Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Mathilde Haukanes

Developing Realistic Eye Movement in Medical Simulation Manikins: A Study using Eye-Tracking and Machine Learning

Master's thesis in Engineering and ICT
Supervisor: Bjørn Haugen
Co-supervisor: Knut Einar Aasland
June 2023

NTNU
Norwegian University of
Science and Technology

Mathilde Haukanes

Developing Realistic Eye Movement in Medical Simulation Manikins: A Study using Eye-Tracking and Machine Learning

Master's thesis in Engineering and ICT
Supervisor: Bjørn Haugen
Co-supervisor: Knut Einar Aasland
June 2023

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

PREFACE

This thesis marks the end of the five-year study program Engineering and ICT, with specialization in ICT and Machine Technology, at the Norwegian University of Science and Technology (NTNU). The master thesis was written during the spring semester of 2023, from January to June, and is a continuation of the project thesis written in the fall semester of 2022.

This thesis was written in close collaboration with Laerdal Medical AS, as the company wanted to explore the possibilities for implementing realistic eye movement in their manikins.

Mathilde Haukanes

Trondheim, 11.06.2023

ACKNOWLEDGMENT

This master thesis was written in collaboration with Laerdal Medical AS, and I would like to express my deepest gratitude for the interesting problem for my thesis, close collaboration, supervision, and great discussions. In particular, I would like to thank Andreas Kråkenes, Marius Auflem and Magnus Ove. Besides, I would like to thank my supervisor Bjørn Haugen and co-supervisor Knut Einar Aasland, for their insight and supervision during the conduction of this master thesis, as well as making the collaboration with Leardal Medical possible. I also want to extend my kindest regards to family and friends who encouraged me during my years of study.

Mathilde Haukanes

ABSTRACT

This research introduces an innovative approach to understanding visual attention by combining data collection through eye-tracking experiment and object detection. The data collected are used in the training of the gaze point prediction model to accurately predict the next gaze points. These methodologies also provide valuable insights into participants' gaze behavior and attention patterns. The objective is to develop an algorithm that allows Laerdal Medical's patient simulator to replicate realistic human eye movements, improving the realism of simulated medical training scenarios.

In the study, a important aspect involved analyzing the collected data to gain insights into visual attention patterns among the 10 participants. This analysis revealed patterns in dwell time for different objects, with all participants spending the most time fixating on the object that were most salient in the experimental video. Furthermore, while certain patterns were identified among the participants, individual differences were also found, highlighting the variations in visual attention across the group of participants.

Based on the collected data and analysis, a gaze point prediction model was developed. The model's performance was evaluated, demonstrating improved accuracy with a reduced number of participants in the dataset. Handling larger and more diverse datasets presented significant challenge. The need to effectively manage diverse data, reflecting the viewing patterns of multiple participants, highlights the importance of further improvements in the model's scalability and computational efficiency.

The study's limitations include the use of pre-collected non-real-time data from ten participants, which presents challenges for real-time implementation. Future research can focus on enhancing real-time performance and refining the object detection model to to avoid the need for data cleaning.

SAMMENDRAG

Denne oppgaven introduserer en innovativ tilnærming for å forstå visuell oppmerksomhet ved å kombinere øyesporing og objektdeteksjonsdata. Innsamlede data brukes til å trene en modell som predikerer neste punkt å feste blikket. Metodene brukt i studien gir verdifull innsikt i deltakernes fokus og mønstre i hvor de ser. Målet for oppgaven er å utvikle en algoritme som gjør det mulig for Laerdal Medical sin pasientsimulator å etterligne realistiske øyebevegelser, noe som kan øke følelsen av realisme i simulerte medisinske opplæringsscenarier.

Studien innebar analyse av innsamlede data for å avdekke visuell oppmerksomhet blant 10 deltakere. Fra analysen ser man mønstre i fokuset på ulike objekter blant deltakerne, der de i stor grad fokuserte mest på objektet i eksperimentvideoen som er mest kontaktsøkende. Samtidig ble det identifisert individuelle forskjeller, som viser variasjonene i oppmerksomhet blant deltakerne.

