifying and adding a large number of jobs would quickly become tedious, especially since the jobs usually only differed in one or two parameters. The Auto-queuer automates the work flow even further by allowing us to specify several different arguments for each parameter. All possible combinations of those arguments could then be run through the Queuer, to create a large number of jobs very quickly:

```
Start value for iterations: 28000
End value for iterations (28000): 30000
Stepping value for iterations (2000):
Batch sizes (32): 32 64
Learning rates (0.0001):
First fully connected layer sizes (128):
Epsilons (0.1):
Flip vertical? [y/N/b]: n
Flip horizontal? [y/N/b]: y
Randomize brightness? [y/N/b]: y

Available datasets:
essex: N(390), P(1246)
essex_eyes: N(389), P(1244)
lfw: N(6368), P(2976)
```

```
lfw_eyes: N(6271), P(2921)

mugshots: N(131), P(920)
mugshots_eyes: N(129), P(913)
stirling: N(662), P(895)
stirling_eyes: N(441), P(895)

Datasets: lfw_eyes mugshots_eyes stirling_eyes

Datasets usage [C(ombined)/s(eparately)/a(ll combos)]: C
Results:
Iterations 28000 Batch_size 32 Learning_rate 0.0001 Fc1_layer_size 128
   Epsilon 0.1 Flipv 0 Fliph 1 Bright 1 Dataset lfw_eyes mugshots_eyes
   stirling_eyes
Iterations 28000 Batch_size 64 Learning_rate 0.0001 Fc1_layer_size 128
   Epsilon 0.1 Flipv 0 Fliph 1 Bright 1 Dataset lfw_eyes mugshots_eyes
   stirling_eyes
Iterations 30000 Batch_size 32 Learning_rate 0.0001 Fc1_layer_size 128
   Epsilon 0.1 Flipv 0 Fliph 1 Bright 1 Dataset lfw_eyes mugshots_eyes
   stirling_eyes
Iterations 30000 Batch_size 64 Learning_rate 0.0001 Fc1_layer_size 128
   Epsilon 0.1 Flipv 0 Fliph 1 Bright 1 Dataset lfw_eyes mugshots_eyes
   stirling_eyes
Good? [Y/n] y
```

5.4.2 Training

By training a neural network, you create models to which you can later input similar data and hopefully get a correct answer. It is possible to create many models from the same neural network simply by throwing away the old model and starting over. A training session consists of feeding the model a subset of the data, letting it guess what the truth (eye contact, no eye contact) for each data point is, and adjust the neurons activation thresholds in response to how well it guessed. This process is done many times, slowly forcing the neural network to become better at guessing what the data means.

The training was done iteratively, starting with only a few thousand images from the LFW[41] data set. These training sessions were mostly done to try out the network and the data set to see if it worked at all. With an even distribution of images en each label in the training set, the training consisted of beating the previous estimation. With only a few thousand images the estimations where circling around 50% with a few exceptions here and there where it probably just got lucky. At this stage, all the hyper parameters were hard coded in the neural net, so the testing only changed in what images we provided. When the neural network appeared to act as it was supposed to, the hard coded parameters could be changed by giving different arguments to the training directly from the command-line.

A problem with this solution was that our inexperience with training neural networks forced us to try a lot of different combinations of hyper parameters to get a “feel” for what works and what does not. Keeping track of the results and the inability to try out several different combinations in succession without manually starting them each time showed the need of the Queuer tool described in section 5.4.1. With this tool we could specify several training sessions in a row and let them execute sequentially during the night.

The results at this time were disappointing with the accuracy barely being over 60 percent. This was quickly attributed to a lack of data, prompting labeling the rest of the LFW data set and finding other data sets to label. The validation and the training accuracy greatly improved after adding data, but when we tested the predictor on the same images, the predictor always labeled the images as either true or false, giving an accuracy of around 50%. This result was disappointing as well as odd since the validation tests where performing at 80% and in theory the predictor should at least come close to that when used on the same batch of images.

This was later found to be a fault in the code, where the images where being reshaped twice, resulting in this odd behaviour.

After these initial bugs and problems the training followed the same pattern during the rest of the project. We queued several training sessions over night, using different parameters that could change the result. The next day we analyzed the performance of the training and tried to find what works and what does not. We found that a batch size of at least 64, and the lowering of the learning rate gave promising results. As the number of different input parameters for each training increased each time we added more data sets, manually entering all of these different combinations of parameters became a daunting task. To help with this the “Auto-queuer” was used. It helped us with only changing the parameters we wanted, as well as providing a simple interface to do so.

5.5 Platform

Detecting eye contact, and other "tells", for each person in view is a continuous task that must be on-going at all times while the robot is on. Keeping track of people, updating their estimates over time, and allow easy integration of more body language-reading modules, requires a continuously running program which the robot can interact with. Since the calculations for tracking people and estimating their eye contact, among other things, are quite heavy, the platform will have to run on the cloud, and communication with it done over network sockets. The platform is a collection of processes with different responsibilities:

1. Person registry system - the core of the platform
2. Person identifier system - identifies individuals in each frame received from the robot
3. Command-line - a text-based interface to the platform, operating over network sockets
4. Video stream - receives video from the robot (or any other camera) and sends the latest frame, on request, to the person identifier system

5.5.1 Person registry system

The Person Registry System (PRS) is the main part of the platform, responsible for keeping track of people currently (or very recently) in the robots field of view, and estimating their individual willingness to engage in conversation. Every individual is represented by an object, containing data about their current position and conversation initiation probability estimations. As people move into and out of the camera's field of view, such objects are continually added and removed from a list of the currently tracked people.

The PRS continuously receives an image and a list of bounding boxes, denoting people found in the image, from the Person Identifier System (PIS). Using the bounding box tracker (see 5.6), the PRS updates all known humans' positions, simultaneously adding new and removing old human objects as necessary. After updating all positions, each individual is sent to each estimation module, together with the image received from the PIS, to update its probabilities. Finally, the image and the list of individuals is sent to the command-line, from where data can be requested.

5.5.1.1 Estimation modules

The PRS has access to a list of currently active estimation modules, such as EGE. Every frame, all currently tracked individuals have their estimations updated by each estimation module. Utilizing the dynamic nature of the Python language, the estimation modules can add additional data to each human object as necessary. For example, since eye contact must be held for a short period of time to be considered an invitation of engagement (people typically do not consider engaging with others who let their eyes sweep over the room, even though they may obtain some measure of eye contact by doing so), the EGE module must be able to take time into consideration when calculating its estimates. The longer someone maintains eye contact, the likelier it is that they are interested in conversation. The time can be tracked easily by dynamically creating delta-time attributes in each human object, and reading and manipulating the value stored there in subsequent estimations:

```
# To track the time a person has maintained eye contact.
if not hasattr(human,"ege_delta_pos"): human.ege_delta_pos = 0
human.ege_delta_pos += (frame.creation_time -
    self.previous_frame_time).total_seconds()
```

5.5.2 Person identifier system

The PIS is a small but vital part of the platform. Its job is to identify areas in an image that contain a person and create a list of rectangles, or bounding boxes, that correspond to all such areas. Initially the goal was to create bounding boxes that included as much of a persons body as possible, though no more than what was visible. To do this, we tried both the Haar Cascade[51] classifier and the Histograms of Oriented Gradients (HOG)[52] classifier available through OpenCV. There exists a few pre-trained models for detecting various types of objects for each classifier. For example, the Haar Cascade classifier has models available for detecting either humans' upper bodies, lower bodies or whole bodies. Unfortunately, it turned out that correctly detecting human bodies in an image is actually quite difficult, probably because of the lack of distinct features and shapes on a human body.

Since none of the proposed probability estimation modules (eye contact, distance, head pose) actually need the whole body to function, we decided to focus on detecting faces, which is much easier due to the human face's multitude of distinct features, rather than whole bodies. Dlib[25] is a library containing many tools for machine learning and computer vision. It also contains a trained Convolutional Neural Network (CNN) as well as a HOG classifier for human face detection, both of which are very easy to use. Testing both, it turned out that the CNN is very good at detecting faces, but also quite slow. On the other hand, the HOG classifier is quite fast, and more than good enough for the purposes of this project.

5.5.3 Command line

Since the robot is meant to react to the data produced by the platform, it is important to allow the retrieval of the data, such as where the seemingly most interested person is, at any time. Additionally, for testing purposes it is important that we be able to communicate with the platform, to request data or start recording some data. The command-line operates only over a network socket, and features a very limited but dynamic instruction set. The following basic actions can be requested from the command-line:

- Print data - immediately return some requested data
- Stream data - set up a continuous stream of data, either to a network port, or to file

- Stream video - copy the incoming video stream and forward it, either to a network port, or to file

Each of these actions expect a number of qualifiers, specifically which data to get, and where to put it. For the video stream the possibilities are limited to either getting just the original video stream, or to add the detection bounding boxes and the estimation to it. This can then either be sent to a network port, or to a file. Example:

```
stream video base to network port 5000
stream video people to MyFileName
```

The data printer and streams have many more possibilities, too many to list. It is also easy to add more in the future, though the parser becomes exponentially more complex, since it was implemented as a very primitive if-then-else function to save time. It is possible to request multiple types of data at the same time, which will be returned as a comma-separated string. For example, to get the number of humans currently in view:

```
print count humans
    2
```

To get the individual estimations from each module (although the EGE is currently the only one implemented, for this example we simulate an extra module), together with the internal id assigned the respective person:

```
print estimations, id
    (0.786, 0.291):0
    (0.594, 0.912):1
```

To start a stream of individuals id's, bounding boxes and total estimations:

```
stream id, rectangles, total estimates to network port 5000
    Streaming data to port 5000...
```

The stream command does not return any data immediately, but rather starts listening to the specified port. Afterwards, any user can connect to that port and start receiving the data. A packet of data is available once per frame, so all streams are always in sync. This means that it is possible to request a video stream, and a stream of rectangles, draw the rectangles onto the video received and have them line up correctly to the faces present.

5.5.4 Video stream

To make good predictions about who the robot should engage with next, the platform requires a continuous video stream from the robot from which it can grab individual frames to work on. The first implementation used a standard network socket to send data, grabbed from a web camera through OpenCV's *VideoCapture* function, over TCP. While such a solution is very easy to get up and running, the quality and performance was underwhelming and not nearly good enough for this project's purposes.

5.5.4.1 FFmpeg

FFmpeg is a media conversion, editing and streaming tool, mainly aimed at transcoding media files. While not necessarily made for streaming a never-ending video stream (from a web camera) over network, it turns out it is quite easy to do so using, for example, Python's *socket* objects. Opening an FFmpeg process from with Python, using the *subprocess* library, FFmpeg can continuously stream to Linux's *stdout* file handle, which the *subprocess* library captures as an object. From this pipe object, it is then possible to read a single frame at a time given that we know the number of bytes in a frame ($width \times height \times colorchannels$), and send that data over a network socket.

```
frame_size = width*height*color_channels
command = ['ffmpeg', '-framerate', str(framerate), \
        '-video_size', '{}x{}'.format(width, height), \
        '-i', source, '-pix_fmt', 'gray', '-f', 'image2pipe', \
        '-vcodec', 'rawvideo', '-']
color_channels = 1
pipe = sp.Popen(command, stdout=sp.PIPE, bufsize=frame_sizes)
while True:
    socket.send(pipe.stdout.read(frame_size))
```

While it was easy to set up, and much more performant than the previous solution, there was still quite a bit of lag involved in the stream. This was possibly due to the use of the TCP protocol for sending the data over the network. Ideally, UDP should be used in such cases where speed is important, but while a few dropped frames would not matter (UDP makes no guarantee that packets arrive at all), frames arriving out of order (which UDP makes no guarantees about either) could cause serious problems with the estimation calculations and people tracking (see 5.6). While it is possible to create a small protocol on top of UDP to counter such problems, we decided to instead try out GStreamer.

5.5.4.2 GStreamer

GStreamer is a framework for combining multiple multimedia processing elements into a complete program, or pipeline, in order to do most anything regarding streaming, recording, transcoding, and editing of almost all types of media data. While quite complex to setup, its performance and stability proved to be very good. After testing various combinations of encodings, network protocols and cameras, we arrived at the following pipeline:

```
gst-launch-1.0 -v v4l2src !
   video/x-raw,format=I420,width=?,height=?,framerate=?/1 ! videoconvert
   ! videoscale ! jpegenc ! rtpjpegpay ! udpsink host=? port=?
```

With this pipeline, a Raspberry Pi 3 is capable of sending 1280x720 resolution at 30 frames per second, or 1920x1080 resolution at 15 frames per second. While other encodings, such as h264 offers somewhat better performance, there were some problems with graphical artifacts which is not tolerable.

5.6 Frame by frame tracking

Because the EGE module (see sec. 5.7.1) needs to keep track of the time that each individual has had eye contact, it is necessary to keep track of all individuals in front of the camera, without mixing them up, and recognize when someone new enters the image.

5.6.1 A very naive approach

Since the PIS creates a list of bounding boxes for every frame, tracking people should be possible by simply checking if two such rectangles, one from each of a pair of succeeding frames, are “almost equal”. The first naive implementation of this idea merely checked if two rectangles could be considered the same based on their positions and sizes.

```
def rects_kinda_equal(a, b, slack=5):
    return abs(a.top() - b.top()) <= slack and
           abs(a.left() - b.left()) <= slack and
           abs(a.right() - b.right()) <= slack and
           abs(a.bottom() - b.bottom()) <= slack
```

The slack parameter allowed some control over how exact two rectangles had to be to be considered equal, and while the solution did what it was supposed to do, allowing work on the rest of the PRS to continue, it was far from good enough to be used in the final implementation.

5.6.2 Dlib correlation tracker

Dlib contains a tool called the *correlation tracker* which can be used to track items through multiple frames. The tracker seems to work for any kind of object, and only requires an initial bounding box to work from. However, when using the tracker to detect faces in real-time, a few problems arose. The tracker does not seem to take the time between frames into consideration, leading to different results depending on frame rate. As a consequence, if the frame rate is low enough that people can move significant distances between frames, the tracker may fail to reacquire the correct person. Additionally, the tracker is only made to track one target at a time, and while it proved possible to work around this problem with more trackers working simultaneously, the most important problem remained: people may walk into and out of the image at any time.

Ironically, the problem that we attempted to solve with the help of the correlation tracker remained. Whenever a new person entered the image, that person had to be detected through the use of the HOG, and then their bounding box must be matched against all the ones created by the correlation trackers, to determine if this is indeed a new person or not. In effect, we were tracking people, but had no way to tell who was who. Together with the other difficulties encountered, we decided to forego the Dlib correlation tracker and instead create our own.

5.6.3 A slightly less naive approach

The third attempt once again compares rectangles, although in a more sophisticated and thought-through manner. Since many people can be gathered close together, it is important to get a confidence value, rather than just true/false, for each of the possible combinations between previous and new rectangles. The elapsed time between frames should also be taken into account, since longer time means that people can move further between frames. People can also move towards or away from the camera, which, in a 2d image, is represented as them getting larger or smaller respectively. Additionally, the further away from the camera a person is, the smaller their changes between frames (for a given movement) becomes (in terms of pixels), meaning that two rectangles, with a distance X between them, are less likely to be "the same" if they are further away from the camera (smaller) than if they are closer to the camera (larger).

5.6.3.1 Rectangles approximately equal?

All these things were taken into consideration when creating the new rectangle comparison function. A few assumptions and limitations had to be made in order to keep the code manageable:

1. Peoples' faces are all the same size, and the bounding boxes around those faces are always equally tight.
2. Image depth is a distance in the same way as height and width is, and can be specified to be any value.
3. The real distances represented in the image are irrelevant. Instead, one pixel is the basic unit of length.
4. Peoples' real movement speed is irrelevant. How far they are "allowed" to move between frames is entirely dependent on the time elapsed between frames, relative to the intended frame rate.

The first assumption allows for the abstraction that differences in rectangle sizes is actually a difference in distance to the camera. Together with the second assumption, it is possible to calculate distances on the z-axis in the same way as for the other two axes, as explained by assumption three. The inverted distance between two rectangles (calculated from their center points) serves as the basis for the final confidence value. A confidence value of 1 means that these rectangles occupy the same exact point in space, while a confidence value of 0 means that one is in a corner of the image (on all three axes), while the other is in the opposite corner (on all three axes).

```
def rectangles_approximately_equal(a, b, image, delta_time):
    # Calculate the depth of the rectangles as a fraction of the image
        size.
    a_depth = 1 - (Vec2D(a.width(), a.height()).hypo_2d() /
        image.hypo_2d())
    b_depth = 1 - (Vec2D(b.width(), b.height()).hypo_2d() /
        image.hypo_2d())

    # Calculate the absolute z-axis for the position vectors.
```

```
a_z = a_depth * image.z
b_z = b_depth * image.z

# Create position vectors.
v1 = Vec3D(a_center(), a_z)
v2 = Vec3D(b_center(), b_z)
dst = (v1 - v2).hypo_3d()
```

The distance between the two rectangles a and b, as calculated above, is in pixels (or voxels, since the image is now considered to be three dimensional). The image variable is an object representing the image, containing a width (x) and height (y) (that are the same as what the camera records with) and a depth (z) (whose value is arbitrarily chosen, though a value in the same neighborhood as the width and height seem to give best results).

To account for a variability in the time elapsed between frames, a variable is set by the system to be equal to the video streams intended frame rate (i.e. what frame rate the camera is set to record at). If the frame rate holds, the distance calculated above can be immediately used to calculate the confidence value that the rectangles are the same. However, if the time between frames gets larger due to lag somewhere, the distance “allowed” between rectangles must be compensated to give equal confidence values. The distance can then be converted to an inverted confidence value between 1 and 0 by dividing it with the image’s size in voxels. Then the average depth at which the rectangles reside is taken into account, adjusting the inverted confidence value upwards the further away from the camera the rectangles are. Finally, the confidence value is inverted, so 1 means that two rectangles are the same, and 0 means they are not the same. The resulting value is raised to the fourth power to give a less linear curve.

```
delta_time = delta_time * target_framerate
dst = dst / (delta_time * image.hypo_3d())
return (1 - (dst*mean(a_depth, b_depth)))**4
```

5.6.3.2 Rectangle pairing by confidence

The Tracker is an object that keeps a registry of any number of rectangles, representing people, tracking them through multiple frames. Each frame, the tracker is fed a list of bounding boxes detected in the current image, which it uses to update each of the rectangles in its registry. If there are multiple people in the image, each rectangle is updated according to what combination gives the highest total confidence value. Additionally, people can leave or enter the image between one frame and the next, further complicating tracking.

The tracker uses the rectangle equality approximation function described above to generate a table of confidence values for all combinations between the old and the new rectangles. From this table, the combination of pairs between old and new that gives the highest total confidence is selected, and for each of these pairs, the old rectangle is replaced with the new rectangle. If there are fewer new rectangles than old rectangles, meaning that someone has walked out of the image or turned their head away, the old rectangles that did not get updated are moved to a list of inactive rectangles.

If there are more new rectangles than old ones, someone has walked into the image or otherwise reappeared. For each of those leftover rectangles, that were not part of updating an old rectangle, the inactive list is searched for a matching rectangle, with those more recently added given priority. If no match of a high enough confidence is found, the remaining rectangles are returned to the trackers update call site. From there, they can be requested to be tracked as new rectangles, representing previously unknown people. Every rectangle in the inactive list has a value that keeps track of the time it has resided in the list. Once this time reaches a threshold, such as three seconds, the rectangle is deleted. This represents people that have walked away completely from the robot and are unlikely to return.

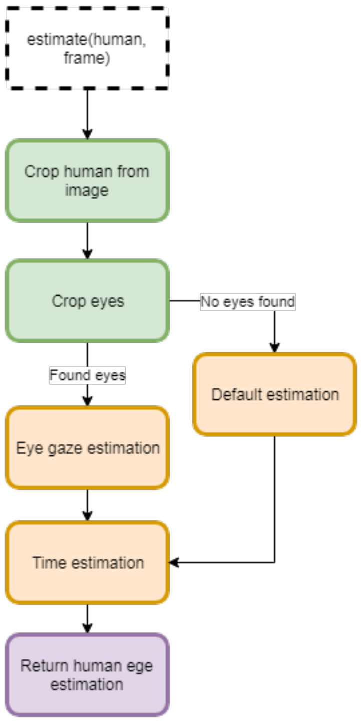


Figure 5.6: The different parts in the eye gaze estimation module

5.7 Estimation modules

The estimation modules are responsible for recognizing one select "tell" each, and produce corresponding parts of the Conversation Initiation Probability Estimation (CIPE) for every currently tracked individual. The modules are meant to be completely separated from each others, to make changes easy, and addition, removal or replacement of any module trivial.

5.7.1 Eye gaze estimation module

The EGE module is both an interface to the trained neural network, and an algorithm that helps determine eye contact with respect to time. The interface towards the PRS is shared between all the modules, and expects data about the person for whom the estimation is to be made, as well as data about the most recent frame. The complete estimation is dependent both on the EGE that the neural network provides, as well as the amount of time that the person has had eye contact. Fig. 5.6 illustrates how the different steps in this process connect to each other.