Basert på de innsamlede dataene og analysen ble det utviklet en modell for å forutsi det neste blikkpunktet. Modellens ytelse ble evaluert og viste forbedret nøyaktighet med færre deltakere inkludert i datasettet. Bruken av større og mer varierte datasett førte til utfordringer. Behovet for bedre håndtering av variert data, som gjenspeiler blikkmønstre hos flere deltakere, understreker viktigheten av ytterligere forbedringer av modellens skalerbarhet og effektivitet.

Studiens begrensninger inkluderer bruk av innhentede ikke-sanntidsdata fra ti deltakere, noe som skaper utfordringer for sanntidsimplementering i pasientsimulatoren. Videre forskning kan fokusere på forbedring av sanntidsytelse og forbedring av objektdeteksjonsmodellen for å unngå behovet for rengjøring av data.

ABBREVIATIONS

- **CNN** Convolutional Neural Network
- **Google Colab** Google Colaboratory
- **LSTM** Long Short-Term Memory
- **MAE** Mean Absolute Error
- **MSE** Mean Squared Error
- **NTNU** Norwegian University of Science and Technology
- **OSNet** Omi-Scale Network
- **re-ID** person re-identification
- **RMSE** Root Mean Squared Error
- **RNN** Recurrent Neural Network
- **YOLO** You Only Look Once
- **YOLOv8** You Only Look Once Version 8

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER

ONE

INTRODUCTION

Laerdal Medical is a company that is dedicated to their goal of "Helping Save Lives" through the advancement of resuscitation, patient care, and global health initiatives [1]. The products developed by Laerdal Medical are used in education, among other things, and one of their products are manikins that provides realistic preparation for difficult health cases. This product allows for repeatable training on real life health scenarios, allowing for errors to be made without harming anyone [2]. The SimMan is provided with many simulators, which makes it possible to test students in clinical and decision-making skills during realistic patient-care scenarios [2]. Medical training manikins are used to educate amateurs, as well as professionals in various medical situations [3]. Figure 1.0.1 show an example of one of their manikins.

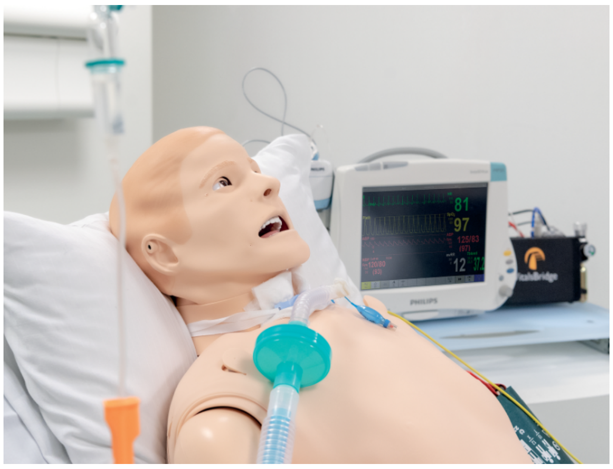

Figure 1.0.1: SimMan [4]

The modern manikins include an array of sensors, actuators, and communication system, which make an educational training setup with varying level of stress [3]. Laerdal Medical is world-leading producer of such manikins [3], and are also continuously working on making the simulators more lifelike. This bring a new level of realism to the learning, and to make this possible high-fidelity devices are required [5].

While these manikins are able to simulate a wide range of interactions, there are some elements of the manikin that are “too dead” to be used in training [3]. An example of this is the eyes of the manikin . This is a potential area for improvement, to implement movement in the eyes. This is because the eyes can be important both in diagnostic procedures and in interaction with medical personnel. Laerdal Medical is exploring the possibility of implementing eye movement in the manikin, in order to improve the realism of the training experience.

1.1 Backgound and Motivation

Eye movement are studied in relation to patient-centered communication variables, and can impact the medical practitioners awareness of the patient, as well as the patients’ psychological and cognitive functioning [6]. Eye contact is for example an important nonverbal behavior that can communicate a lot of information from a patient [6]. In emergency situations, the diagnosis often involves interaction with the patient’s eyes. We have the voluntary (follow a pen) and involuntary (oscillations of the eye) movements that are indicators of several injuries, like neural damage and brain injuries [3]. This is just some examples where eyes are important in diagnostics. In addition to diagnostics, the eyes are a factor that would help to make the manikins more realistic, and therefore make the simulations more realistic.

The realism of robots can be improved by making their eyes move in a natural manner, but this can be challenging. Care must be taken when robots become more alike a human. The more human the robot feels, until a point called the “uncanny valley” [7], the more realistic it is experienced. The uncanny valley is where robots appear humanlike, but in a way that is uncomfortable [8]. In order to avoid this, the reactions of the eyes should be similar to those of a human, so it does not seem unnatural. This is something that have to be considered when implementing movements in the eyes of the manikin.

1.1.1 Project thesis

This master’s thesis was preceded by a project thesis that was completed as part of the Engineering and ICT Master’s program at NTNU, earning 15 ECTS credits. The aim of the project thesis was to gain an understanding of eye-tracking experiment setup, and to identify the most suitable technology and method for data collection using eye-tracking. This research included the following tasks:

- Literature study
- Planning experiment
- Conducting experiment
- Analysing data

This research project was carried out over a period of four months in 2022, and the knowledge obtained has been used as the basis for this master's thesis. Certain aspects of the work conducted and documented in the project thesis will be reused and integrated into this master's thesis, where considered relevant and appropriate. This project provided a valuable learning experience on the protocols essential for conducting an eye-tracking experiment. The most significant knowledge gained was learning the necessary preparations prior to, during, and after the experiment, to ensure optimal data quality. The study also involved the testing of two different eye-tracking methods to determine their efficacy and accuracy. The results of this comparison provide a foundation for selecting technology for data collection in this master thesis. In the final part of the project, the collected data was analyzed, giving a clearer idea of what kind of information an eye-tracking experiment can provide.

1.2 Thesis Objective

The main objective of this master thesis is to explore and build an algorithm that enables Laerdal Medical's patient simulators, which is a full-body patient simulator that mimics human anatomy and physiology, to have realistic eye movements. This work on exploring implementation of eye movement is driven by two main goals: improving the realism of the simulated medical scenarios to make them more effective as teaching tools, and enabling the simulator to mimic important signs related to eye movement and gaze direction, which can be key to medical diagnosis.

The specific goals of this research include:

1. Data Collection: To collect and analyze eye-tracking data in order to understand how humans focuses and shifts their attention visually. This includes identifying the objects they focus on, the duration of gaze fixation, and how

the gaze shifts when multiple objects are present in the field of view. A point of particular interest is determining whether salient objects attract more visual attention. The advantage of using an eye-tracker is that it allows for the study of actual eye movement.

2. Object Detection: To use an existing object detection algorithm to identify the objects present within the field of view. This object detection model will be used on the video of the field of view, recorded by the eye-tracker for each participant.

3. Interest Identification: Combine the data from eye-tracking experiment and the object detection. The goal is to determine which elements in a scene are likely to attract a human's attention. Specifically, we want to understand whether salient objects, such as a person interacting with the viewer through eye contact and talking, attract more visual attention compared to people in the background who are not engaged with direct interaction. This involves studying the patterns of eye movements across different participants to determine what consistently attract focus. These patterns can provide valuable insights into understanding human visual attention and can be a significant factor to include in the gaze point prediction model. Ultimately, this information is important in making the simulator's eye movements realistic and adaptive.

4. Eye Movement Prediction: To design and implement a machine learning model that can predict future gaze points based the data collected in the eye-tracking experiment and the detected objects in the visual field.

The goal of this thesis is to investigate how these techniques can improve the realism of patient simulators. The primary ambition is to use the collected data to train a model capable of recognizing "interesting" elements in a scene, and then directing the simulated gaze towards these elements. This exploration holds potential for improving the effectiveness of medical training, and may lead to improved patient outcomes in real-life medical scenarios.

1.3 Thesis Structure

Chapter 2 gives an introduction to relevant theory to better understand the work conducted in this thesis. This chapter is divided into three parts: theory related to eye-tracking for data collection, theory related to object detection, and theory related to the model predicting gaze points.