The frame data contains the complete image as given from the camera. Since the face detection and tracking steps have already been completed at this point, the personal data sent to the module contains, among other things, the bounding box that correlates with the given person's location in the image. By making a copy of only that part of the image, we massively reduce the amount of data that the rest of the algorithm needs to operate on. From this smaller image, the eyes are then found and similarly copied, since the neural network is trained to operate on eyes only.

The copying of the eyes is done in the same way as the auto-crop tool, described in section 5.3.1. If no eyes can be found in the image, the estimator continues anyway, using a default estimation, which we tentatively set to 0.5.

Once the neural network has finished its estimation, which we will call *NNE*, the complete estimation can be calculated with regards to the time that the person has had eye contact. The intent of this time-dependent behaviour is to avoid giving large estimations to people who happen to have eye contact for just a frame or two, such as when sweeping across the room with your eyes. The aim was first to arrive at the real estimation after 1.8 seconds of maintained eye contact (a number based of previous research about interaction between humans[30]), however this was later changed to 2.5 seconds to make it slightly easier to notice while testing. We call this the *AIM*. How long someone has maintained eye contact is added to each individual's personal data, and increased for each frame that the EGE remains positive, a variable we will call *T*. A temporary estimation is then calculated as:

$$\textbf{temporary estimation} = NNE \times \frac{min(T,\ AIM)^2}{AIM^2}$$

The last eight temporary estimations are always kept, and their mean value is returned as the total EGE. Whenever a person has negative eye contact, that duration is saved separately. The same equation is used for calculating that estimation, which is also kept as one of the eight last, but with an aim-time of 3 seconds instead, to make the drop-off a bit slower. The negative eye contact start time is reset whenever positive eye contact is restored.

5.7.2 Distance estimation module

The distance between two people can be interpreted as weak evidence of interest in conversing, as people rarely start their conversations when they are several meters apart. Sorokowska et al.[32] show that, while culture and the interaction participants' relationship create quite a bit of variation, people generally interact with each other at a distance of approximately one meter. Based on this, we can create a distance module that calculates a conversation initiation probability by how far away a person is. This probability can then be used together with the other modules' probabilities to calculate a total probability. In effect, people who are further away should have a slightly smaller chance of gaining the robot's attention than a person who is closer.

Ideally the distance between the robot and the people in front of it should be measured with radar or some other highly accurate method. However, since the distance module will mostly be used to prove the validity of the platform (that a person's "tells" can be determined separately and then combined to a total CIPE), we decided to save time by using a more straightforward approach. The fact that objects that are further away appear smaller than those that are closer is true for images as well. As such, the bounding box around a person's face, as supplied by the face detection function, should be smaller the further away that person is. While internal testing showed that the bounding box can change quite a bit in size depending on minute head rotations, it also seemed to hold true that the further away a person is, the smaller their bounding box is, most of the time.

We defined a probability of 1 when a person is one meter away from the camera. There is no hard lower limit, since everyone detected by the face detector should get a probability, however small. After averaging our own bounding boxes' sizes at two meters distance, the fraction of the image area covered by that bounding box was calculated to *0.0846* (at 640x480 pixels in the image). Using a logistic function, any bounding box's area can be converted to a probability between 1 and 0:

```python
class DistanceEstimator:
    def __init__(self, frame_width, frame_height, magic_number=0.0846):
        # magic_number is the fraction of the screen that a person should
            occupy to get P=1
        self.area = frame_width * frame_height * magic_number

    def estimate(self, human, frame):
        human.probabilities["dst"] = self.logistic(human.get_rect().area()
            / self.area)

    def logistic(self, x, x0=1, L=1, k=2):
        return L / (1 + math.e**(-k * (x - x0)))
```

This solution resulted in probabilities approaching 1 for anyone standing closer than one meter, 0.5 at two meters, and 0.2 at about three meters. While there are a lot of assumptions made for this solution, it can and will be used for testing the validity of the platform's approach. However, the distance calculation should ideally be replaced with dedicated hardware, such as radar, to more accurately measure the distance, and to provide linearity between points at different distances.

5.7.3 Head pose estimation

The Head Pose Estimation (HPE) module determines a person's interest in conversation through their head pose, i.e. the direction their nose is pointing. If the nose is pointing straight towards the robot, they should have a HPE of 1, while any rotation on either axis results in a smaller estimation. Dlib contains a facial landmark detector, based on the paper "One Millisecond Face Alignment with an Ensemble of Regression Trees"[53], which can be used to extract the positions of certain facial features from an image. Using this detector, we are able to extract the positions of the corners of both eyes, the tip of the nose, the corners of the mouth, the points where the jaw and ears meet, and the chin, in real-time.

Using a similar 3D/2D projection setup to that of Mallick's[54], we could extract a rotation vector and translation vector, representing the transform needed to go from the model coordinate system to the camera coordinate system. The rotation vector was then transformed to a 3x3 rotation matrix using Rodrigues' rotation formula[55] which, together with the translation vector, becomes the 3x4 projection matrix. Utilizing OpenCV's *decomposeProjectionMatrix* function, we can extract, among many other things, the three euler angles that represent pitch, yaw and roll.

```python
success, rotation_vector, translation_vector = cv.solvePnP(\
              self.MODEL_POINTS, image_points, camera_matrix, \
              dist_coeffs, flags=cv.SOLVEPNP_ITERATIVE)

rotation_matrix = cv.Rodrigues(rotation_vector)[0]
projection_matrix = np.c_[rotation_matrix, translation_vector]
_, _, _, _, _, _, euler_angles =
   cv.decomposeProjectionMatrix(projection_matrix)
pitch, yaw, roll = euler_angles[:,0]
```

Testing showed that yaw and roll were both 0 degrees while pitch was, on average, 173.5 degrees when facing the camera directly. The maximum range in either direction, for each of the three axes, was determined with a web camera. Any movement on either axis is divided by the axis' max value, resulting in three values between 0 and 1, representing as a fraction how much the head is turned. Since movement on the three axes may not be equally indicative of interest in conversation, these values were weighted against each other (0.35 for pitch, 0.4 for yaw and 0.25 for roll), and then added together, providing the final estimation.

5.7.4 Estimation weights

The different social cues that are, or may be, used to determine someone's interest in conversation may have wildly different predicting powers. According to Nummenmaa and Calder, eye contact is among the most important cues for determining a persons interest[4], and while distance and head pose are also important, they are likely not equally important. Therefore, each of the estimations provided by the estimation modules is weighted, similarly to the internal head pose weighting, and added together to provide the total estimation. While rigorous user testing is necessary to accurately determine the importance of each, internal testing provided what seemed like good starting values, weighting eye contact at 0.6, distance at 0.2 and head pose at 0.2. During the internal testing, which consisted of sitting in front of the camera, testing different values and discussing the results, these values gave a clear advantage to the eye contact, underlining its importance. The distance and head pose however, though not able to overthrow the eye contact, still provided alternative total estimation with visible differences between people standing close to the camera and far away, as well as between people with varied head poses.

6 Testing

This is the milestone of testing the solution from taking a picture, identifying humans, cropping and tracking any people in the image and then using the EGE-module to get an estimation of that persons interest in a conversation. These tests are used to evaluate the platform, our solution and the validity of this approach in solving the proposed problem, but will not work as material for further iterations of the prototype due to time constraints.

6.1 In-house image testing

A good deal of this project consisted of testing our solutions. To do this several tools were developed, data sets were created and to find and weed out the largest flaws of the neural-net, such as the uneven distribution of data, this was done in-house. The neural net was first evaluated by removing a chunk of the training data to use for validation. This validation was used to get an idea of how general the neural-net had become, and how well it does on images that it had not trained on. This validation set is however not enough to evaluate the solution as its just a chunk taken from the training set. This means that if the original data set had some flaw from the beginning, like not enough diversity in look or not enough images of a certain type, this flaw could also be inherited by the validation set, since its derived from the same data set. This is were some testing data sets must be used.

6.1.1 Data sets

Several data sets were created with the purpose to test the neural net on edge cases, in order to find possible shortcomings. These were pictures of people in the development team, trying to challenge the neural-net. The testing set quickly revealed a few shortcomings.

The neural net was consistently good at correctly estimating the images labeled positive in the data set, but less so with the negative ones. Since the images in the testing set was taken with a laptop web camera, many of the images were the subject looking down on the screen, below the camera. These kinds of images, even though very clear for a human of not having eye contact, was mistaken for being eye contact. When examining the training set, there was a lack of people looking down below the camera, meaning that it probably made its estimation based on something else entirely.

6.1.2 Tools

To make the process of making predictions on batches of images easier, a predict program was developed. This took the path of a generated estimation model and the folder for the images to test.

```
predict.py pos-neg-model/283.meta datasets/ch_eyes/positive/
```

This then ran a prediction on each individual image in that folder, generating an output that detailed how sure it was on its prediction, how many was positive and negative and which images those were.

```
...
eye_2-147.jpg
N: 0.0002220364403910935 | P: 0.9997779726982117
eye_2-105.jpg
N: 1.3779980690742377e-05 | P: 0.9999861717224121
eye_2-109.jpg
N: 3.522544744782863e-08 | P: 1.0
--- Labels ---
Negative: 10
Positive: 68
Undecided 3
--- Highest confidences ---
Negative: eye_2-12.jpg at C 1.0
Positive: eye_2-171.jpg at C 1.0
--- Confidence values (average) ---
Negative confidence: 0.1493589860204847
Positive confidence: 0.8506410159831109
Total confidence: 0.9789196422070633
```

This gave us another way of evaluating, letting us look at what individual images it had falsely labeled and speculate on why that is.

6.2 In-house live testing

While the previous in-house testing did not depend on any of the PIS PRS systems, the live testing did. The big advantage with doing the live tests was the shortened feedback loop as we could try to find edge cases and seeing the result live on the screen. This is much less work than taking pictures, putting them in a folder and then running a prediction and then again check the pictures that failed. The in-house live tests were conducted using a very simple interface (see fig 6.1)

Around each face in the image a colored square was drawn, and above that the current EGE is displayed in the form of how likely it is that this person is searching for eye contact. Each new person that enters the camera view gets a new color assigned to them, and these colors help us keep track of how well the PRS system is handling the tracking of multiple people in an image. We could now during tests observe how the estimation worked together with the time calculation and evaluate

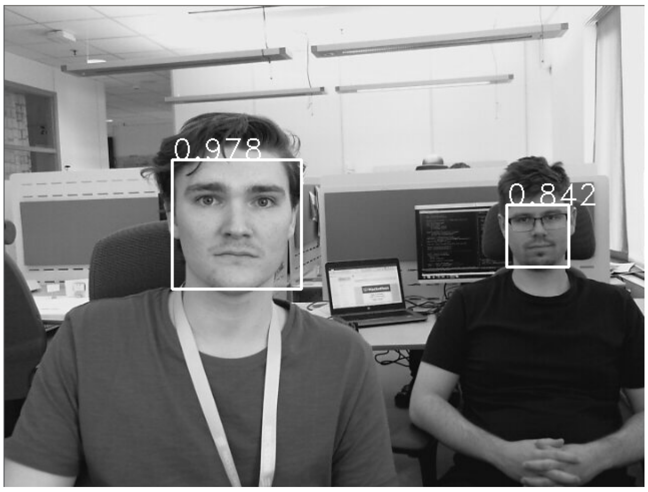


Figure 6.1: The first simple interface for live testing eye-gaze estimations.

how well the systems worked together.

6.3 Live testing

This section describes how the user tests were conducted and the preparations that had to be done.

6.3.1 Form

Before the live test, each participant has to answer a form asking some short questions about the participants, for example their height, age and whether or not they were wearing glasses. Each one of these papers are marked with an ID, something that will be used to keep track of individuals throughout the testing process. This form can be viewed in appendix B.

6.3.2 Paper prototype

For the live testing a small paper prototype was constructed(fig. 6.2). This enabled us to evaluate our testing area without having to build the full physical setup. The prototype consisted of three screens, one with the face of the robot on it and two other screens that would be used to show participants of the test how the robot sees them when they are not looking at it. Markings on the floor details where the participants should stand with a different floor area dedicated to each person. Using this prototype we found some different scenarios to test out during the actual test, using small paper notes as participants, placing them in different combinations on the markings.

As we wanted to gather a lot of data from each session the testing had to be very well thought out to effectively collect the data so that the actual testing would go smoothly without any long delays for both us and the participants. As much of

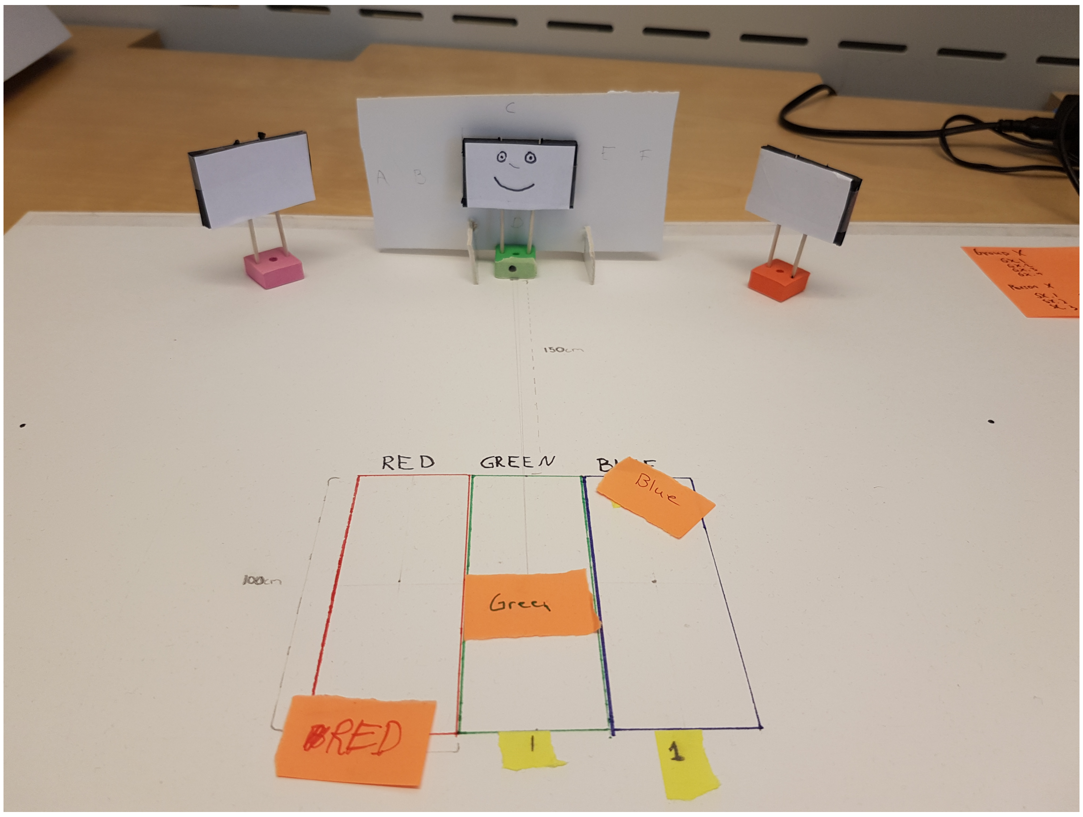


Figure 6.2: Paper prototype of the testing area

the data(the prediction for each person, the image, the position) was being handled in real-time while doing the test, we needed an effective way of saving each test.

This is why the test was decided to be organized much like a photo-shoot where we tell the participants where to stand and where to look with the help of clear signs and markings, and then taking an image and saving all the results with each "photo". This means that a snapshot of the state of the program is logged each time we "take a photo" and that file is placed in a folder named after the specific test being conducted, containing with each estimation the unique ID associated with each participant.

Using a unique ID for each participant lets us combine the answers from both the live test and the form they answered before the test to make profiles of how well it performs with different age groups, different heights, sexes and whether or not they are wearing glasses. This could give us a clearer image of how generalized the EGE is.

6.3.3 Pilot testing

We did some tests of the testing environment on beforehand, with a participant not directly involved in development. We tried out the manuscript, giving instructions and the logging and saving of data. Here we discovered quite a few bugs with the logging program which we could fix before the actual test. We made sure it saved all the data correctly, and that nothing was lost. Unfortunately we did not have time

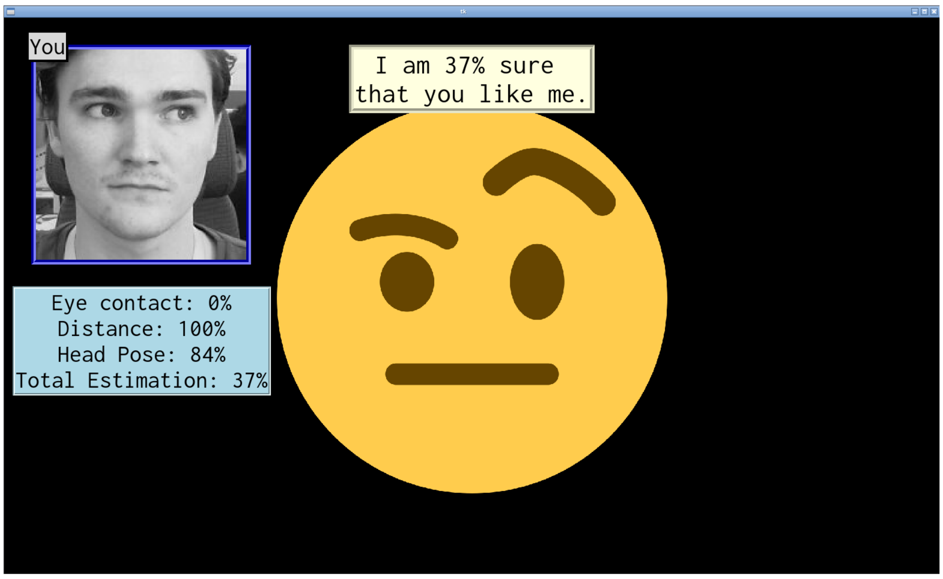


Figure 6.3: The robot screen GUI

to test the group test to the same extent, and this proved later to be a problem as the images from the group tests did not get saved properly.

We also found that the data was saved in the wrong order, as the red participants were saved as blue and the other way around.

6.3.4 Conveying information

Each participant was informed that there would be images taken of them and that these images would not be published anywhere. They were also informed about how the test would be conducted, and their role in the test. The full manuscript used before the test can be found in appendix B.

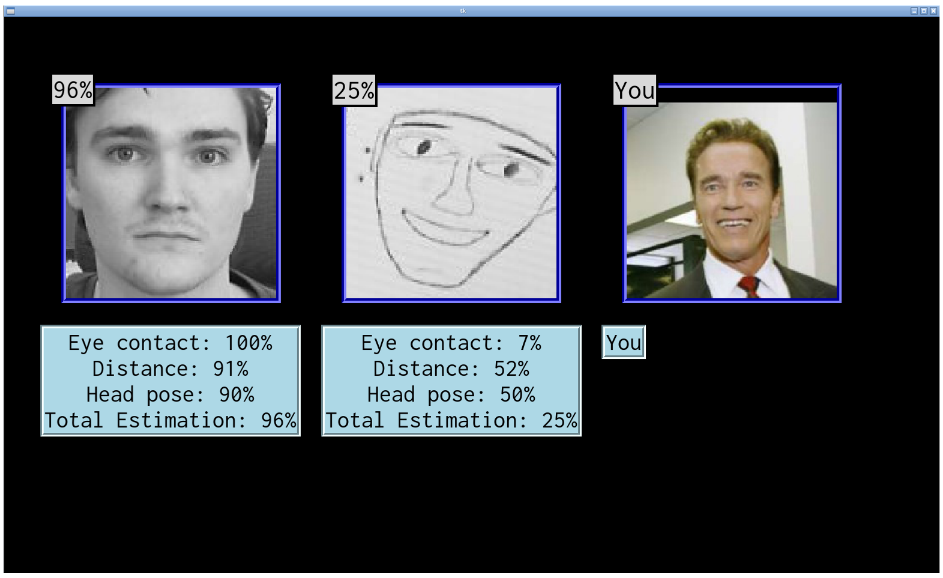
6.3.5 The test screens

To display and visualize the data, we use two different screens. One is the robot screen(Fig. 6.3), detailing who is most likely to wanting a conversation showing all the different estimations that goes into the total estimation.

The other screen is the two side screens, which are supposed to give the participant some feedback even when they are not looking at the robot(fig. 6.4). This screen will show up to three participants at a time, showing an image of the participant, along with their calculated estimations. This is placed side by side with the other participants.

6.3.6 The physical setup

The testing area was built according to the paper prototype with three screens, standing quite close to each other, with markings around the robot screen in the middle, see fig. 6.7. These screens were placed on a adjustable table effectively letting us adjust the height of the setup. The height of the camera was in the end



Eye contact: 100%
Distance: 91%
Head pose: 90%
Total Estimation: 96%

Eye contact: 7%
Distance: 52%
Head pose: 50%
Total Estimation: 25%

Figure 6.4: The side screen

set to 165 cm. The distance from where we placed the markings on the floor to the camera was 150 cm for the closest marking, and 250 cm for the one furthest from the camera. Each marking had a color and a number on them, detailing which marking it is(see fig. 6.5). Unlike the paper prototype, we did not make squares for people to stand in, but rather just taped the markings on the measured out places on the floor.

6.3.7 Single test

The single test was planned to evaluate the robustness of the EGE, giving each participant some one on one time with the robot. Following our instructions there were a few key things that were tested:

1. How much the distance from the camera affects the performance of the EGE.
2. How the weighing of distance vs eye contact works, evaluate the validity of a distance module.
3. How well it detects where a person is looking when they are looking at points close to the robot

The test consisted of showing the participant what marking to stand on and what markings to look at. For a thorough description of the test, see appendix A. The single tests were conducted with 46 different people, of whom 8 were female and 38 were men. Each person participated in a series of 14 different tests so in the end of the day we had conducted 644 single person tests. One person acted as the test leader, giving the participant instructions of where to stand and where to look, while one person prepared the next participants by describing what the test is about and giving them the form to fill out. Each form had an ID in the top right corner. When a new participant were to do the test, they simply gave their form to the test leader, who then logged the current test session with the ID on the form.

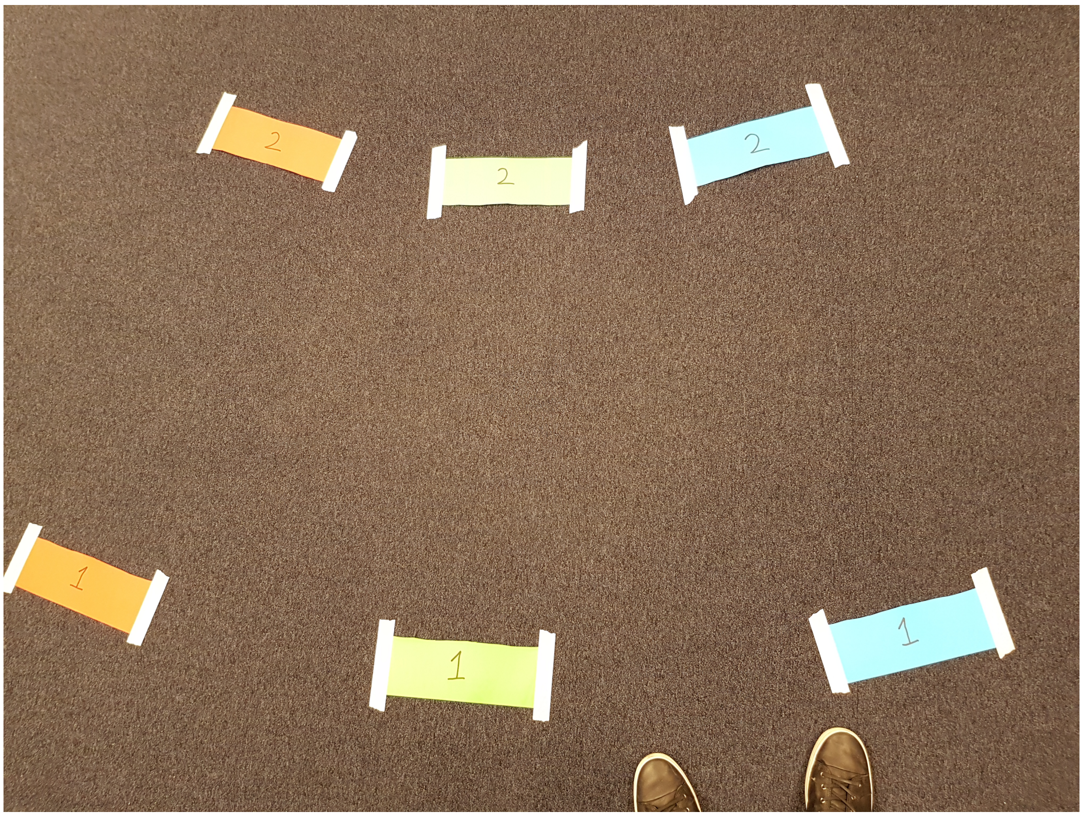


Figure 6.5: The floor markings

6.3.8 Group test

The group test was supposed to test the robustness of the whole system, including the EGE, the tracking of people in an image, the finding of people in an image and the decision making of the whole platform. The key points tested were

1. How does the multi-tracking work when three people are in the image.
2. How quick and viable is the finding of the faces in an image containing three people in real-time.
3. Does the system decide between people fairly/correctly, ie. does it look at people searching for eye contact, and leaving people not searching an interaction alone?

This test was conducted by placing each person in a designated area on the floor, telling them individual instructions of what to do throughout the test. For a thorough description of the group test, see appendix A.

The group tests were conducted with 27 different people forming 9 test groups. Each group did 4 tests each making the total group tests made 36 tests. The group tests were the most difficult to conduct with having to give different instructions to the different people in the test. It was harder to get the participants to follow the instructions.

For the group test, the test leader had to insert the ID of a person based on what marking color they were standing on, so that their data could be logged to their ID.

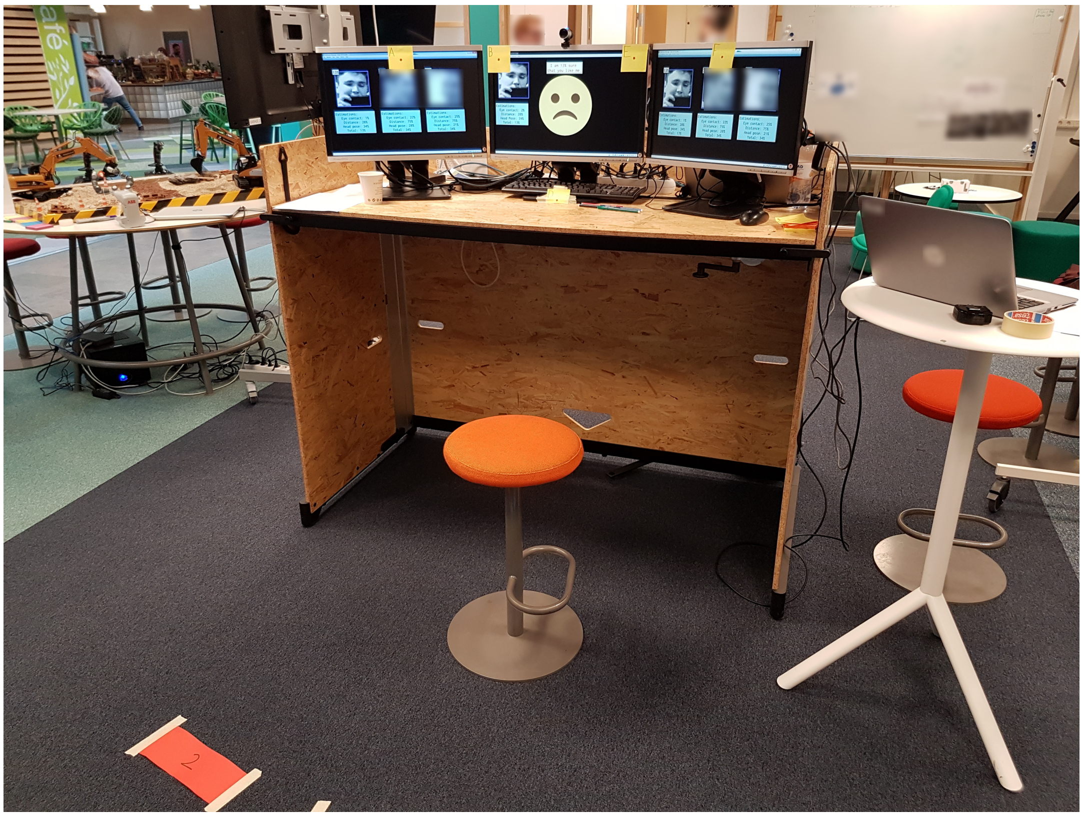


Figure 6.6: The testing setup, the tall table on the right is where the test leader conducted the test from.

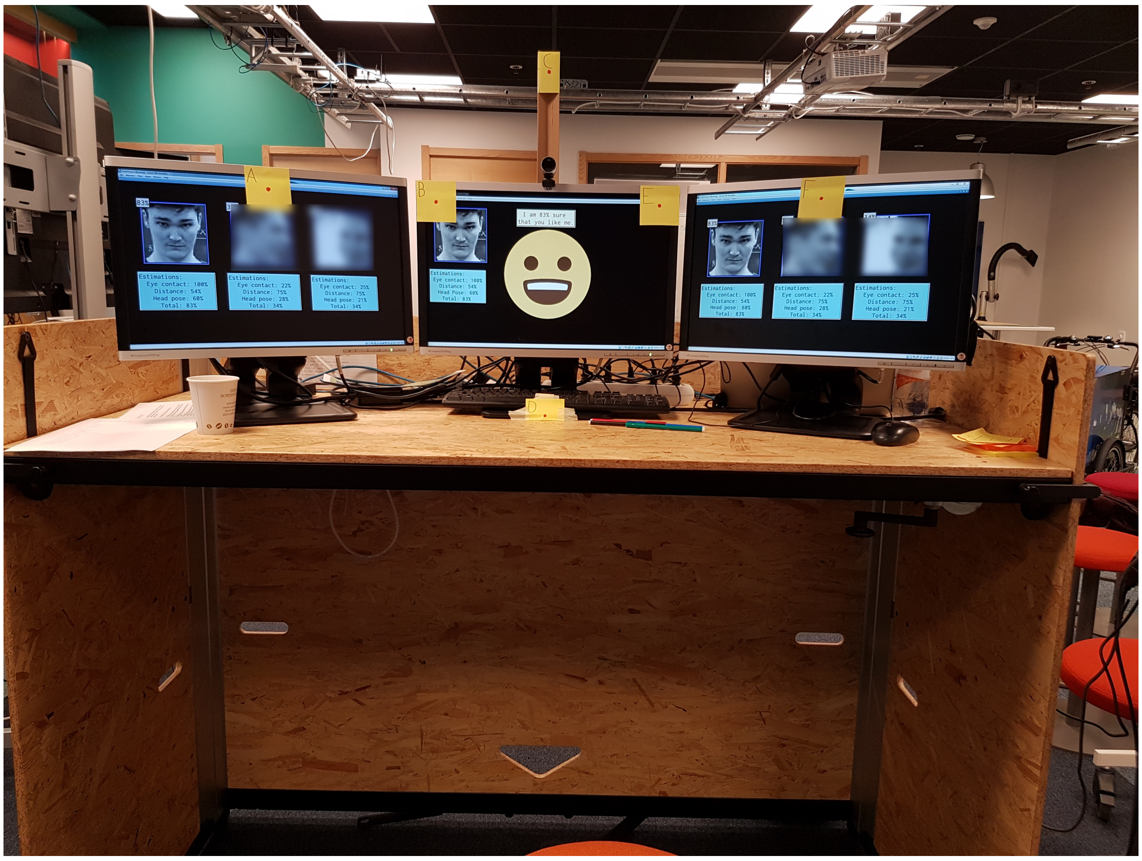


Figure 6.7: The screen setup with yellow markings from A to F showing the participant where to look.

6.3.9 Test results

The results from these tests where in the form of long data sheets, one for each test. The data saved was the distance, head pose and eye gaze estimation along with the ID and an image. This data was stored in tables, as seen in appendix D, figures D.2 - D.12. From these tables we constructed graphs that helped us visualize the data and draw conclusions.

7
Results

7.1 Data sets

26170 images were manually labeled, from which 17713 were deemed usable for training the neural net. Out of these images 9687 were labeled negative and 8026 positive. After cropping out eyes from these images and evening out the data set (to have equal amounts negative and positive images), 15732 images remained. After data augmentation, that number rose to 47196 images which the neural network trained on.

7.2 Estimation modules

7.2.1 Eye gaze estimation

The EGE neural network works quite well on average, although there are a number of problems with its results. From our test results we can draw a number of conclusions:

1. It is very good at determining eye contact when looking directly at the camera (figures C.1, C.2, C.3).
2. It has problems with determining eye contact for people wearing glasses (fig. C.1).
3. The test results indicate a problem with accurately determining non-eye contact with people looking to the right of the camera, though not with people looking to the left (figures C.1, C.2, C.3).
4. There is a definite problem with accurately determining non-eye contact with people looking above and below the camera (figures C.1, C.2, C.3).
5. Subject age does not seem to matter very much for the results (fig. C.2).
6. Subject height does matter, with taller people getting, on average, higher estimations, especially if they are further away from the camera (fig. C.3).

7.2.2 Distance estimation

Distance estimation is important for CIPE determination, especially when the system has to choose one among multiple people to engage with. Fig. C.4 shows that the current implementation works with the weights assigned (0.7 for EGE and 0.3 for distance), and tends to bring the CIPE down somewhat when standing far away (2.5 meters away). However, fig. C.5 shows that the distance estimation could, and probably should, also be used as a weight arbiter for the EGE: the further away the subject is, the less reliant the EGE becomes (though its maximum distance can be improved with a higher resolution video stream), and therefore, the EGE's estimation should weight less. The results shown use an EGE base weight of 0.8, which is modified by multiplying it with the square root of the distance estimation. This value is then used as a weight for the EGE, while the distance estimator is weighted at 1 minus the EGE weight. The results indicates that this approach gives a consistent and significant reduction in CIPE for people far away when compared to fig. C.4.

7.2.3 Head pose estimation

While the HPE module was not tested as much as the EGE and distance modules, the results from the group tests (fig. C.6), and test subjects' reactions to it, indicate that the HPE is not very good. It turns out that most people have a natural head pose that is not perfectly straight, with many people having their heads tilted to the left or right (roll), or up and down (pitch). The weights for pitch, yaw and roll used for these tests were 0.35, 0.40, 0.25 respectively. The variations in test subjects natural head poses, together with the fairly equal weights used, may have caused the quite equal values shown in the graph. A possible solution may be to put almost all weight on yaw, since that does not seem to vary as much between natural head poses.

7.3 Conversation initiation probability estimation

The user tests shows that the system is capable of determining who has the most interest in starting a conversation. If there is a very clear difference where two people are not showing interest, and one is (or if there is only one person present), it performs very well detecting that one person (see test case 3 in appendix A.2 and fig. C.6). However, if all three look at the camera, the head pose and distance estimations are basically nullified due to the uneven weighting of the modules. This causes the results to be inconclusive and subject to the problems of the EGE (see tests 2 and 4 in appendix A.2 and fig. C.6).

7.4 Platform

The platform is responsible for keeping track of people in the field of view, finding new people and creating CIPEs for every person. The various parts that make up the platform are intentionally decoupled from each other, running in their own separate threads (see fig. 7.1).

This is both for performance reasons and due to the asynchronous nature of some of the parts, such as the video stream receiver and command-line interface. The decoupling of major parts of the system also means it is very straightforward to replace most, if not all, parts with others that fulfill the same jobs. Adding and removing estimation modules is trivial, which will be very valuable for potential future work on the solution. The majority of the individual parts are very fast, with

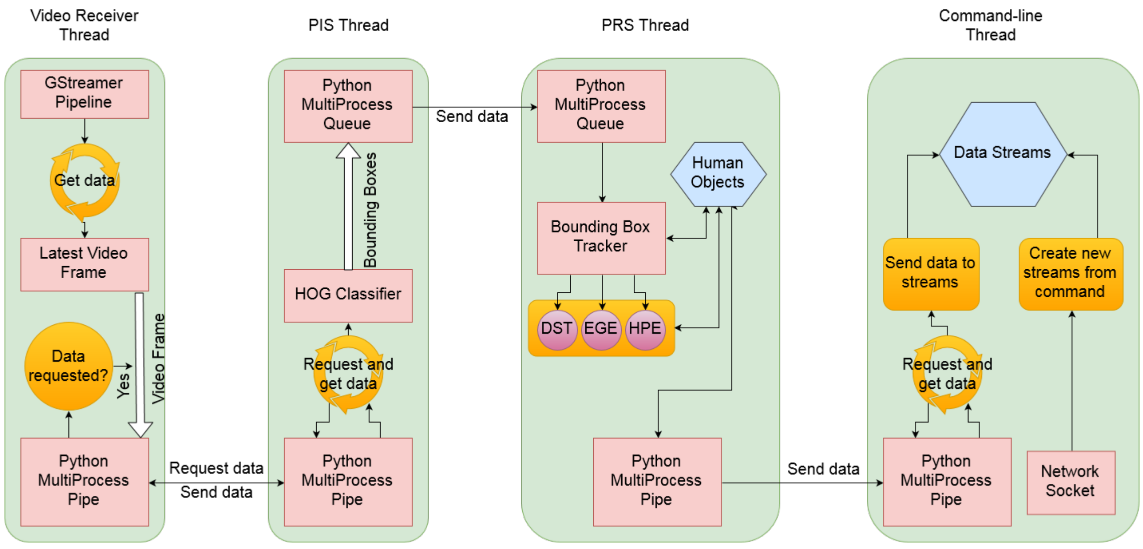


Figure 7.1: A highly abstracted depiction of the platform architecture and flow.

	Face detection	Tracking	EGE	DST	HPE	Total
Time	0.12745s	0.000085s	0.0032s	0.000009s	0.0021s	0.132844s
Complexity	$\mathcal{O}(1)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	- -

Table 7.1: Time consumption and complexity of major parts of the platform.

the major time consumer being the face detection. The table 7.1 shows the run-time costs and complexity of the main parts of the platform, where n is the number of people present in front of the camera. Note that these numbers are for $n = 1$, and were achieved on a PC with the following specifications:

- Processor: Intel Core i5, (8400)
  - Cores/Threads: 6/6
  - Speed: 2.8 GHz, 4 GHz with turbo
- Graphics card: Nvidia GeForce GTX 1060
  - Memory: 6 GB, (GDDR5)
- RAM
  - Type: DDR4
  - Memory: 8 GB, (2x4096MB)
  - Speed: 2666MHz

8
Discussion

8.1 Data labeling

When working with both the labeling and the data sets in this project, there were many realizations made, both around the validity of the method, but also around some pitfalls easy to fall into when working with labeling data, creating data sets from a purpose.

8.1.1 Difficulties with objectivity

The labeling in the project were done on images found online of people, either caught in a crowd or alone in front of a photographer. These images were labeled with either negative, for not having eye contact, or positive, for having eye contact, and the ones doing the labeling was us in the development team.

8.1.1.1 What is eye contact?

Since eye contact were such a normal and natural thing that everyone has experienced through out their lives, we naively thought we had the same idea of what constituted eye contact. This proved later to be wrong. One of us made most of their decisions purely on feeling meaning that if he did not feel the eye contact from the image, he labeled it as negative. The other one labeling did not however do it the same way. He also used the "feeling of eye contact" when labeling but he also did not trust his ability to detect eye contact a hundred percent. That meant he stopped at edge cases, evaluating them from the point of view of "Could this be interpreted as eye contact?" something that is vastly different from just disregarding the images when the feeling was not there. An example of an image that was disagreed on can be seen in fig. 8.1.

This produced an uneven labeling, where a lot of the times images would be labeled as positive by one, and negative by the other. If the people labeling some times could not agree on or decipher if something was eye contact or not, how can you expect the robot to?

8.1.1.2 Biases

Subconscious bias could possibly be a problem when labeling. Could it be that people smiling might be labeled as positive more often than people who were not? Was this because people often smile when looking into a camera? Or was it because

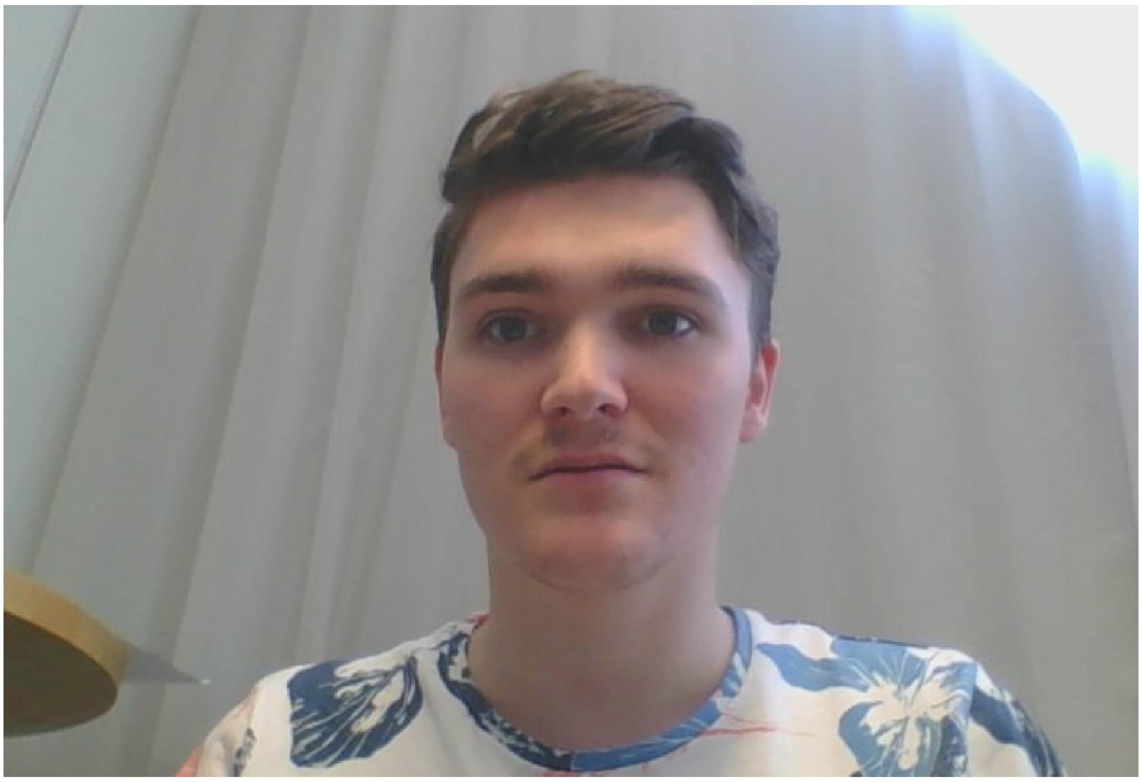


Figure 8.1: Example of an image where the people labeling would disagree.

the people labeling felt the eye contact more when it was accompanied with a smile, or maybe the person labeling just wanted this person to have eye contact just because they looked nice? Could the subjective attractiveness of the person being labeled, factor into the result? These are all things to be aware of if attempting this type of labeling again, and if it exists, it could be countered by involving more people in the labeling process.