Chapter 3 explains how the experiment was designed and conducted, as well as how the data was cleaned afterwords.

Chapter 4 describe the methodology used to implement the YOLOv8 object detection model. This chapter also discusses the use of various tools and describe the necessary code modifications made to adapt it for the specific requirements of the research.

Chapter 5 aims to provide an understanding of how the model that predicts gaze points is designed and executed, covering feature extraction, model architecture and training, as well as the evaluation of the model.

Chapter 6 provide an understanding of the outcomes from the eye-tracking experiment, the object detection analysis, the combination of eye-tracking and object detection, and the evaluation of the model predicting gaze points.

Chapter 7 discusses the results, as well as highlighting some limitations of the study. The chapter also include suggestions for future research for further advancements in the field.

Chapter 8 presents the conclusion of the study.

CHAPTER

TWO

THEORY

2.1 Theory Eye-Tracking

This section introduces key concepts in the field of eye-tracking, aiming to enhance the understanding of the practical applications of eye-tracking technology and its relevance to the research conducted in this study. By exploring these topics, valuable insights are gained in the use of eye-tracking data and its significance in the work undertaken.

2.1.1 Eye-tracking

Eye tracking is an experimental method of recording eye movements and gaze point over time, and during different tasks [9]. The meaning of gaze point is illustrated in Figure 2.1.1. Eye tracking allow researchers to study visual attention by identifying where individuals are looking [9][10]. Historically, eye tracking was expensive and effortful, requiring researchers to manually observe and record individual behavior [9]. However, over the years there has been many improvements in the eye tracking technology, and it has become more affordable and user friendly [9]. Advances in eye tracking technology have made it accessible for use as a research tool for multiple disciplines. It has been used in numerous fields including human factors, cognitive psychology, marketing and human-computer interaction [11]. To conduct eye tracking experiments, specialized equipment called an eye-tracker is required [10].

Eye movements are influenced by cognitive processes such as perception, memory, language and decision making [9]. They are largely reflexive and outside of conscious control [9]. Perturbations in the movement control network of the brain produce unique and measurable signs, making eye movement useful for character-

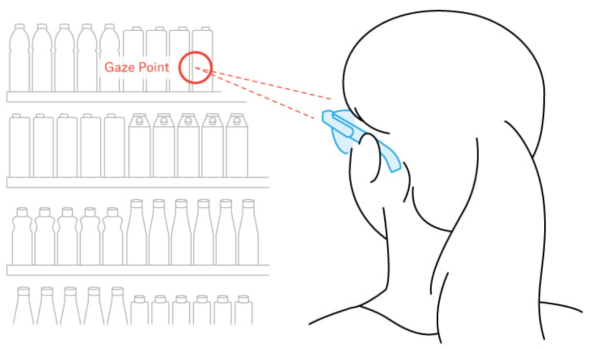

Figure 2.1.1: Here we see a gaze point[12]

izing brain lesion and breadth [9]. Eye movement have the potential to be used as diagnostic criteria [13], and this is an example of a area where we can use eye tracking to gain more information.

2.1.2 Eye-trackers

Most modern eye-trackers use video-based technology to track eye movements [9]. Infrared light is shone into the eye, which is invisible to humans, and produces a reflection on the cornea [9]. The eye-tracking software then identifies and tracks this reflection to determine the movement of the eye [9].

There are a wide variety of commercial eye-trackers, and when selecting an eye-tracker it should be based on the intended use and the features it offers [9]. The trackers vary in many of the features, and one of them is the speed of data collection. The sampling frequency is measured in Hertz (Hz), i.e. number of samples per second [14]. Sampling rate impct the precision of the measurements, and should be chosen based on the specific needs of the task at hand. Some require high-speed tracking of small saccades, while other may not require such a high sampling rate [14]. Most modern eye-trackers have sampling frequencies from 25 to 2000 Hz, so there is a wide range to choose from.

It is important to note that a higher sampling frequency does not necessarily mean a better eye-tracker. We can expect marginal benefit for increased frequency [14]. In some cases, a higher frequency may result in more noise and a larger amount of data that may not be beneficial. For tasks that simply require recording where participants looked, a lower sampling rate is acceptable [9].