8.1.1.3 Unfair advantage

The images that we as humans labeled were images of people, some only the face, some their whole upper body. When training the neural network however we cut out only the eyes accompanied with eyebrows. When we decided whether or not a person had eye contact with us, we had access to much more information than we let the computer train on. The computer could therefore be at a disadvantage as it is trying to decipher what constitutes eye contact, and in some images the feeling of eye contact might have been something that came from a part of the image that the computer does not have access to.

Maybe the most fair way of doing this would have been to do the labeling on just eyes and effectively putting us and the robot on an even playing field.

This all comes down to whether or not eye contact is something that plays out in the area of the eyes, or if you could get tricked into believing someone has eye contact with you by getting that information from somewhere else.

8.1.2 Relabel the data set?

As we discovered the flaws in the labeling described above, we encountered a difficult question: Should we relabel the data set, knowing what we know now? If that could be done we would cut out the eyes before labeling making it more fair for the robot as well as being aware of our biases. The time, mental strain and unpredictable result that we would get from this was deemed not worth it. Sure, relabeling it could grant us a better labeled data set, and it could in return end up increasing the accuracy of the neural net. The difficult thing is do know how much. If its only a change of 1% , is it worth it? Instead of relabeling a data set where most images probably already are correctly labeled, we could instead spend that time labeling new images, and just expanding the data set we have. That could also add to the generalization of the solution, which could be worth even more. Adding more images also dilutes the currently existing bias possibly fixing what you could hope to fix with relabeling.

8.1.3 Patching the data set

When discovering flaws in how the neural net works, we could decipher some clear edge cases where we thought it was clearly eye contact, and the neural net was 99% sure it was not. These cases were quickly identified to be not very well represented in the training images, as we could not recall labeling a single image that looked at the camera the way we did. If we never gave the computer such an image to learn from, how can we expect it to handle it correctly?

This issue led to discussions about patching the data sets with those types of images that we thought were missing in the training data. Intuitively this sounds like a good idea, just presenting the the neural net with these edge cases so that it could learn. The problem here was the time it could take, and how generalized we could get it to be.

8.1.4 Reflection

The labeling of the images was a tedious and time consuming process that could be made better in many ways. Although the labeling tool worked, it would probably have been a bit difficult to use if you were to involve more people in the labeling.

A clear and limited definition of eye contact when labeling was something that was never attempted and the disparity in the results between the two people labeling was found too late in the process to start over. For any future project that involves labeling data like this, identifying what the people labeling agree and disagree on would be a good idea. This could, for example, be done by letting them label a small subset of images, and then comparing the result, seeing what they agree on and where their differences lie.

Letting several people label data could show where differences are but it could also be incorporated in the labeling process, letting a bigger group of people label the same images and then taking the most agreed on label and using that as ground truth when training. Images with no clear majority on either label could be sorted out and removed.

Another way would be to create the data set, with eye contact in mind, could be to distribute an app which lets people take images and labeling them at the same time, by providing clear instructions on what the user should do for each image. This way, more people are involved in the labeling and you get information on how the data was obtained and basic information about the person in the images. This knowledge of distribution of ages, sexes and ethnicity could help in the evaluation, knowing how general the solution is and in what areas more data is needed.

8.2 Training

The training of the neural net was the biggest question mark when going into the project. Whether or not it would be able to get anywhere with the data we provided, and how long time the training would take were questions that we did not know how to answer. If we were lucky we would get some promising results early, showing that the training actually worked and got better as the dataset got bigger. We got promising results but it raised some questions that we will bring up below.

8.2.1 Data set bias

The first sign you look for when evaluating whether or not the neural net is properly built is that when training, the neural net gets better accuracy, and stops guessing randomly. This you can see when the accuracy steadily rises above 50%. Very early in the project the accuracy rose to around 80%. That, as we discovered later, falsely told us that the neural net got better at differentiating the images. When testing the model on a batch of images, it estimated them all as negative, without exceptions. What the neural net had found was the bias that were built into the data set. We trained the neural net on more than twice as many negative images than positive. This meant that just guessing negative on each image gave the neural net an estimation higher than 50%.

8.2.2 Changing the hyper parameters

When doing different iterations of the training we had different parameters we wanted to try out. The auto-queuer (see section 5.4.1) helped us try these out in an effective way, where we could queue training over the weekend or over night, and then just analyze the results after. This approach to training, instead of manually constructing the different test cases, suited our work very well, as we were unsure of what parameters and what values that would prove to be important when solving this problem. To just try them in bulk, and then letting the results tell us whether or not a change had a positive effect or not let us focus on other thing whilst training. As we also had different batches of data sets, meaning we kept each new bulk of labeled images in separate folders, we could also test out different data set combinations, and also evaluate how much each data set contributed to increase the accuracy or decreasing the loss.

8.2.3 Identifying success

When doing the training and looking at the results of testing the accuracy on the validation set and getting an accuracy around 93% you easily think that that is good. After all it did successfully distinguish eye contact in over 90% of the images in the validation set when tested on around 1000 images. The mistake is assuming that the validation set reflects reality enough. It might be very good at a certain type of images and perform very poorly on others. When tested using images we took with our webcam, we could quickly see where it fails. The truth is that the validation and training data just does not challenge the neural net enough during training and the edge cases are far and few between. And as the validation images is just cut out of the same set that it trains on, it inherits the same flaws that the training set has.

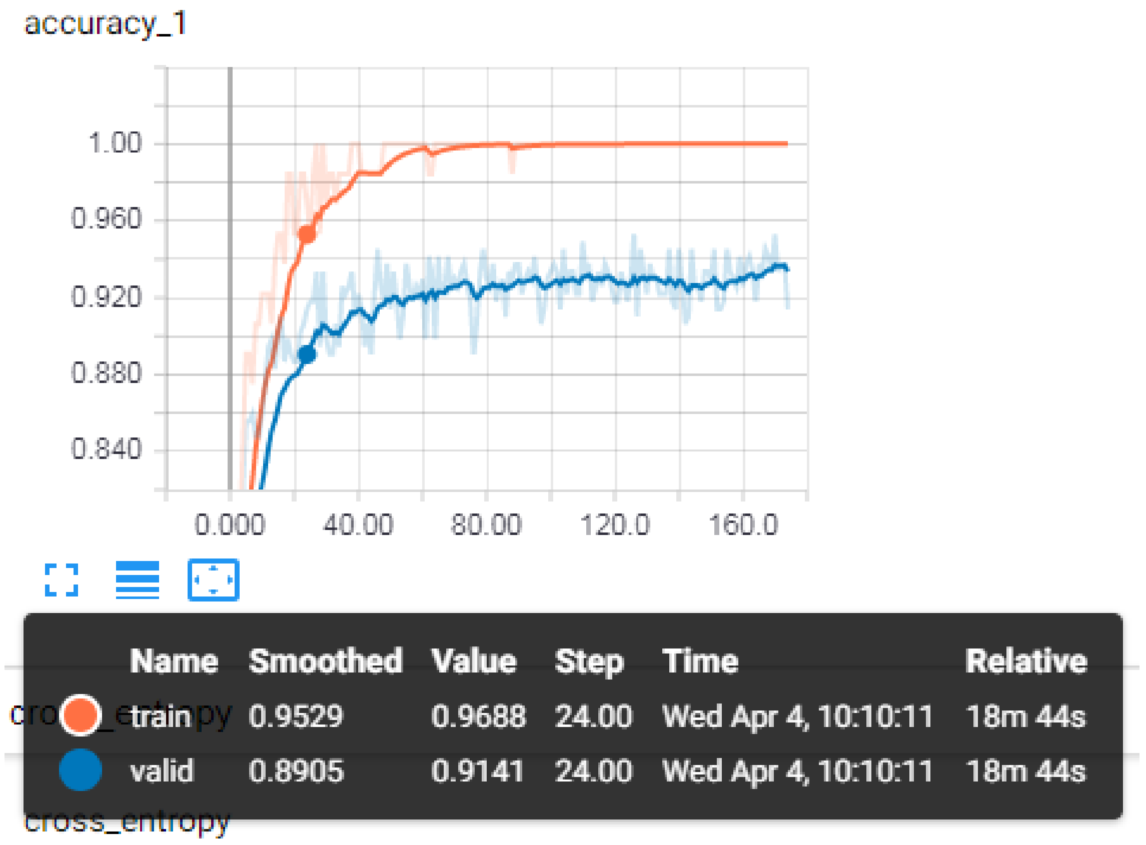
8.2.4 Data augmentation

Data augmentation was used to increase the size of the data set and make it more diverse. In hindsight however, more could have been done to increase the size of the data set with this technology. The data augmentation that were explored were horizontal flipping (mirroring the image), adjusted contrast (producing an image with a different contrast) and vertical flipping (flipping the image upside down). The vertical flipping was not really used, as it did not feel like a true representation of reality as no one would look at the robot upside down. This fear of images not representing reality in combination with us being unsure what different augmentation techniques could do to the “eye contact” in the image stopped the pursuit of more ways of augmenting the data. After the user tests we identified a range of problems that could be solved by having more data. Some of the augmentation techniques that could be used to make the data set more diverse are listed below.

- *Rotating image*
  Rotating the image 10° to either side, effectively making the data set three times larger. Since eye contact detection appears to be worse when tilting your head, this could help with that scenario.
- *Changing brightness*
  Changing the contrast and brightness of the image in more ways than one, which is how it is being handled in the current implementation. Instead you could have several different contrasts in a sliding scale giving many versions of the same image.

8.2.5 Planning

It was difficult estimating how long the training would take. Based on the research and seeing other projects (such as mnist[21]) successfully using CNNs to classify images with low quality, we felt confident that the approach was valid. It was however hard to estimate how long it would take. How long would a training session take, that depends on how many images and what resolution the images are, how many training steps and what learning rate you choose. The number of images was especially difficult to foresee, as we knew more is better but we did not know how many you would need to make it “good”. If we would have had hundred of thousands of images, how long would a session take then? Luckily the iterative process of training, evaluating the training result and then labeling some more images gave us some early indicators on that the algorithm and the approach was working. Each time we trained on a new batch of images we saw an increase in performance. Analyzing both our own files of data from each session along with the graphs generated with the visualizing tool TensorBoard[56] (see fig. 8.2) the progress of the training was easy to follow.



	Name	Smoothed	Value	Step	Time	Relative
●	train	0.9529	0.9688	24.00	Wed Apr 4, 10:10:11	18m 44s
●	valid	0.8905	0.9141	24.00	Wed Apr 4, 10:10:11	18m 44s

Figure 8.2: Graph generated with TensorBoard[56], showing the increase in accuracy over iterations. The blue line is the accuracy on the validation set, and the orange is for the training set.

8.3 The platform

In section 1.3 Aim, we state that “the goal is to develop a platform for the robot on which human body language interpretation modules can be deployed, and to develop at least one such module. The idea is that the platform can be easily extended with more modules, each interpreting one human social cue, and that the modules will collaborate to determine the likelihood of a nearby human’s willingness or intention to engage in conversation with the robot”.

This vision was kept in mind throughout the project, and the final implementation accurately reflect that. The estimation modules on the platform are kept completely separated from each other, a feature that can help facilitate the addition and testing of new modules. The parts that make up the platform itself are also highly decoupled from each other, which should be a huge help for any future endeavours to improve it. For example, as can be seen in table 7.1, one frame currently takes approximately 0.133 seconds (resulting in about 7.5 frames per second), and the face detection is responsible for almost all of that time. If a faster solution can be found, replacing the HOG currently in place would be very straightforward.

Similarly, in the future it might be interesting to add additional types of input data, such as audio. Doing so would only require some changes to the PIS (see fig. 7.1) and quite possibly no changes to any of the other components. Despite this, there are some flaws, both large and small, to the platform.

8.3.1 Security

Software residing on internet-connected computers ought to always take security seriously. Since this project is only of a prototype, that aspect was never considered during development. It is likely that the prototype contains multiple security issues, which of course means that it should not be deployed to any real-world situations.

8.3.1.1 Multiple users

One problem that exists currently is the lack of any sort of real user separation on the platform. Users requesting streams of data specify a port on which the data should be streamed, and the platform starts streaming on that port, accepting any and all connections. Two users requesting data, even if it is the same data, on the same port is currently undefined behaviour, and will likely crash the system. Anyone else can also listen in on the streamed data, simply by connecting to that port, which may not be desirable.

8.3.2 Performance

As can be seen in table 7.1, the platform is unlikely to scale, especially with higher resolutions which may be desirable in the future. While the face detection is the major weak point right now, future estimation modules may be more demanding than the three presented here. It should therefore be noted that the estimation modules are currently called sequentially, which may cause large slowdowns if a more demanding module is present.

Ideally, the whole solution should be rewritten in a faster language with better support for threads and asynchronous function calls, such as C++. While Python has proved very useful for rapid prototyping, its threading and multiprocessing facilities are lacking, and the overall performance of the language is, naturally, not as good as that of a compiled language. In hindsight, while it is generally slower to implement a solution in C++ than in Python, due to Python's higher level abstractions and dynamic type system, the easier-to-use threading library in C++ could possibly have made a C++ implementation process quicker.

8.3.3 Nitpicks

Some sub-components of the platform are unfinished, due to a lack of time and importance. While this is to be expected, they are nevertheless presented here for completeness.

Streaming video from the platform to a client is currently done via Python's network sockets, over TCP. Ideally, GStreamer should be used here too, but due to the difficulty of using it correctly in Python code (and the general lack of documentation), the attempt that was made was abandoned. The video out streaming is therefore probably nowhere near as fast as it could be.

The command-line parser is currently in a very unfinished state, and will not scale well when adding more commands. The syntax used in the commands has some hard requirements that are not evident to the user, there is no help or feedback on what went wrong if the command is syntactically wrong (except the message "Unknown command"), and it is overall not a nice interface.

8.4 User tests

8.4.1 Potential flaws in the test

During the user tests, we discovered a number of flaws or problems in the procedure, which are outlined and discussed below.

1. *Test subjects were instructed not to move their heads*
   While this parameter helps isolate the testing to the EGE only, the test subjects had obvious difficulties with it. How well they managed to follow this instruction also varied between subjects, which may have skewed the results somewhat. In retrospect, it might have been better to let the test subjects turn their heads however they liked, producing more natural results as well as providing more data for the HPE module.

2. *It was not clear, from the markings on the floor, exactly where test subjects should stand*
   The initial idea was that test subjects should stand behind the markings, but during pretesting and initial tests it was made clear that people naturally felt they should stand on the markings. The instructions were therefore changed to allow people to stand on the markings (none of the test results are from people standing behind the markings, they all stood on them). However, this change removed quite a bit of exactness to the tests, as test subjects stood at various positions on the markings. To fix this, the markings could have been in the shapes of shoes to clearly communicate where test subjects should stand.

3. *The group tests were very difficult for the participants*
   Instructing three people in sequence to assume a position, head pose and eye gaze direction, and then maintaining that pose proved very difficult. Oftentimes the one first instructed would have lost some of their pose by the time the last person had been instructed, even though it only took a few seconds. This may be fixable with more thought through scenarios and instructions, but it might be better to rework the test from the ground up into something more natural to the test subjects (see point 3 in 8.4.2 below).

4. *Imbalanced number of men and women*
   Out of 66 test subjects, only 13 were women, which is, of course, not an accurate representation of the overall population for which the prototype is intended to work. Imbalances are likely to arise in one way or another, be it age, gender or ethnicity, but doing more tests, and especially in different places, would probably help to lessen them.

5. *Distracting elements in the test*
   During the tests we tried to reduce the number of distracting elements as much as possible, by placing the testing area in a place where people would not walk into the picture or behind the camera as much. We did however not consider the possibility of the screens being distracting, since there were moving faces

as numbers on them. The screens were more designed to illustrate the idea, bit might not have been optimal for the type of test we conducted.

8.4.2 Additional tests and parameters

In addition to the flaws described above, we also identified a number of extra parameters and scenarios that could be worth investigating in future tests.

1. *Height of the camera*
   The camera was set at the same height throughout the test. It could prove valuable to test the accuracy of the EGE from different viewing angles for the robot, and get some data of how well this would work on different sizes of robots.

2. *Having more markings*
   During the tests we used six markings on the floor as indicators for where the test subjects should stand. There might be something to gain from having more indicators, or more variation to their relative positions, creating more extreme cases for the three estimation modules.

3. *More natural tests*
   Creating tests that feels more like natural situations and less like laboratory tests could have given us more and better feedback on the concept. This would have produced an amount of data that we would not have time to go through, but could have given some very good insights in how the system works in a natural setting and what people thought about it. Such a test may be constructed by saving videos of the tests, and continuously streaming data to a file that correspond to the video. This would quickly result in massive quantities of data to analyze, but it would allow test subjects to behave in ways that are both more natural to them, and more realistic with regards to the situations that the prototype should actually work in. This was how the user tests were supposed to work when planned in the beginning of the project(see section 4.1.4), and strides towards this type of testing was done creating a graphical interface with a smiley face acting as the friendly robot face.

4. *Changing the lighting condition*
   Another factor that we did not test was different lighting conditions. This would also have added more data to go through and another factor to consider. However, when testing how generalized the solution is, different lighting is important to accurately assess how it will perform in different environments.

8.5 Test results

This section discusses the results of the user tests, the conclusions drawn, and possible solutions and improvements for future implementations.

The results highlighted in section 7.3 indicate that the approach taken in this project is possible, though the prototype has a number of flaws and require quite a bit of polish. Further work on this approach, with better weighting of modules, deeper training of the EGE and a better solution for accurately determining distance should yield a system that is very capable of automatically engaging people who want to have a conversation.

8.5.1 Eye gaze estimation

7.2.1 lists a number of conclusions, 1 to 6, drawn from the test data regarding the EGE. The list is reproduced, though shortened, below for convenience. For each conclusion, we analyze the possible causes and give potential solutions for each of them.

1. *The detection of actual eye contact works.*
   To draw this conclusion accurately for a binary classifier, the classifier must of course be equally good at classifying both the positive and negative cases correctly. Looking at the data and comparing the *far left* and *left* cases to when test subjects looked directly at the camera, there is a clear trend: the closer to the camera the test subjects looked, the higher the estimation, which is exactly what we expected. Point 3 below discusses the peculiar discrepancies between the left and the right side tests.

2. *Glasses are a problem.*
   This problem has a few possible causes, and is probably the result of some combination of them:
   - The data set used during training contains too few images of people with glasses.
   - The lighting conditions in the testing area were such that the camera had trouble seeing through the glasses due to reflections.
   - People tend to tilt their head forward somewhat which, in combination with eyeglass-rims that do not extend upwards very much, puts their eyeglass-rims right past the pupils from the camera's perspective.

   If the data set lacking in diversity is the main cause of the problem, the problem is easily fixed by labeling more images of people with glasses, and continue the training with them. If, however, the lighting conditions and the glasses themselves cause problems for the camera, so that crucial details are lost, the issue becomes more difficult. If that is the case, a brute force solution with more data might not work. Instead, a work-around could possibly be made by first estimating how visible the eyes and surrounding details are (i.e. is the person wearing glasses?), and then enhance the image by brightening it and/or increasing the contrast until the details become visible. If that still doesn't work, the weight of the EGE may need to be adjusted downwards for that person, letting other estimation modules (possibly invented for this specific case) take care of it in some other way.

3. *The solution works better for people looking to the left of the camera than those looking to the right of the camera.*
   This is a very strange problem, with many possible explanations. It may be that the test setting was skewed in some way, either in terms of lighting conditions, camera angle, distances between the targets the subjects looked at, or some combination of all of the above. It may also be that the data set contains a very uneven distribution of people looking to the left or to the right. Further testing should make it possible to reveal the underlying cause but, no matter the cause, the solution will probably be to increase the size and variance of the data set and retrain the neural network.

4. *Looking a little above or below the camera is detected as eye contact.*
   This mainly indicates a deficiency in the data set, and it should be solvable with more training. By collecting images specifically of people looking slightly above or below the camera, the deficiency can be corrected. However, it is possible that number 4 is not actually a problem, but rather an unintended feature; not all robots have their cameras mounted in their eyes, but rather slightly above or below the eyes. As such, looking directly into its eyes may lead to false negatives if the deficiency is corrected. Further testing is required to accurately measure the limits of the angles above and below the camera where the false positives exist.

5. *Subject age is not a problem.*
   While the test subjects mostly fell in the range 26-64 years old (see fig. xyz for age data), within this range, there were only some minor differences in EGE values. This makes sense, since most of the images labeled were also of people in the same age range, but it is a positive result nonetheless.

6. *Subject height is a problem, taller people get, on average, higher estimations when they are further away from the camera.*
   This is strange in that the problem is more pronounced when the test subject is further away from the camera. If the variance in head pose is what causes the problem (i.e. taller people have to tilt their heads more to look directly at the camera), which is the most likely cause, the problem should actually be more pronounced at close range. However, it may be that the same error exists at close range, but is overshadowed by the fact that the EGE is much more accurate at close range. Therefore, one possible solution would be to increase the resolution of the video stream. Another solution would be to find more images of tall people, expand the data set with those, and continue training the model.

The problems present in the current model have some different possible causes. While the problem with glasses is a bit more complex than the others, an improvement to the variance and size of the data set should greatly increase the performance and accuracy of the model across the board. This is true for most machine learning models, and our training set of roughly 45000 images (after augmentation) is not very large when compared to some other sets. Although not explicitly tested during the user tests, the time aspect of the EGE was observed to both working filtering out short accidental eye contact. It was however evaluated in how well it performed or how natural it felt to the users.

8.5.2 Distance estimation

While the test results show that the distance estimation module, in its current form, definitely works as intended, it is also evident that it is very crude. The current implementation only reports one of a few possible values, and it is quite possible for the test subject to stand on the border between 0.28 and 0.38 estimations, switching between them without actually moving and without there being any numbers in between the two. The implementation, as it is, also has problems with the values reported being almost entirely arbitrary; the sigmoid function used is tweaked to evaluate to 1.0 at one meter and then drop off rather quickly, but the numbers have very little to do with the real-world distances. While the values do get lower at further range, the minimum possible value is around 0.28, due to the HOG not being able to detect people at further distances at the set resolution.

Using distance as a small part of the CIPE equation makes sense, both as an immediate way to lower the total estimation for people further away, as well as a way to dynamically adjust the weights of the other modules. In order to use it with any real confidence however, the data should come from specialized hardware, such as radar or stereoscopic cameras, that can accurately measure real-world distances. Fortunately, adding support for such data to the platform should prove to be a straightforward exercise.

8.5.3 Head pose estimation

The HPE module was, we feel, not adequately tested during the user test session, and so the conclusions drawn may be wrong due to too small a sample size. Nevertheless, observations made during the test indicate that the HPE module, in its current state, is either unfair, wrongly calibrated, or just inaccurate. As can be seen in fig. C.6, the HPE rarely went over 0.6, even when test subjects were instructed to face the camera directly. From the images captured during the test, we can see that most people actually do have some natural roll and/or pitch of their head, even when they are instructed to face the camera (they may in fact think that they hold their head perfectly straight). However, none of the test subjects appear to have any yaw to their head pose. While no definite conclusions could or should be drawn from this, it indicates that the HPE's main problem could lie with its weights of the three axes of rotation. Adjusting the weights, so that yaw is almost the sole arbiter, might make the whole estimation more realistic.

It is also possible that the implementation, and the techniques used, is just not accurate enough. Internal testing showed that there exists significant difficulties with determining pitch accurately, for example. Indeed, the solution is based on finding certain facial landmarks and determine the head pose from the relative distances between them. While it could possibly be improved further, there might also exist completely different approaches that may be more worthwhile, such as another neural network.

8.6 Future work

The platform is built to simplify the process of adding more estimation modules. During the research stage there were a few different modules discussed and evaluated as possible additions. These are listed and discussed here to help any future attempts to further refine the system.

Apart from head pose, a persons body language consists of many small parts, such as the direction of their torso (similar to head pose) and how their feet are positioned. Each of these, and more, are candidates for estimation modules, and could contribute a small part of the whole. However, they do require that the PIS be improved to detect whole bodies rather than just faces, which is actually harder to do.

There are also some things to analyze with regards to mouth movement, identifying smiles and people opening their mouth to start speaking. Identifying that someone, who is already determined to be interested in conversation, is opening their mouth could be used to make sure a robot is not interrupting or trying to talk at the same time as that person.

9
Conclusion

In this paper we have shown how a robot can detect and analyze, with varying performance and accuracy, three different human behaviour patterns. Eye contact seem to be the best indicator of a human's interest in conversation, while distance and head pose contribute in smaller degrees. We further show how each of these three modules can, in concert on a common platform, work to deduce, out of multiple people in front of the camera, who is most interested in engaging in conversation with the robot.

Our results show that a correctly trained convolutional neural network is capable of accurately detecting eye contact in multiple test subjects in real-time, on a mid-range gaming PC, at a distance of 1.5 meters from a video stream with a resolution of only 640×480 pixels.

The distance estimation module implemented here is not very good, and it is our conclusion that the software-based solution used is not adequate for the task. It is our recommendation that the distance estimation module is built upon specialized hardware, such as radar or stereoscopic cameras, instead. Similarly, our head pose estimation implementation is lacking in accuracy and reliability, but it should be possible to vastly improve it by using multiple cameras and interpolating between the results.

Bibliography

[1] "MIT cheetah robot lands the running jump," February 2018. [Online]. Available: https://www.youtube.com/watch?v=_luhn7TLfWU

[2] M. Taylor, "Toyota's violin-playing robot," February 2018. [Online]. Available: https://spectrum.ieee.org/automaton/robotics/robotics-software/toyotas_violinplaying_robot

[3] E. Strickland, "Autonomous Robot Surgeon Bests Humans in World First," January 2018. [Online]. Available: https://spectrum.ieee.org/the-human-os/robotics/medical-robots/autonomous-robot-surgeon-bests-human-surgeons-in-world-first

[4] L. Nummenmaa and A. J. Calder, "Neural mechanisms of social attention," *Trends in Cognitive Sciences*, vol. 13, no. 3, pp. 135 – 143, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1364661309000242

[5] J. Minford and J. Lau, *Classical Chinese Literature: An Anthology of Translations*, ser. Classical Chinese literature : an anthology of translations / ed. by John Minford. Columbia University Press, 2000, pp. 232–233.

[6] *Leonardo's Programmable Automaton and Lion.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 21–68. [Online]. Available: https://doi.org/10.1007/3-540-28497-4_2

[7] "ABB - Industrial Robots," February 2018. [Online]. Available: http://new.abb.com/products/robotics/industrial-robots

[8] "Robot Dragonfly," February 2018. [Online]. Available: http://techject.com/robot-dragonfly/

[9] D. Muoio, "Japan is running out of people to take care of the elderly, so it's making robots instead," February 2018. [Online]. Available: http://www.businessinsider.com/japan-developing-carebots-for-elderly-care-2015-11op=1&r=US&IR=T&IR=T

[10] M. A. Goodrich and A. C. Schultz, "Human-robot interaction: A survey," *Found. Trends Hum.-Comput. Interact.*, vol. 1, no. 3, pp. 203–275, Jan. 2007. [Online]. Available: http://dx.doi.org/10.1561/1100000005

[11] K. Capek and D. Wyllie, *R.U.R (Rossum's Universal Robots).* Echo Library, 2010. [Online]. Available: https://books.google.se/books?id=cJ-jOWPstjkC

[12] I. Asimov, *I, Robot*, ser. Robot series. Bantam Books, 1950. [Online]. Available: https://books.google.se/books?id=MD0GAQAAIAAJ

[13] W. Wallach, "Android ethics: Bottom-up and top-down approaches for modeling human moral faculties," 01 2005.

[14] A. M. TURING, "I.—computing machinery and intelligence," *Mind*, vol. LIX, no. 236, pp. 433–460, 1950. [Online]. Available: http://dx.doi.org/10.1093/mind/LIX.236.433

[15] D. Feil-Seifer and M. Matarić, *Human-Robot Interaction.* New York, NY: Springer, April 2009, pp. 4643–4659.

[16] M. G. DYER, "Intentionality and computationalism: minds, machines, searle and harnad," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 2, no. 4, pp. 303–319, 1990. [Online]. Available: https://doi.org/10.1080/09528139008953728

[17] M. M. Adankon and M. Cheriet, *Support Vector Machine.* Boston, MA: Springer US, 2009, pp. 1303–1308. [Online]. Available: https://doi.org/10.1007/978-0-387-73003-5_299

[18] K. L. Priddy and P. E. Keller, *Artificial Neural Networks: An Introduction (SPIE Tutorial Texts in Optical Engineering, Vol. TT68).* SPIE- International Society for Optical Engineering, 2005.

[19] E. Reingold, "Artificial Neural Networks, What They Are," February 2018. [Online]. Available: http://psych.utoronto.ca/users/reingold/courses/ai/nn.html

[20] "THE MNIST DATABASE of handwritten digits," February 2018. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[21] "Deep MNIST for Experts," February 2018. [Online]. Available: https://www.tensorflow.org/versions/r1.1/get_started/mnist/pros

[22] "PyTorch," May 2018. [Online]. Available: https://pytorch.org

[23] "Caffe," May 2018. [Online]. Available: http://caffe.berkeleyvision.org

[24] "Tensorflow," April 2018. [Online]. Available: https://www.tensorflow.org/

[25] "dlib C++ Library," April 2018. [Online]. Available: http://dlib.net/

[26] M. Flasiński, *Introduction to Artificial Intelligence*, 1st ed. New York, NY, USA: Springer-Verlag New York, Inc., 2016.

[27] R. Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed. New York, NY, USA: Springer-Verlag New York, Inc., 2010.

[28] C. Shi, M. Shiomi, T. Kanda, H. Ishiguro, and N. Hagita, "Measuring communication participation to initiate conversation in human–robot interaction," vol. 7, 01 2015.

[29] S. Satake, T. Kanda, D. F. Glas, M. Imai, H. Ishiguro, and N. Hagita, "How to approach humans?-strategies for social robots to initiate interaction," in *2009 4th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, March 2009, pp. 109–116.

[30] C. Rich, B. Ponsler, A. Holroyd, and C. L. Sidner, "Recognizing engagement in human-robot interaction," in *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, March 2010, pp. 375–382.

[31] S. Asteriadis, K. Karpouzis, and S. Kollias, *The Importance of Eye Gaze and Head Pose to Estimating Levels of Attention*, 06 2011, pp. 186 – 191.

[32] A. Sorokowska, P. Sorokowski, P. Hilpert, K. Cantarero, T. Frackowiak, K. Ahmadi, A. M. Alghraibeh, R. Aryeetey, A. Bertoni, K. Bettache, S. Blumen, M. Błażejewska, T. Bortolini, M. Butovskaya, F. N. Castro, H. Cetinkaya, D. Cunha, D. David, O. A. David, F. A. Dileym, A. del

Carmen Domínguez Espinosa, S. Donato, D. Dronova, S. Dural, J. Fialová, M. Fisher, E. Gulbetekin, A. H. Akkaya, I. Hromatko, R. Iafrate, M. Iesyp, B. James, J. Jaranovic, F. Jiang, C. O. Kimamo, G. Kjelvik, F. Koç, A. Laar, F. de Araújo Lopes, G. Macbeth, N. M. Marcano, R. Martinez, N. Mesko, N. Molodovskaya, K. Moradi, Z. Motahari, A. Mühlhauser, J. C. Natividade, J. Ntayi, E. Oberzaucher, O. Ojedokun, M. S. B. Omar-Fauzee, I. E. Onyishi, A. Paluszak, A. Portugal, E. Razumiejczyk, A. Realo, A. P. Relvas, M. Rivas, M. Rizwan, S. Salkičević, I. Sarmány-Schuller, S. Schmehl, O. Senyk, C. Sinding, E. Stamkou, S. Stoyanova, D. Šukolová, N. Sutresna, M. Tadinac, A. Teras, E. L. T. Ponciano, R. Tripathi, N. Tripathi, M. Tripathi, O. Uhryn, M. E. Yamamoto, G. Yoo, and J. John D. Pierce, "Preferred interpersonal distances: A global comparison," *Journal of Cross-Cultural Psychology*, vol. 48, no. 4, pp. 577–592, 2017. [Online]. Available: https://doi.org/10.1177/0022022117698039

[33] A. Strupczewski, "Commodity camera eye gaze tracking," Ph.D. dissertation, WARSAW UNIVERSITY OF TECHNOLOGY, Warsaw, 2016.

[34] S. Baluja and D. Pomerleau, "Non-intrusive gaze tracking using artificial neural networks," Pittsburgh, PA, USA, Tech. Rep., 1994.

[35] X. Zhang, Y. Sugano, and A. Bulling, "Everyday eye contact detection using unsupervised gaze target discovery," in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '17. New York, NY, USA: ACM, 2017, pp. 193–203. [Online]. Available: http://doi.acm.org/10.1145/3126594.3126614

[36] "Matthews correlation coefficient," February 2018. [Online]. Available: https://en.wikipedia.org/wiki/Matthews_correlation_coefficient

[37] H. Sharp, Y. Rogers, and J. Preece, *Interaction Design: Beyond Human Computer Interaction.* John Wiley & Sons, 2007.

[38] B. Meyer, *Agile!: The Good, the Hype and the Ugly.* Springer Publishing Company, Incorporated, 2014.

[39] G. van Rossum, "PEP 8 – Style Guide for Python Code," February 2018. [Online]. Available: https://www.python.org/dev/peps/pep-0008/

[40] Y. Wadsworth, *Do it yourself social research / Yoland Wadsworth*, 3rd ed. Allen & Unwin Crows Nest, N.S.W, 2011.

[41] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, October 2007.

[42] "NIST Special Database 18," May 2018. [Online]. Available: https://www.nist.gov/srd/nist-special-database-18

[43] "The BioID Face Database," May 2018. [Online]. Available: https://www.bioid.com/facedb/

[44] "Caltech 10, 000 Web Faces ," May 2018. [Online]. Available: http://www.vision.caltech.edu/Image_Datasets/Caltech_10K_WebFaces/

[45] "Description of the Collection of Facial Images," May 2018. [Online]. Available: http://cswww.essex.ac.uk/mv/allfaces/index.html

[46] "FEI Face Database," May 2018. [Online]. Available: http://fei.edu.br/~cet/facedatabase.html

[47] "Psychological Image Collection at Stirling (PICS)," May 2018. [Online]. Available: http://pics.stir.ac.uk

[48] "Detect eyes, nose, lips, and jaw with dlib, OpenCV, and Python," April 2017. [Online]. Available: https://www.pyimagesearch.com/2017/04/10/detect-eyes-nose-lips-jaw-dlib-opencv-python/

[49] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 12 2014.

[50] L. B. i Rossell, "Task Spooler," May 2018. [Online]. Available: http://vicerveza.homeunix.net/~viric/soft/ts/

[51] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–511–I–518 vol.1.

[52] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *In CVPR*, 2005, pp. 886–893.

[53] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *CVPR*, 2014.

[54] S. Mallick, "Head Pose Estimation using OpenCV and Dlib," May 2018. [Online]. Available: https://www.learnopencv.com/head-pose-estimation-using-opencv-and-dlib/

[55] "Rodrigues' rotation formula," May 2018. [Online]. Available: https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula

[56] "TensorBoard: Visualizing Learning," May 2018. [Online]. Available: https://www.tensorflow.org/programmers_guide/summaries_and_tensorboard

A
Test Scenarios

A.1 Single scenarios

These scenarios are mainly used to test the robustness of the EGE-module, testing how it performs with different people doing the same tasks. It will also test how much the performance changes when the participant increases the distance to the camera.

1. Participant walks and stands in the green space on marking 1, facing the left screen not looking at the robot.
   (a) Participant looks at robot
   (b) Looks at A
   (c) Looks at B
   (d) Looks at C
   (e) Looks at D
   (f) Looks at E
   (g) Looks at F
2. Walks to blue marking looking at right screen.
   (a) Participant looks at robot
   (b) Looks at A
   (c) Looks at B
   (d) Looks at C
   (e) Looks at D
   (f) Looks at E
   (g) Looks at F

   Expected results:
1. The only positive eye contact noticed should be when the participant looks at the robot. The other markings will be placed close to the robot, but should not count as eye contact. If they do, this would be a false positive.
2. Same as 1, but we expect a different result due to the change in distance.

A.2 Group scenarios

Invite three people at a time to stand in each of the markings on the ground. They will each have to answer to the color/name in front of them (Red, Green, Blue). The different scenarios are as follows:

1. Each participant stands on the marking 1
   - Red stands facing the left screen, looking at the left screen.
   - Green stands facing the robot, looking at the left screen
   - Blue stands facing the right screen, looking at the right screen
2. Each participant remains on markings
   - Red looks at the robot
   - Green looks at the robot
   - Blue looks at the robot.
3. Red stands on marking 1, Green and Blue stands on marking 2
   - Red faces the robot, looking at robot
   - Green faces the left screen, looking at the left screen.
   - Blue faces the robot looking at the right screen.
4. Each participant remains on their markings.
   - Red looks at the robot
   - Green looks at the robot
   - Blue looks at the robot.

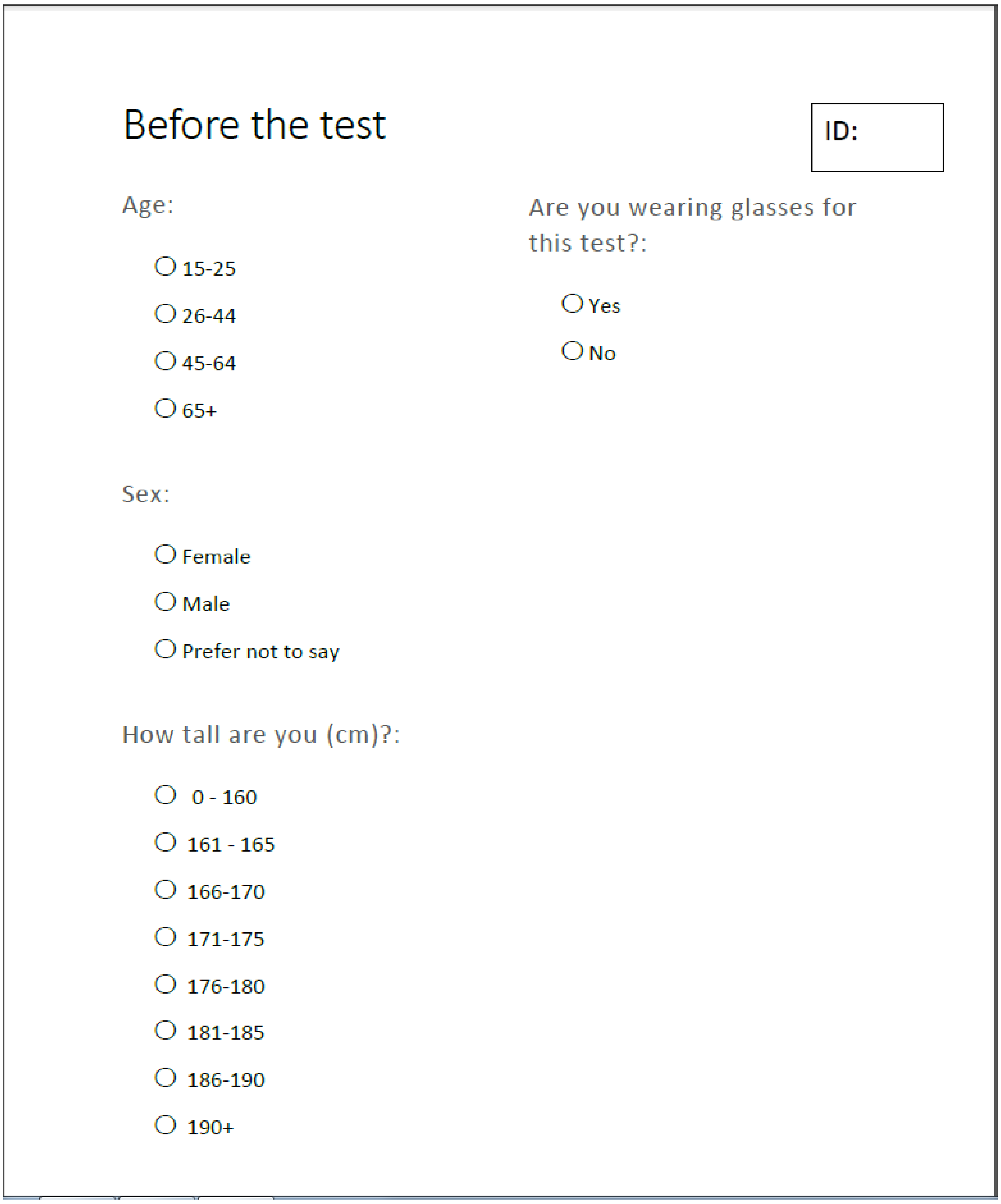
   Expected results:

1. Green should have a higher total estimation due to head pose, but still have a low estimation on eye contact. Red and blue should have an equally low total estimation.
2. Green should have the highest total estimation due to head pose and positive eye contact. Green and blue should have equally high total estimation and it should be positive for contact, but have a lower total estimation than green.
3. Red should have the highest estimation, despite being furthest away, as red has eye contact with the robot, and blue and green do not.
4. Red should have a lower total than both blue and green, due to being further away from the robot. Since green and blue are standing on the same distance, the deciding factor between them should be the head pose which should give blue a better total.