Eye-tracking technology comes in various forms, including stationary and portable devices, and can be used for a range of purposes. Some eye-trackers require the use of a chin rest to stabilize the head, while others do not require this [9]. The use of a chin rest can be useful in controlled laboratory settings, where high levels of accuracy are required. However, the use of chin rest may be problematic in certain experiments, such as those involving natural head movement, or working with infants or children [9]. In these cases, eye-trackers that do not rely on a chin rest can still provide acceptable levels of accuracy for most purposes.

In addition to the head-mounted eye-trackers, there are also low-cost solutions that use web-cameras to extract and track eye features on the face [10]. This type of eye-tracking is often used in remote testing, allowing participants to use their computers without having to come to a specific location [10]. Webcam eye-tracking provides a convenient and accessible way to collect eye-tracking data, but has some limitations in terms of accuracy and sampling rate compared to standard infrared devices [10]. The sampling rate of webcam eye-tracking is relatively low, which can limit the accuracy of the data and its potential applications in research [10]. Despite these limitations, the potential for improvement and availability make webcam eye-tracking a valuable technology for studying human behavior and cognition. It is to expect that the algorithms used in webcam eye-tracking is continuously improving.

2.1.3 Challenges with Eye-Tracking

In order to conduct a successful eye-tracking experiment, it is crucial to consider the potential challenges that may occur. This can include issues with using the equipment, and how to do correct calibrations. By addressing these challenges from the start, researches can improve the quality of the data and ultimately, the quality of the study.

One of the main challenges in eye-tracking is ensuring proper calibration of the equipment. This is essential for collecting accurate and precise data [9]. Therefore, it is important for researchers to have a understanding of how eye-trackers work, and to test the equipment before starting the experiment. Additionally, problems can occur when connection the equipment for the first time, so it is crucial to have done this in advance to avoid any issues during the experiment. A pilot study should be done to assure this, and to assure that the eye-tracker work as intended for the experiment.

Another challenge within eye-tracking, when using eye-trackers with infrared light, is ensuring that the infrared light is able to reach the eye in sufficient quality [9].

If this does not happen, the tracking will not work well [9]. This can be caused by participants wearing glasses with strong prescription, or if the glasses are dirty. To prevent this, researchers should not allow participants to wear glasses during the experiment. Lenses could be a potential solution, or researchers could choose participants who do not need any corrective eyewear.

The infrared light can also be prevented from reaching the eye if the participant is too far away from the light source, or if the light is not pointed in the right direction [9]. The tracker should be adjusted before each new participant in the experiment to prevent this issue.

Point of gaze estimation can be difficult or impossible for the eye-tracker without the pupil, so factors that make the pupil hard to identify can reduce the quality of the data [15]. These factors could include partially occluded pupils, which can happen with sleepy participants [9]. Another factor is if there are other dark areas around the eye, such as dark eyelashes, which the camera may mistake for the pupil. To prevent these factors, participants should not wear excessive eye makeup or be too sleepy.

Overall, by understanding and addressing these challenges, researchers can conduct successful eye-tracking experiments and produce high-quality results. By considering potential pitfalls and addressing them before starting the experiment, researchers can ensure that their data is accurate and their research is robust, and it also assure good calibration.

2.1.3.1 Calibration

Ensuring the quality of data in an eye-tracking experiment is crucial for the success of the study. One important part of ensuring data quality is performing a proper calibration before starting the experiment. Calibration involves establishing a mathematical mapping between features in the eye, such as the position of the pupil and any corneal reflections, and the position of the calibration target [16]. However, the eye is never completely still during calibration, which can be a challenge [16]. To overcome this, it is important to choose the right time to sample the coordinates of the eye [16]. The steps involved in calibration are shown in Figure 2.1.2, and it is important to remember that it is possible to perform the calibration multiple times.