B
Test preparations

B.1 Form

Each participant got to fill out a form giving some details about themselves. This form can be seen in Fig. B.1



Before the test

ID:

Age:

- ○ 15-25
- ○ 26-44
- ○ 45-64
- ○ 65+

Sex:

- ○ Female
- ○ Male
- ○ Prefer not to say

How tall are you (cm)?:

- ○ 0 - 160
- ○ 161 - 165
- ○ 166-170
- ○ 171-175
- ○ 176-180
- ○ 181-185
- ○ 186-190
- ○ 190+

Are you wearing glasses for this test?:

- ○ Yes
- ○ No

Figure B.1: The form each participant filled out before the test

B.2 Manuscript

This manuscript consisted of a few bullet points of what information that the participants would receive before participating in the test.

- Images taken during this testing session will be used by us to evaluate and better the performance of the system but will not be published anywhere without your consent.
- You will remain anonymous.
- You may exit the test at any time without any repercussions.
- This test will work much like a photoshoot. We will give you instructions on how to stand, where to look and will essentially walk you through the whole test.
- The middle screen is the one containing the robot. This is also where the camera is located. When we ask you to look at the robot, that means that you should try to look into the camera
- The markings on the ground are of three colors: Red, green and blue.
- Before we start the test, we will assign you one of these colors, and ask you to stand on a specified marking for that color.

C
Graphs

This appendix contains a number of graphs that represent parts of the data presented in appendix D. These graphs are used to highlight various trends and behaviours that appear in the data.

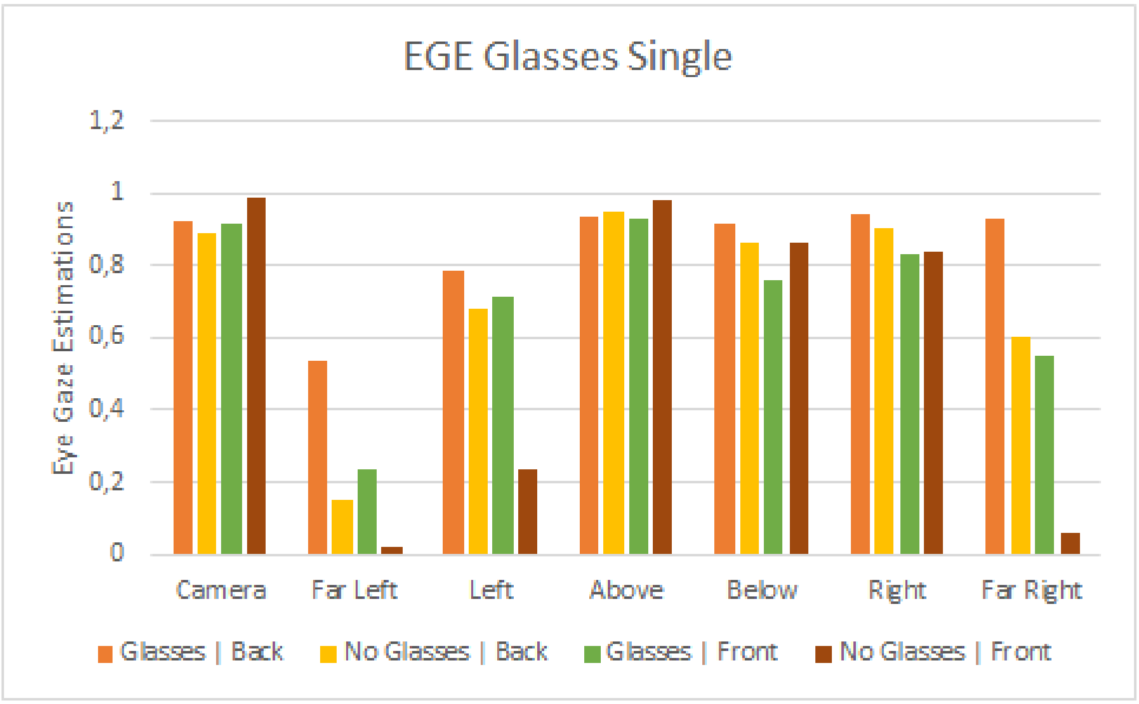


Figure C.1: Bar chart showing the different results of the EGE with changing factors being a) distance to the camera and b) if the participant was wearing glasses

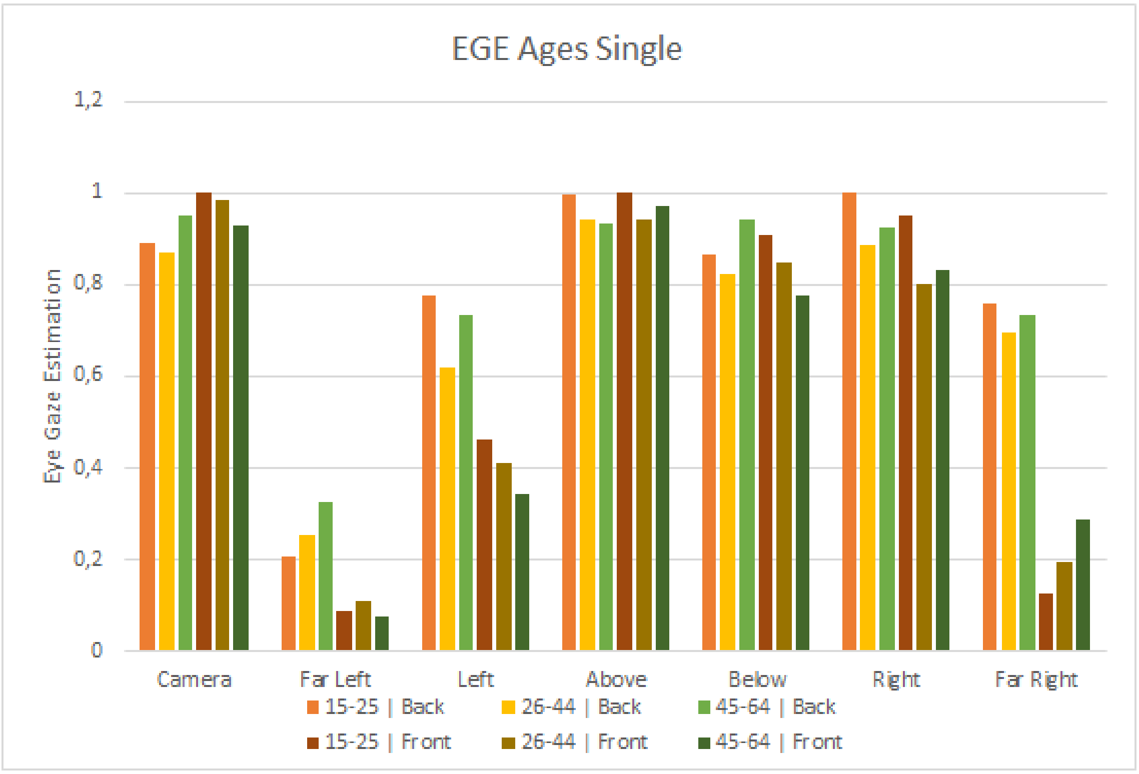


Figure C.2: Bar chart showing the different results of the EGE with changing factors being a) distance to the camera and b) the age of the participant

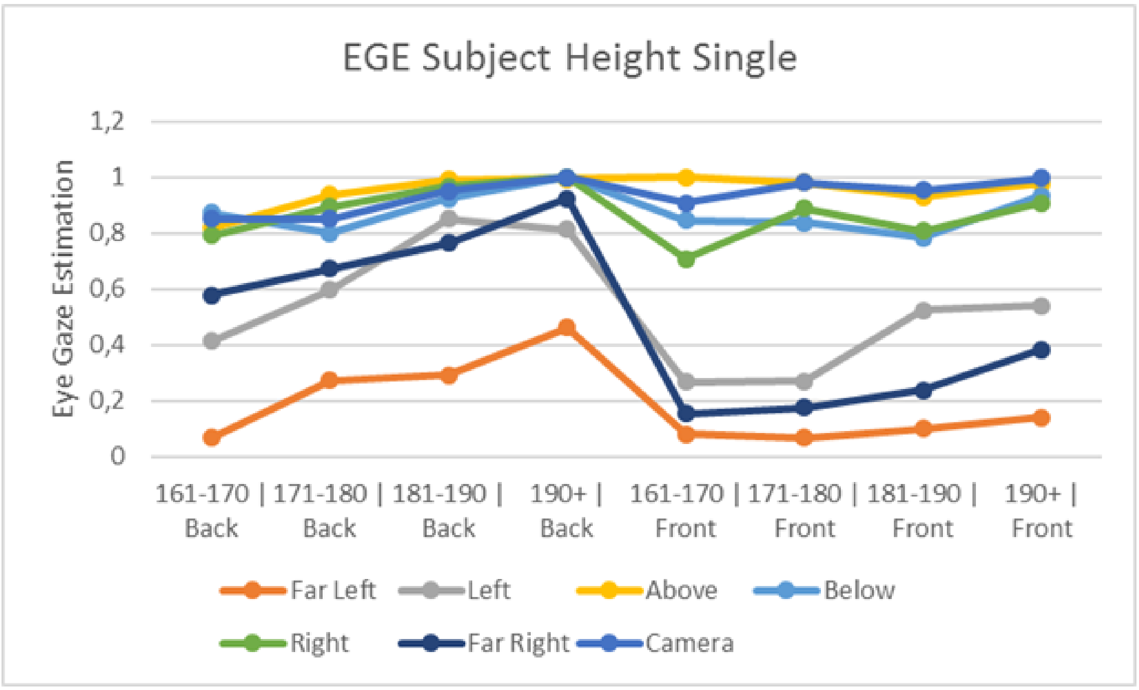


Figure C.3: A line graph showing the different results of the EGE with changing factors being a) distance to the camera and b) the height of the participant

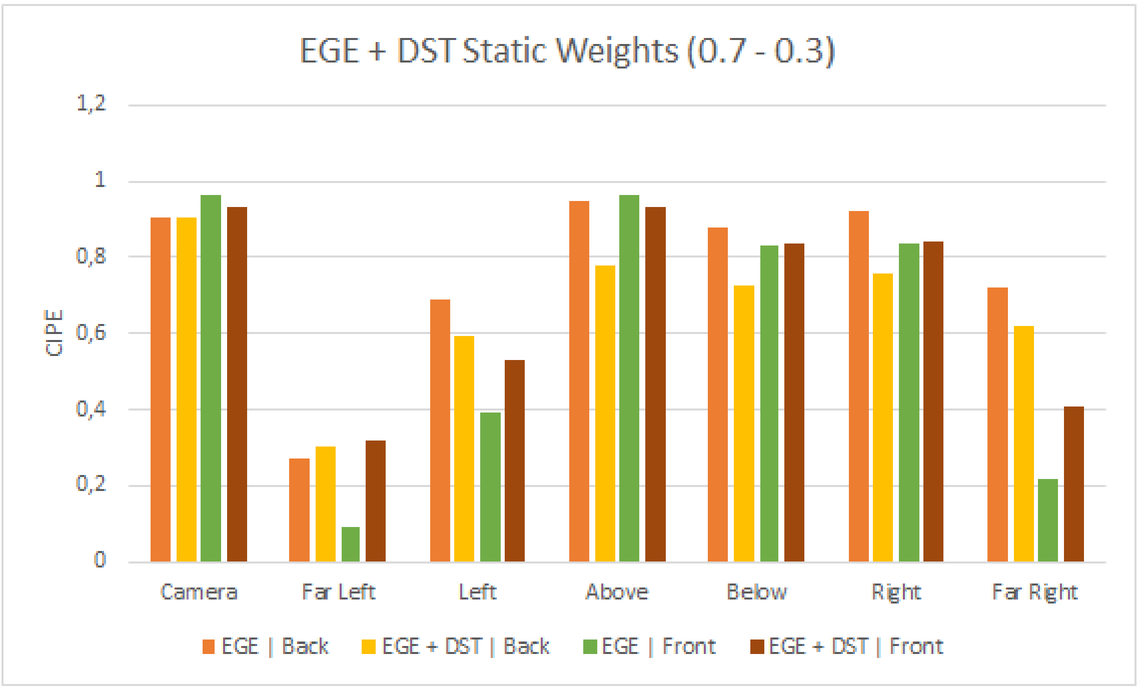


Figure C.4: A bar chart showing the different results of the CIPE with static weights.

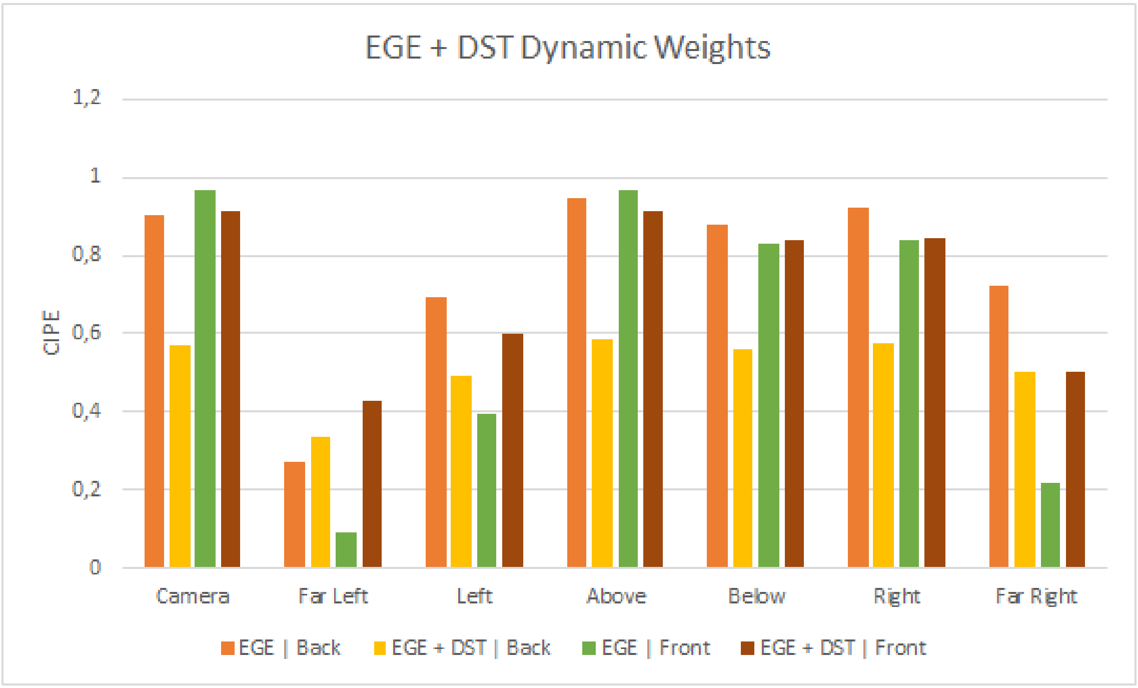


Figure C.5: A bar chart showing the different results of the CIPE with dynamic weights.

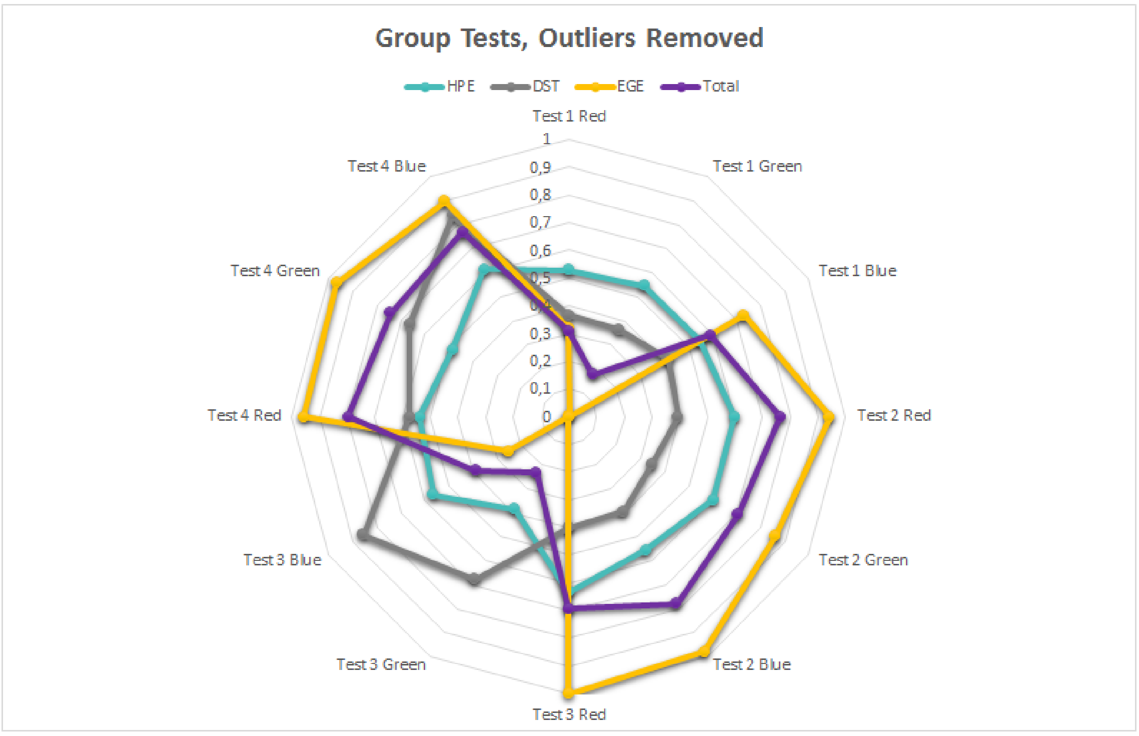


Figure C.6: Chart showing the performance of the three estimation modules in the group tests. The total is also shown and is equal to $EGE \times 0.6 + DST \times 0.2 + HPE \times 0.2$. Outlying data points that were more than 80% removed from the median are not part of this graph. Going through images of the test subjects, almost all of those points seem to be from people who somehow, without intention, corrupted the data. A number of them were people with eyeglass-rims going right past their pupils (due to the head tilting forward somewhat), others happened to blink just as the data was saved.

D
Data

This appendix contains the complete data gathered during the user tests. They appear in order: personal data (from form), single user tests, group tests. The single user results, of which there are seven, have two parts to each table: on the left is presented data for when the test subject was positioned at 2.5 meters from the camera, and on the right is data for 1.5 meters distance. In the group results tables, the rows are split in groups of three, corresponding to the test subjects' id:s and assigned colors (red, green, blue).

Table D.1: The data collected about our test subjects from the form. Each number (apart from the Id) correspond to indices in the possible answers for each question (e.g. a 2 on Glasses is a No).

Id	Age	Sex	Height	Glasses	Id	Age	Sex	Height	Glasses
1	3	2	6	1	34	2	2	6	1
2	2	2	5	2	35	2	1	1	1
3	3	2	6	2	36	3	2	6	2
4	1	1	3	2	37	2	2	6	2
5	1	2	6	2	38	2	2	7	1
6	2	2	6	1	39	3	2	5	2
7	2	2	6	1	40	2	2	5	2
8	3	2	4	1	41	1	2	7	2
9	3	2	7	1	42	2	2	5	2
10	3	2	7	2	43	1	1	1	2
11	2	2	8	2	44	2	2	6	2
12	1	2	7	2	45	1	2	1	2
13	2	2	5	2	46	3	2	4	1
14	1	2	5	2	47	3	2	6	2
15	1	2	5	2	48	2	2	5	2
16	3	2	5	2	49	2	1	3	2
17	1	2	7	2	50	2	1	4	2
18	2	2	6	2	51	2	2	4	2
19	2	1	2	1	52	3	2	8	2
20	2	1	1	2	53	3	2	8	1
21	2	2	6	1	54	3	1	3	2
22	3	1	2	1	55	3	1	5	2
23	3	1	4	2	56	3	2	6	1
24	1	2	8	1	57	2	2	4	2
25	2	2	5	2	58	2	2	5	2
26	2	2	7	1	59	2	2	7	1
27	3	1	2	2	60	2	2	3	1
28	3	2	7	1	61	2	2	6	1
29	3	2	5	2	62	2	2	4	2
30	3	2	6	1	63	3	2	4	2
31	3	2	6	2	64	2	1	2	2
32	3	2	8	2	65	2	2	8	2
33	2	2	4	1	66	2	2	5	1

Table D.2: Single test 1, left column is at 2.5 meters, right column is at 1.5 meters.

Id	CIPE	EGE	DST	HPE	Id	CIPE	EGE	DST	HPE
2	0,7577	0,9355	0,3831	0,7683	2	0,8894	0,9576	0,7508	0,8233
4	0,7658	0,9997	0,2796	0,5503	4	0,8551	1,0000	0,7416	0,5337
5	0,6552	0,7498	0,3831	0,6435	5	0,8713	0,9998	0,7416	0,6156
6	0,7790	1,0000	0,3752	0,5199	6	0,8607	1,0000	0,7416	0,5621
7	0,8642	1,0000	0,5302	0,7908	7	0,9822	1,0000	0,9857	0,9252
8	0,7447	0,9575	0,3752	0,4757	8	0,8423	0,8651	0,9202	0,6959
9	0,7768	0,8750	0,3752	0,8839	9	0,5171	0,2506	0,9158	0,9179
10	0,7792	1,0000	0,3831	0,5131	10	0,9046	1,0000	0,9112	0,6118
11	0,7784	1,0000	0,3831	0,5089	11	0,8906	1,0000	0,9158	0,5374
12	0,7504	0,8750	0,3752	0,7518	12	0,9154	1,0000	0,9202	0,6568
13	0,5727	0,6216	0,3752	0,6237	13	0,8823	1,0000	0,7508	0,6606
14	0,5899	0,6251	0,3752	0,6988	14	0,9584	1,0000	0,9202	0,8719
15	0,7875	1,0000	0,3752	0,5622	15	0,9151	1,0000	0,9158	0,6597
16	0,7867	1,0000	0,3752	0,5584	16	0,9082	1,0000	0,7508	0,7902
17	0,7683	0,8746	0,3752	0,8426	17	0,9348	1,0000	0,9202	0,7540
21	0,7185	0,8284	0,3831	0,7241	21	0,9274	0,9987	0,9202	0,7205
22	0,5683	0,6210	0,2796	0,6992	22	0,8301	0,8751	0,7416	0,7836
23	0,7734	0,9641	0,3831	0,5917	23	0,9009	0,9999	0,9202	0,5845
24	0,8229	1,0000	0,5403	0,5744	24	0,8841	0,9964	0,9158	0,5158
25	0,7717	0,9751	0,3831	0,5501	25	0,8437	1,0000	0,7416	0,4768
26	0,8313	1,0000	0,3752	0,7816	26	0,9486	1,0000	0,9202	0,8229
27	0,7496	1,0000	0,3752	0,3729	27	0,8938	1,0000	0,7416	0,7272
31	0,8088	1,0000	0,5403	0,5037	31	0,8894	1,0000	0,9202	0,5269
32	0,7892	1,0000	0,3831	0,5627	32	0,9012	0,9997	0,9158	0,5911
36	0,7812	0,9995	0,3831	0,5244	36	0,9029	0,9991	0,9158	0,6015
37	0,8181	1,0000	0,3831	0,7076	37	0,9256	1,0000	0,9158	0,7124
38	0,7680	1,0000	0,2796	0,5606	38	0,9037	1,0000	0,7416	0,7769
39	0,6933	0,8750	0,3831	0,4583	39	0,8666	1,0000	0,7416	0,5915
40	0,7618	0,9770	0,2796	0,5987	40	0,8659	1,0000	0,7508	0,5785
41	0,7917	1,0000	0,3831	0,5754	41	0,9040	1,0000	0,9202	0,6000
42	0,8060	1,0000	0,5403	0,4895	42	0,9043	1,0000	0,9202	0,6014
46	0,7868	0,9735	0,3831	0,6305	46	0,8860	1,0000	0,9158	0,5141
47	0,8377	1,0000	0,3831	0,8053	47	0,9223	1,0000	0,9158	0,6955
48	0,7277	0,8739	0,3752	0,6416	48	0,8967	1,0000	0,7508	0,7328
49	0,7663	0,9574	0,3752	0,5842	49	0,8593	1,0000	0,7416	0,5550
50	0,5892	0,5675	0,3752	0,8681	50	0,8602	0,9856	0,7508	0,5933
51	0,7610	1,0000	0,2796	0,5254	51	0,8688	1,0000	0,9202	0,4238
52	0,7734	0,9996	0,3831	0,4849	52	0,8610	1,0000	0,7416	0,5636
53	0,7926	0,9999	0,3831	0,5800	53	0,9064	0,9997	0,9202	0,6127
54	0,7002	0,8750	0,2796	0,5966	54	0,7822	0,8311	0,9202	0,4976
55	0,8005	0,9719	0,2796	0,8071	55	0,8390	0,8735	0,7416	0,8328
56	0,8428	0,9854	0,5403	0,7173	56	0,9126	1,0000	0,9158	0,6471
60	0,5682	0,6501	0,3752	0,5158	60	0,7223	0,7497	0,7416	0,6209
61	0,7925	0,9998	0,2796	0,6834	61	0,8981	0,9874	0,7416	0,7868
62	0,1944	0,0297	0,3752	0,5077	62	0,8786	1,0000	0,9202	0,4726
66	0,8117	1,0000	0,2796	0,7791	66	0,8655	0,9996	0,7508	0,5780

Table D.3: Single test 2, left column is at 2.5 meters, right column is at 1.5 meters.

Id	CIPE	EGE	DST	HPE	Id	CIPE	EGE	DST	HPE
2	0,1853	0,0000	0,3752	0,5515	2	0,3869	0,1250	0,9158	0,6437
4	0,2122	0,0673	0,2796	0,5795	4	0,2552	0,0000	0,7416	0,5344
5	0,1997	0,0000	0,3831	0,6153	5	0,2687	0,0000	0,7416	0,6019
6	0,4890	0,5001	0,3752	0,5697	6	0,8423	0,9020	0,9158	0,5896
7	0,4039	0,2559	0,3752	0,8765	7	0,4174	0,2070	0,9158	0,5502
8	0,7403	0,8722	0,5403	0,5444	8	0,3053	0,0122	0,9202	0,5697
9	0,4524	0,3471	0,3752	0,8456	9	0,4369	0,1283	0,9158	0,8840
10	0,2110	0,0121	0,3831	0,6356	10	0,3104	0,0000	0,9112	0,6408
11	0,4049	0,3677	0,3831	0,5382	11	0,3094	0,0000	0,9158	0,6311
12	0,2988	0,0994	0,3752	0,8207	12	0,3416	0,0000	0,9202	0,7877
13	0,2162	0,0000	0,3752	0,7056	13	0,3495	0,0000	0,9158	0,8315
14	0,2787	0,0988	0,3752	0,7220	14	0,3006	0,0000	0,7508	0,7521
15	0,2037	0,0283	0,3752	0,5587	15	0,3152	0,0000	0,9158	0,6604
16	0,2022	0,0000	0,3752	0,6356	16	0,2975	0,0000	0,7508	0,7365
17	0,2453	0,0000	0,3752	0,8514	17	0,3286	0,0000	0,9202	0,7227
21	0,6093	0,6504	0,3831	0,7124	21	0,6095	0,4412	0,9202	0,8038
22	0,1720	0,0028	0,3752	0,4765	22	0,6225	0,4920	0,9202	0,7164
23	0,7152	0,8856	0,3831	0,5361	23	0,3115	0,0000	0,9202	0,6372
24	0,7790	0,9768	0,3752	0,5893	24	0,6992	0,7026	0,9158	0,4723
25	0,1866	0,0000	0,3831	0,5497	25	0,2226	0,0000	0,7416	0,3713
26	0,7380	0,8447	0,2796	0,8764	26	0,3517	0,0009	0,9202	0,8353
27	0,2399	0,1249	0,3752	0,4496	27	0,2802	0,0000	0,9202	0,4810
31	0,4082	0,3285	0,5403	0,5151	31	0,3157	0,0110	0,9857	0,5599
32	0,4585	0,3743	0,5403	0,6295	32	0,2950	0,0000	0,9158	0,5594
36	0,2140	0,0000	0,5302	0,5396	36	0,3142	0,0000	0,9158	0,6551
37	0,1566	0,0000	0,3752	0,4079	37	0,3016	0,0000	0,9202	0,5879
38	0,6956	0,8653	0,2796	0,6027	38	0,2458	0,0304	0,5403	0,5972
39	0,1719	0,0014	0,3831	0,4722	39	0,2670	0,0000	0,7416	0,5933
40	0,3261	0,2641	0,2796	0,5584	40	0,2663	0,0000	0,7508	0,5808
41	0,4101	0,3748	0,2796	0,6463	41	0,3018	0,0000	0,9202	0,5888
42	0,2182	0,0000	0,5403	0,5507	42	0,3076	0,0000	0,9158	0,6223
46	0,7828	1,0000	0,2796	0,6345	46	0,6846	0,6638	0,9158	0,5159
47	0,4301	0,3553	0,3831	0,7017	47	0,3353	0,0000	0,9158	0,7607
48	0,2144	0,0000	0,3752	0,6966	48	0,2813	0,0000	0,7508	0,6559
49	0,1670	0,0000	0,3752	0,4599	49	0,2393	0,0000	0,7416	0,4549
50	0,1815	0,0000	0,2796	0,6278	50	0,2712	0,0000	0,7508	0,6053
51	0,2320	0,1239	0,3831	0,4051	51	0,2752	0,0000	0,9202	0,4559
52	0,2788	0,1522	0,3831	0,5546	52	0,2529	0,0000	0,7416	0,5230
53	0,4672	0,4416	0,3831	0,6280	53	0,3294	0,0000	0,9202	0,7268
54	0,2944	0,1958	0,2796	0,6050	54	0,2688	0,0000	0,9202	0,4238
55	0,5856	0,6160	0,2796	0,8007	55	0,3200	0,0000	0,7416	0,8582
56	0,3386	0,1203	0,5403	0,7918	56	0,3181	0,0000	0,9158	0,6748
60	0,1996	0,0065	0,3752	0,6035	60	0,2994	0,0000	0,9112	0,5858
61	0,3109	0,2030	0,2796	0,6659	61	0,2995	0,0000	0,7416	0,7561
62	0,1889	0,0008	0,3752	0,5673	62	0,5496	0,4335	0,9202	0,5272
66	0,7670	1,0000	0,2796	0,5552	66	0,2799	0,0000	0,7508	0,6489

Table D.4: Single test 3, left column is at 2.5 meters, right column is at 1.5 meters.

Id	CIPE	EGE	DST	HPE	Id	CIPE	EGE	DST	HPE
2	0,7479	0,8380	0,3752	0,8501	2	0,3492	0,0492	0,9158	0,6827
4	0,6134	0,7499	0,2796	0,5374	4	0,2548	0,0025	0,7416	0,5250
5	0,5547	0,5913	0,2796	0,7202	5	0,2719	0,0017	0,7416	0,6130
6	0,5348	0,5606	0,3752	0,6169	6	0,9002	0,9931	0,9158	0,6056
7	0,8412	1,0000	0,5302	0,6757	7	0,8543	0,9042	0,9158	0,6434
8	0,7966	1,0000	0,3831	0,6001	8	0,4842	0,3305	0,9202	0,5090
9	0,8139	1,0000	0,3752	0,6943	9	0,4237	0,2035	0,9857	0,5220
10	0,7965	0,9882	0,3831	0,6346	10	0,3543	0,0993	0,9112	0,5625
11	0,4871	0,4890	0,3831	0,5854	11	0,7894	0,8016	0,9158	0,6266
12	0,8447	0,9981	0,3752	0,8540	12	0,5384	0,3217	0,9202	0,8067
13	0,3058	0,1584	0,3752	0,6784	13	0,2800	0,0000	0,7508	0,6494
14	0,2519	0,0635	0,3752	0,6941	14	0,2913	0,0103	0,7508	0,6749
15	0,6918	0,8029	0,3752	0,6755	15	0,7680	0,7678	0,9158	0,6209
16	0,8267	0,9977	0,3752	0,7653	16	0,4501	0,2493	0,7508	0,7517
17	0,8214	1,0000	0,3752	0,7320	17	0,7568	0,7207	0,9202	0,7020
21	0,7807	0,9502	0,3831	0,6697	21	0,7973	0,8645	0,7416	0,6513
22	0,1934	0,0161	0,2796	0,6391	22	0,9472	0,9994	0,9202	0,8176
23	0,7659	0,9445	0,3831	0,6128	23	0,5401	0,3775	0,9202	0,6478
24	0,7997	1,0000	0,3752	0,6235	24	0,8892	1,0000	0,9158	0,5304
25	0,3233	0,2016	0,3831	0,6286	25	0,2635	0,0000	0,7416	0,5761
26	0,8329	1,0000	0,3752	0,7894	26	0,8155	0,7833	0,9202	0,8073
27	0,2742	0,1999	0,3752	0,3962	27	0,3081	0,0339	0,9202	0,5186
31	0,7426	0,8751	0,5403	0,5475	31	0,7080	0,6426	0,9857	0,6268
32	0,7019	0,8549	0,3831	0,5614	32	0,2974	0,0007	0,9158	0,5692
36	0,2732	0,1393	0,3831	0,5649	36	0,3696	0,0959	0,9158	0,6445
37	0,6863	0,8750	0,3752	0,4315	37	0,4867	0,2949	0,9202	0,6290
38	0,5474	0,6222	0,2796	0,5905	38	0,2788	0,0890	0,5403	0,5865
39	0,2575	0,1140	0,3752	0,5705	39	0,2946	0,0101	0,9202	0,5223
40	0,6866	0,8748	0,2796	0,5290	40	0,2786	0,0237	0,7508	0,5710
41	0,7800	1,0000	0,2796	0,6206	41	0,7998	0,8729	0,7416	0,6387
42	0,5248	0,5636	0,3831	0,5500	42	0,4243	0,1988	0,9158	0,6092
46	0,7610	1,0000	0,2796	0,5255	46	0,7986	0,8735	0,9158	0,4570
47	0,8701	1,0000	0,5403	0,8103	47	0,4434	0,1987	0,9158	0,7053
48	0,4452	0,3818	0,3752	0,7052	48	0,3125	0,0605	0,7508	0,6304
49	0,6432	0,7502	0,3752	0,5900	49	0,5086	0,4099	0,7416	0,5716
50	0,2096	0,0658	0,2796	0,5712	50	0,2721	0,0000	0,7508	0,6098
51	0,5647	0,6676	0,3831	0,4374	51	0,2555	0,0000	0,7416	0,5356
52	0,7910	0,9999	0,3831	0,5719	52	0,3450	0,1083	0,9158	0,4845
53	0,6403	0,7262	0,3831	0,6398	53	0,8070	0,7879	0,9202	0,7509
54	0,6257	0,7447	0,3831	0,5112	54	0,3088	0,0000	0,9158	0,6283
55	0,6346	0,7159	0,3913	0,6339	55	0,4414	0,2092	0,7416	0,8380
56	0,7329	0,8747	0,3752	0,6651	56	0,8863	0,9730	0,9158	0,5967
60	0,1940	0,0210	0,3752	0,5317	60	0,3688	0,1627	0,7416	0,6142
61	0,7735	1,0000	0,2796	0,5881	61	0,8472	0,8667	0,9158	0,7202
62	0,3916	0,3447	0,3752	0,5489	62	0,7852	0,8351	0,9202	0,5005
66	0,7681	1,0000	0,2796	0,5608	66	0,7921	0,8580	0,7508	0,6358

Table D.5: Single test 4, left column is at 2.5 meters, right column is at 1.5 meters.

Id	CIPE	EGE	DST	HPE	Id	CIPE	EGE	DST	HPE
2	0,8373	1,0000	0,3752	0,8114	2	0,9177	0,9995	0,9158	0,6741
4	0,7794	1,0000	0,2796	0,6176	4	0,8925	1,0000	0,9202	0,5421
5	0,7944	0,9916	0,3831	0,6140	5	0,8730	1,0000	0,7416	0,6234
6	0,7854	1,0000	0,3752	0,5518	6	0,9381	0,9996	0,9202	0,7714
7	0,8616	0,9995	0,5302	0,7791	7	0,9119	1,0000	0,9158	0,6438
8	0,7889	1,0000	0,3831	0,5615	8	0,9001	0,9900	0,9202	0,6105
9	0,8350	1,0000	0,3752	0,7999	9	0,9358	0,9961	0,9158	0,7751
10	0,8020	1,0000	0,3831	0,6270	10	0,8961	1,0000	0,9112	0,5692
11	0,7989	1,0000	0,3831	0,6117	11	0,9059	0,9999	0,9158	0,6143
12	0,8306	1,0000	0,3752	0,7776	12	0,9006	1,0000	0,7416	0,7614
13	0,7959	1,0000	0,3752	0,6045	13	0,8881	1,0000	0,7508	0,6899
14	0,8309	0,9995	0,3752	0,7811	14	0,8863	1,0000	0,7508	0,6806
15	0,7911	1,0000	0,3752	0,5802	15	0,9040	1,0000	0,9158	0,6043
16	0,8220	1,0000	0,3752	0,7349	16	0,8906	1,0000	0,7508	0,7021
17	0,8067	1,0000	0,3752	0,6581	17	0,8804	1,0000	0,7416	0,6604
21	0,8165	1,0000	0,3831	0,6992	21	0,9348	0,9996	0,9202	0,7551
22	0,2509	0,0549	0,3752	0,7147	22	0,9568	1,0000	0,9202	0,8636
23	0,7694	0,9434	0,3831	0,6338	23	0,9117	0,9998	0,9202	0,6390
24	0,8116	1,0000	0,5403	0,5178	24	0,8689	1,0000	0,9158	0,4288
25	0,7615	0,9886	0,2796	0,5621	25	0,8433	0,9650	0,7416	0,5800
26	0,8223	1,0000	0,3752	0,7363	26	0,9249	1,0000	0,7416	0,8829
27	0,7775	1,0000	0,3752	0,5125	27	0,8831	1,0000	0,9202	0,4954
31	0,7749	1,0000	0,3831	0,4916	31	0,9143	0,9999	0,9857	0,5859
32	0,8192	0,9753	0,5403	0,6299	32	0,8460	0,8995	0,9158	0,6158
36	0,7869	1,0000	0,3831	0,5513	36	0,7757	0,7998	0,9158	0,5635
37	0,7289	0,8714	0,3831	0,6469	37	0,9016	1,0000	0,9202	0,5880
38	0,7742	1,0000	0,2796	0,5914	38	0,2602	0,0002	0,5403	0,7602
39	0,7711	1,0000	0,3831	0,4724	39	0,8815	1,0000	0,9202	0,4872
40	0,7743	1,0000	0,2796	0,5917	40	0,7870	0,8750	0,7508	0,5591
41	0,7785	1,0000	0,2796	0,6129	41	0,8724	1,0000	0,7416	0,6201
42	0,7880	1,0000	0,3752	0,5649	42	0,8933	1,0000	0,9202	0,5464
46	0,7972	1,0000	0,3831	0,6029	46	0,8692	0,9964	0,9158	0,4408
47	0,8313	1,0000	0,3831	0,7736	47	0,9357	1,0000	0,9158	0,7629
48	0,8120	0,9999	0,3752	0,6848	48	0,8840	0,9999	0,7508	0,6696
49	0,7038	0,8749	0,3752	0,5190	49	0,8943	1,0000	0,9158	0,5559
50	0,7471	0,9300	0,2796	0,6659	50	0,8847	0,9991	0,7508	0,6752
51	0,7589	1,0000	0,3831	0,4113	51	0,8683	0,9979	0,9202	0,4277
52	0,7919	1,0000	0,3831	0,5765	52	0,8764	1,0000	0,9158	0,4664
53	0,8151	1,0000	0,3831	0,6924	53	0,9458	1,0000	0,9202	0,8091
54	0,7984	1,0000	0,2796	0,7123	54	0,8857	1,0000	0,9202	0,5085
55	0,7195	0,8495	0,3913	0,6575	55	0,8289	0,8275	0,9202	0,7417
56	0,7938	1,0000	0,3752	0,5939	56	0,9081	1,0000	0,9158	0,6249
60	0,7689	0,9922	0,3752	0,4926	60	0,8809	0,9997	0,9158	0,4895
61	0,8012	1,0000	0,2796	0,7264	61	0,8996	0,9993	0,7416	0,7587
62	0,3012	0,1705	0,3752	0,6195	62	0,8850	1,0000	0,9202	0,5050
66	0,7698	1,0000	0,2796	0,5695	66	0,8769	1,0000	0,7508	0,6339

Table D.6: Single test 5, left column is at 2.5 meters, right column is at 1.5 meters.