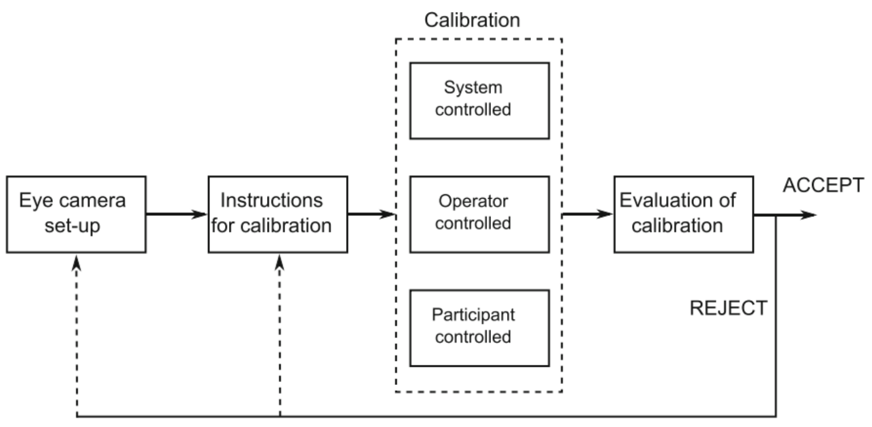

Figure 2.1.2: The steps to conduct calibration [16]

Data quality in eye-tracking is also evaluated using metrics such as accuracy and precision. Accuracy is the average difference between the location of the fixation target and the recorded gaze position [16]. Precision, on the other hand, refers to the eye-tracker's ability to reliably reproduce a measurement of a point of interest [16]. Figure 2.1.3 illustrates the difference and relationship between accuracy and precision.

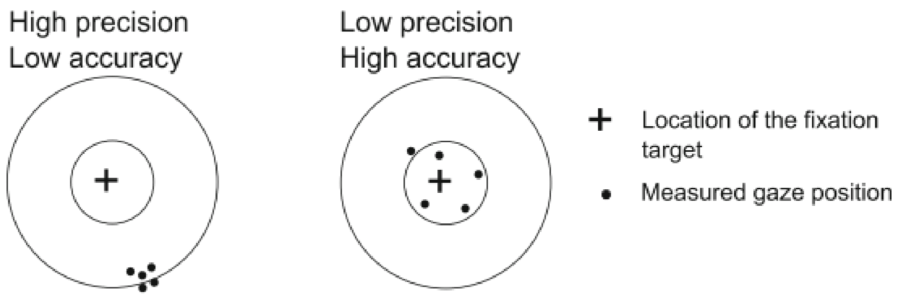

Figure 2.1.3: Difference between accuracy and precision [16]

2.1.4 Understanding the Eye

In order to conduct a successful eye-tracking study, it is important to have a basic understanding of the anatomy and physiology of the eye [9]. This includes knowledge of the different parts of the eye and how the parts work together to allow us to see. Understanding the structure and function of the eye and the visual field is crucial for interpreting the results of the eye-tracking experiments.

The eye is a complex organ that is responsible for gathering, focusing and transducing light [9]. One of the first structures that light encounters as it enters the eye is the pupil, which allows light to pass through [17]. Surrounding the pupil is the iris, a circular muscle that gives the eye its color and controls the amount of light that enters the eye and the intensity of the image [17]. Covering both the pupil and the iris is the transparent external surface of the eye, the cornea, which focuses the light on the retina and forms a sharp image. [9][17]. The retina, located in the back of the eye, contains the fovea centrails, a depressed structure that is responsible for fine detail and color vision. The fovea captures an image of what the eyes are currently pointed at [9]. The outer structure of the eyes are illustrated in Figure 2.1.4.

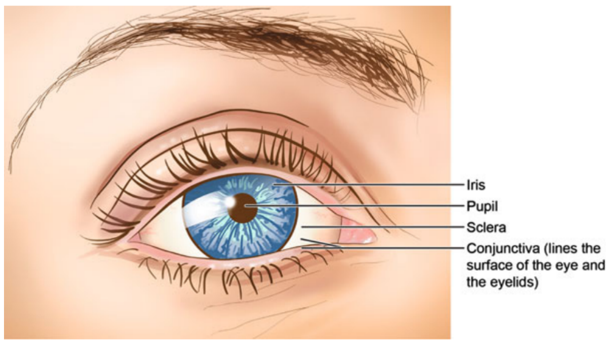

Figure 2.1.4: The outer structure of the eye [18]

It is also important to note the visual field of the eye. The fovea covers approximately 2° around the focal point of each eye, while the binocular visual field in humans extends about 130° vertically and 200° horizontally [12]. This visual field is illustrated in Figure 2.1.5, as well as the placement of the fovea.