Id	CIPE	EGE	DST	HPE	Id	CIPe	EGE	DST	HPE
2	0,7772	1,0000	0,3752	0,5108	2	0,9045	0,9776	0,9158	0,6739
4	0,7070	0,8750	0,2796	0,6302	4	0,7109	0,7477	0,7416	0,5698
5	0,4993	0,4975	0,3831	0,6211	5	0,8426	0,9263	0,9158	0,5180
6	0,7942	1,0000	0,3752	0,5959	6	0,8381	0,8558	0,9202	0,7029
7	0,8245	1,0000	0,5302	0,5922	7	0,8584	0,8991	0,9857	0,6091
8	0,7979	1,0000	0,3831	0,6065	8	0,8940	1,0000	0,9202	0,5497
9	0,8376	0,9894	0,3752	0,8444	9	0,3301	0,0000	0,9857	0,6645
10	0,7737	1,0000	0,3831	0,4853	10	0,8970	0,9790	0,9866	0,5611
11	0,7991	1,0000	0,3831	0,6124	11	0,8019	0,8535	0,9158	0,5330
12	0,8575	0,9999	0,3752	0,9127	12	0,7276	0,7179	0,7416	0,7424
13	0,6344	0,7308	0,3752	0,6048	13	0,7910	0,8731	0,7508	0,5848
14	0,5423	0,5700	0,3752	0,6262	14	0,9004	1,0000	0,9158	0,5861
15	0,7891	1,0000	0,3752	0,5703	15	0,8962	0,9971	0,9158	0,5742
16	0,8064	1,0000	0,3752	0,6571	16	0,9451	1,0000	0,9202	0,8055
17	0,7815	0,9970	0,2796	0,6372	17	0,8314	0,8750	0,9202	0,6116
21	0,8146	0,9959	0,3831	0,7019	21	0,6691	0,6242	0,9202	0,5527
22	0,7389	0,9021	0,2796	0,7088	22	0,9506	0,9989	0,9202	0,8360
23	0,6939	0,8220	0,3831	0,6203	23	0,3686	0,1014	0,9202	0,6185
24	0,7856	1,0000	0,3752	0,5527	24	0,8833	1,0000	0,9158	0,5008
25	0,7713	1,0000	0,2796	0,5771	25	0,8564	0,9978	0,7416	0,5468
26	0,7887	1,0000	0,3752	0,5685	26	0,9233	1,0000	0,7416	0,8747
27	0,7693	1,0000	0,2796	0,5667	27	0,8925	1,0000	0,7416	0,7208
31	0,6458	0,7493	0,5403	0,4409	31	0,5714	0,4424	0,9158	0,6140
32	0,8197	1,0000	0,5403	0,5580	32	0,7974	0,8189	0,9158	0,6145
36	0,7764	0,9992	0,3831	0,5013	36	0,9040	0,9997	0,9158	0,6053
37	0,7821	1,0000	0,3752	0,5356	37	0,9081	1,0000	0,9202	0,6202
38	0,4642	0,4990	0,2796	0,5446	38	0,2349	0,0084	0,5403	0,6089
39	0,5608	0,6204	0,3752	0,5678	39	0,4625	0,2908	0,9202	0,5199
40	0,2952	0,1986	0,2796	0,6005	40	0,8379	1,0000	0,5403	0,6491
41	0,8014	0,9987	0,3752	0,6357	41	0,8693	1,0000	0,7416	0,6049
42	0,7452	0,8750	0,5403	0,5607	42	0,8836	0,9736	0,9158	0,5816
46	0,7805	0,9996	0,3831	0,5205	46	0,6267	0,5850	0,9158	0,4630
47	0,8717	1,0000	0,5403	0,8181	47	0,9219	1,0000	0,9158	0,6938
48	0,7931	0,9979	0,3752	0,5965	48	0,9359	0,9956	0,9202	0,7726
49	0,7579	0,9786	0,3752	0,4785	49	0,8598	1,0000	0,7416	0,5576
50	0,6289	0,7386	0,2796	0,6494	50	0,8668	0,9968	0,7508	0,5928
51	0,6678	0,8516	0,3831	0,4010	51	0,5954	0,5453	0,9202	0,4207
52	0,7831	0,9979	0,3831	0,5386	52	0,8782	0,9998	0,9158	0,4756
53	0,8134	1,0000	0,3831	0,6838	53	0,9587	1,0000	0,9202	0,8732
54	0,7846	0,9995	0,3831	0,5413	54	0,8015	0,8750	0,9202	0,4622
55	0,7308	0,8633	0,3913	0,6727	55	0,8253	0,8718	0,7416	0,7697
56	0,8335	0,9978	0,5403	0,6338	56	0,8759	1,0000	0,7416	0,6378
60	0,4714	0,4854	0,3752	0,5256	60	0,5427	0,4631	0,7416	0,5827
61	0,7991	1,0000	0,3752	0,6202	61	0,8909	1,0000	0,7416	0,7127
62	0,3417	0,2652	0,3752	0,5375	62	0,8092	0,8941	0,9202	0,4434
66	0,7078	0,8721	0,2796	0,6429	66	0,8792	0,9993	0,7508	0,6472

Table D.7: Single test 6, left column is at 2.5 meters, right column is at 1.5 meters.

Id	CIPE	EGE	DST	HPE	Id	CIPE	EGE	DST	HPE
2	0,8149	1,0000	0,3752	0,6994	2	0,8145	0,8495	0,7508	0,7733
4	0,7798	1,0000	0,2796	0,6196	4	0,7731	0,8727	0,7416	0,5059
5	0,7909	1,0000	0,3831	0,5713	5	0,7893	0,8750	0,7416	0,5799
6	0,7775	1,0000	0,3752	0,5125	6	0,9228	1,0000	0,9202	0,6937
7	0,8625	1,0000	0,5302	0,7825	7	0,9048	1,0000	0,9158	0,6083
8	0,7979	1,0000	0,3831	0,6062	8	0,8984	1,0000	0,9202	0,5721
9	0,7694	0,8750	0,3831	0,8387	9	0,4330	0,2491	0,9857	0,4320
10	0,8013	1,0000	0,3831	0,6234	10	0,8397	0,9521	0,9158	0,4266
11	0,7896	1,0000	0,3831	0,5647	11	0,6213	0,5479	0,9158	0,5469
12	0,8323	1,0000	0,3752	0,7861	12	0,8134	0,8640	0,7416	0,7334
13	0,6795	0,8159	0,3752	0,5746	13	0,8674	0,9975	0,7508	0,5936
14	0,8257	1,0000	0,3752	0,7535	14	0,9011	1,0000	0,7508	0,7547
15	0,7841	0,9974	0,3752	0,5534	15	0,8996	0,9995	0,9158	0,5839
16	0,8217	1,0000	0,3752	0,7332	16	0,7987	0,8527	0,7508	0,6848
17	0,7907	1,0000	0,3752	0,5784	17	0,9104	1,0000	0,9202	0,6321
21	0,8254	1,0000	0,3831	0,7439	21	0,9386	0,9885	0,9202	0,8072
22	0,5386	0,5217	0,2796	0,8483	22	0,9283	1,0000	0,9202	0,7214
23	0,7821	0,9991	0,2796	0,6337	23	0,9004	0,9998	0,9202	0,5824
24	0,7885	1,0000	0,3752	0,5674	24	0,8839	1,0000	0,9158	0,5036
25	0,6839	0,8750	0,2796	0,5147	25	0,6986	0,7500	0,7416	0,5015
26	0,7820	1,0000	0,3752	0,5347	26	0,8459	0,8752	0,9202	0,6836
27	0,7867	1,0000	0,3752	0,5582	27	0,5876	0,4958	0,7416	0,7087
31	0,6955	0,8590	0,3913	0,5091	31	0,9095	0,9999	0,9158	0,6322
32	0,7881	1,0000	0,3752	0,5654	32	0,8864	0,9988	0,9158	0,5199
36	0,7770	1,0000	0,3831	0,5021	36	0,6555	0,5353	0,9857	0,6858
37	0,7561	1,0000	0,3752	0,4055	37	0,8975	1,0000	0,9202	0,5675
38	0,6190	0,7502	0,2796	0,5647	38	0,4228	0,3227	0,5403	0,6054
39	0,7763	1,0000	0,3752	0,5061	39	0,8557	0,9996	0,7508	0,5289
40	0,7525	0,9520	0,2796	0,6270	40	0,7783	0,8427	0,7508	0,6124
41	0,8032	1,0000	0,3752	0,6409	41	0,8662	1,0000	0,7416	0,5895
42	0,7860	1,0000	0,3831	0,5471	42	0,8928	1,0000	0,9158	0,5480
46	0,7737	1,0000	0,3831	0,4853	46	0,8717	1,0000	0,9158	0,4429
47	0,8684	1,0000	0,5403	0,8018	47	0,9259	1,0000	0,9202	0,7091
48	0,7777	0,9965	0,3831	0,5158	48	0,9486	1,0000	0,9202	0,8229
49	0,2983	0,2375	0,3752	0,4037	49	0,2534	0,0000	0,9158	0,3512
50	0,7542	0,9400	0,2796	0,6717	50	0,7327	0,7476	0,7508	0,6699
51	0,6725	0,8751	0,3831	0,3541	51	0,8347	1,0000	0,7416	0,4320
52	0,7846	1,0000	0,3831	0,5397	52	0,8775	1,0000	0,9158	0,4716
53	0,8316	1,0000	0,3831	0,7751	53	0,9496	0,9960	0,9202	0,8395
54	0,7900	0,9922	0,3831	0,5902	54	0,8048	0,8750	0,9202	0,4791
55	0,4033	0,3737	0,3913	0,5039	55	0,2913	0,0002	0,7416	0,7141
56	0,7999	1,0000	0,3752	0,6241	56	0,9179	1,0000	0,9158	0,6738
60	0,7707	1,0000	0,3752	0,4782	60	0,8601	1,0000	0,7416	0,5587
61	0,7942	0,9955	0,3752	0,6095	61	0,3381	0,0827	0,7416	0,7010
62	0,3725	0,3108	0,3752	0,5548	62	0,8757	1,0000	0,9202	0,4583
66	0,7735	1,0000	0,2796	0,5879	66	0,8773	0,9999	0,7508	0,6362

Table D.8: Single test 7, left column is at 2.5 meters, right column is at 1.5 meters.

Id	CIPE	EGE	DST	HPE	Id	CIPE	EGE	DST	HPE
2	0,2716	0,0742	0,3752	0,7601	2	0,4351	0,2322	0,7508	0,7280
4	0,3905	0,3689	0,3752	0,4710	4	0,2539	0,0000	0,7416	0,5280
5	0,5348	0,5677	0,3831	0,5878	5	0,2813	0,0000	0,9158	0,4906
6	0,7938	1,0000	0,3752	0,5940	6	0,9320	1,0000	0,9202	0,7401
7	0,8788	0,9999	0,5302	0,8643	7	0,8957	0,9950	0,9158	0,5775
8	0,7927	1,0000	0,3831	0,5805	8	0,8920	0,9999	0,9202	0,5400
9	0,5231	0,4850	0,3752	0,7852	9	0,5922	0,5000	0,9202	0,5407
10	0,2637	0,1226	0,3831	0,5674	10	0,2819	0,0000	0,9158	0,4939
11	0,7104	0,8691	0,3831	0,5618	11	0,2975	0,0000	0,9158	0,5717
12	0,8501	0,9952	0,3752	0,8900	12	0,2879	0,0000	0,7416	0,6978
13	0,7218	0,8728	0,3752	0,6157	13	0,2555	0,0000	0,7508	0,5266
14	0,8317	1,0000	0,3752	0,7835	14	0,2903	0,0000	0,7508	0,7007
15	0,5671	0,6163	0,3752	0,6112	15	0,2867	0,0000	0,9158	0,5177
16	0,7964	0,9878	0,3752	0,6434	16	0,3104	0,0000	0,7508	0,8010
17	0,6547	0,7619	0,3752	0,6129	17	0,2728	0,0000	0,7416	0,6223
21	0,8242	0,9990	0,3831	0,7410	21	0,4002	0,1105	0,9202	0,7490
22	0,5989	0,6123	0,2796	0,8782	22	0,7361	0,6250	0,9202	0,8852
23	0,6762	0,8225	0,2796	0,6339	23	0,3710	0,1094	0,9202	0,6067
24	0,8118	1,0000	0,3752	0,6838	24	0,8946	0,9961	0,9158	0,5689
25	0,7456	1,0000	0,2796	0,4483	25	0,2296	0,0000	0,7416	0,4065
26	0,8357	1,0000	0,3752	0,8032	26	0,5550	0,4187	0,7416	0,7770
27	0,7157	0,8750	0,3752	0,5783	27	0,2866	0,0000	0,7416	0,6912
31	0,2136	0,0014	0,5403	0,5232	31	0,4697	0,2952	0,9202	0,5429
32	0,6757	0,7501	0,5403	0,5878	32	0,2874	0,0000	0,9158	0,5212
36	0,5551	0,6065	0,5302	0,4257	36	0,3150	0,0000	0,9158	0,6591
37	0,6786	0,8692	0,3752	0,4102	37	0,2781	0,0000	0,9202	0,4701
38	0,7537	1,0000	0,2796	0,4890	38	0,2196	0,0000	0,5403	0,5579
39	0,7713	1,0000	0,2796	0,5768	39	0,2558	0,0000	0,7508	0,5281
40	0,5205	0,5852	0,2796	0,5670	40	0,2756	0,0000	0,7508	0,6270
41	0,6327	0,7500	0,2796	0,6338	41	0,2951	0,0000	0,9202	0,5554
42	0,6583	0,7497	0,5403	0,5021	42	0,2781	0,0005	0,9158	0,4734
46	0,7603	0,9917	0,3831	0,4431	46	0,8810	1,0000	0,9158	0,4891
47	0,8502	0,9679	0,5403	0,8068	47	0,4832	0,2494	0,9202	0,7474
48	0,3144	0,2348	0,3752	0,4926	48	0,3583	0,0000	0,9202	0,8712
49	0,1486	0,0000	0,3752	0,3677	49	0,2228	0,0000	0,7416	0,3724
50	0,5851	0,6933	0,2796	0,5659	50	0,3498	0,0978	0,7508	0,7049
51	0,2336	0,1260	0,3831	0,4070	51	0,2605	0,0000	0,9202	0,3822
52	0,7750	1,0000	0,3831	0,4919	52	0,6000	0,5442	0,9158	0,4517
53	0,8453	1,0000	0,5403	0,6860	53	0,5651	0,3760	0,9202	0,7773
54	0,5608	0,6251	0,3831	0,5456	54	0,2727	0,0000	0,9202	0,4431
55	0,4612	0,3772	0,2796	0,8948	55	0,3100	0,0000	0,7416	0,8085
56	0,8026	1,0000	0,3752	0,6380	56	0,5928	0,4589	0,9158	0,6712
60	0,8006	0,9957	0,3752	0,6407	60	0,4369	0,3041	0,7416	0,5306
61	0,7265	0,8749	0,2796	0,7283	61	0,2673	0,0000	0,7416	0,5947
62	0,1843	0,0036	0,3752	0,5355	62	0,4171	0,2486	0,9202	0,4195
66	0,8158	1,0000	0,3752	0,7040	66	0,5689	0,4733	0,7508	0,6738

Table D.9: Group test 1. Results are in groups of three, corresponding to the test subjects' id:s and assigned colors (red, green, blue).

Id	CIPE	EGE	DST	HPE
1	0,8374	0,9995	0,3831	0,8053
2	0,2062	0,0000	0,3752	0,6556
3	0,5891	0,6308	0,3752	0,6778
12	0,7859	1,0000	0,3831	0,5463
13	0,1696	0,0015	0,2796	0,5641
14	0,4048	0,3752	0,3752	0,5233
4	0,3397	0,2459	0,3831	0,5778
11	0,1784	0,0000	0,3831	0,5090
15	0,3406	0,2471	0,3752	0,5867
18	0,7081	0,8750	0,3831	0,5326
19	0,7756	0,9465	0,3752	0,6635
20	0,6876	0,8750	0,3752	0,4378
28	0,5575	0,5598	0,7508	0,3571
30	0,8115	0,9995	0,5403	0,5188
29	0,7667	0,9999	0,3752	0,4586
33	0,1750	0,0027	0,3752	0,4918
35	0,7604	0,9968	0,2796	0,5321
34	0,8084	0,9982	0,3913	0,6563
43	0,2276	0,1154	0,2796	0,5121
45	0,1610	0,0012	0,3752	0,4265
44	0,3214	0,1096	0,5403	0,7377
57	0,1664	0,0000	0,3752	0,4568
58	0,1685	0,0000	0,3752	0,4676
59	0,6028	0,7156	0,5403	0,3269
63	0,3821	0,3544	0,3752	0,4719
64	0,1700	0,0000	0,2796	0,5702
65	0,7776	1,0000	0,3752	0,5127

Table D.10: Group test 2. Results are in groups of three, corresponding to the test subjects' id:s and assigned colors (red, green, blue).

Id	CIPE	EGE	DST	HPE
1	0,8220	1,0000	0,3831	0,7269
2	0,7972	1,0000	0,3752	0,6108
3	0,8263	1,0000	0,5202	0,6112
12	0,8058	1,0000	0,3831	0,6458
13	0,4151	0,3763	0,3752	0,5712
14	0,7716	1,0000	0,3752	0,4828
4	0,6816	0,8534	0,2796	0,5681
11	0,6982	0,8626	0,3831	0,5203
15	0,7516	1,0000	0,3831	0,3747
18	0,7106	0,8750	0,3831	0,5449
19	0,7845	0,9999	0,2796	0,6429
20	0,7826	1,0000	0,3752	0,5379
28	0,8311	0,9986	0,3752	0,7846
30	0,7669	0,8749	0,3913	0,8184
29	0,7523	0,8284	0,3752	0,9013
33	0,8710	1,0000	0,5403	0,8146
35	0,7304	0,8573	0,2796	0,8004
34	0,7997	1,0000	0,3831	0,6154
43	0,6724	0,8750	0,2796	0,4576
45	0,6422	0,7827	0,3752	0,4874
44	0,7860	0,9994	0,3913	0,5406
57	0,7571	0,9959	0,3752	0,4227
58	0,7762	1,0000	0,3752	0,5058
59	0,7581	0,9998	0,3752	0,4157
63	0,7151	0,8725	0,5403	0,4178
64	0,7482	0,9997	0,2796	0,4623
65	0,7807	1,0000	0,3752	0,5284

Table D.11: Group test 3. Results are in groups of three, corresponding to the test subjects' id:s and assigned colors (red, green, blue).

Id	CIPE	EGE	DST	HPE
1	0,8399	1,0000	0,3831	0,8165
2	0,2314	0,0000	0,7508	0,4061
3	0,2927	0,0000	0,9112	0,5521
12	0,7998	1,0000	0,3831	0,6161
13	0,2193	0,0000	0,7508	0,3458
14	0,4428	0,2501	0,9112	0,5526
4	0,7985	1,0000	0,3831	0,6095
11	0,2754	0,0000	0,9202	0,4568
15	0,2631	0,0000	0,9112	0,4041
18	0,2717	0,0000	0,7508	0,6080
19	0,7966	1,0000	0,3831	0,5998
20	0,5328	0,5066	0,7416	0,4027
28	0,8338	1,0000	0,3752	0,7937
30	0,3554	0,1279	0,9202	0,4733
29	0,6768	0,5229	0,9112	0,9040
33	0,8122	0,9999	0,5403	0,5208
35	0,1905	0,0000	0,7416	0,2110
34	0,9024	0,9998	0,7416	0,7709
43	0,3236	0,0000	0,9202	0,6977
45	0,7612	0,9985	0,3752	0,4357
44	0,2443	0,0000	0,7416	0,4800
57	0,7914	1,0000	0,3752	0,5817
58	0,2134	0,0000	0,7416	0,3256
59	0,2588	0,0000	0,9158	0,3781
63	0,7682	1,0000	0,3752	0,4657
64	0,1518	0,0000	0,5302	0,2289
65	0,8194	0,8482	0,9202	0,6323

Table D.12: Group test 4. Results are in groups of three, corresponding to the test subjects' id:s and assigned colors (red, green, blue).

Id	CIPE	EGE	DST	HPE
1	0,7974	1,0000	0,3831	0,6040
2	0,9114	1,0000	0,7508	0,8060
3	0,9007	1,0000	0,9158	0,5877
12	0,7908	0,8550	0,9158	0,4731
13	0,7968	1,0000	0,3752	0,6088
14	0,7826	0,8750	0,7508	0,5372
4	0,8561	1,0000	0,9158	0,3646
11	0,7762	0,9977	0,3831	0,5049
15	0,6743	0,6241	0,9202	0,5793
18	0,7945	0,8750	0,7508	0,5965
19	0,8373	1,0000	0,5403	0,6461
20	0,1929	0,0005	0,5403	0,4229
28	0,8299	1,0000	0,3752	0,7743
30	0,7810	0,8668	0,9857	0,3187
29	0,7568	0,6745	0,9158	0,8449
33	0,8437	1,0000	0,5403	0,6784
35	0,2014	0,0000	0,7416	0,2656
34	0,9519	0,9999	0,9202	0,8395
43	0,6505	0,8749	0,3752	0,2524
45	0,7474	0,8664	0,7416	0,3960
44	0,9319	1,0000	0,9158	0,7437
57	0,7978	1,0000	0,3752	0,6136
58	0,8189	1,0000	0,9158	0,1787
59	0,8790	1,0000	0,9202	0,4750
63	0,8062	1,0000	0,5403	0,4905
64	0,8339	0,9999	0,5403	0,6293
65	0,8452	0,9999	0,7416	0,4849