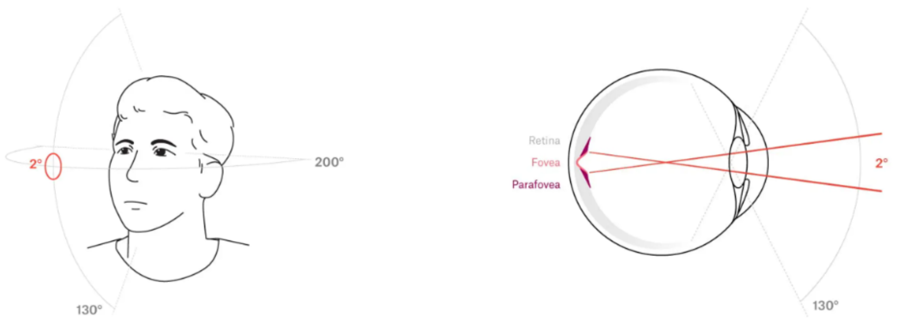

Figure 2.1.5: Visual field of a human [12]

2.1.4.1 Types of eye movements

The visual information we gather through the eyes are of the highest quality in a small area of the visual field, known as the foveal area [19]. In order to see things clear, we need to shift our gaze so that the object of interest falls within the foveal area. The different types of eye movements, can be divided into categories based on their purpose and characteristics.

Fixations are the most common type of eye movement, during which the eyes stop and scan a point of interest, holding the foveal area in one place [19]. Fixations typically last 180-330 milliseconds, and can provide valuable insights into cognitive processes and attention [9][19]. However, the foveal area is small, and the eyes are unable to gater high-quality information from the visual field in a single fixation [9]. As a result, the eyes must move frequently to gather information from different parts of the visual field. These movements, including saccades and smooth pursuit, are important to understand visual attention and cognition, and can be measured using eye-tracking technology.

Another type of eye movement is saccades, which is the rapid movement of the fovea from one point of interest to another [19]. During saccades, visual input is suppressed, meaning that our eyes are effectively blind when making a saccade [9]. This is due to the fast movement during saccades, which results in poor quality images on the retina [19]. After a saccade, a fixation typically follows [20]. The velocity and duration of saccades are directly proportional to the distance between the points of interest [9]. These movements can vary in size and duration, and saccades in scene perception are typically 5° of rotation and last 40-50 milliseconds [9]. These movements can also be tracked and provide information about how out eyes move form one point of interest to another. Figure 2.1.6 shows the eye movements during fixations and saccades, and how they are connected. The blue line illustrates the eye movement, and show how fixations and saccades works, and are connected.

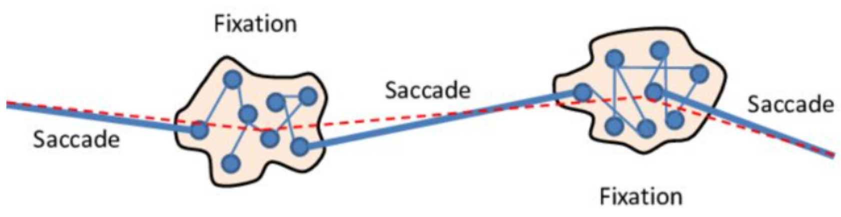

Figure 2.1.6: Illustration of eye movements [21]

When humans are viewing static objects, the eyes primarily exhibit saccadic and fixational movements, while the head remains relatively still [19]. In dynamic situations, the eyes make use of several movements in order to keep the fovea focused on a moving object of interest. These movements include vergence, smooth pursuit, and the vestibular ocular reflex [19]. Vergence refers to the convergence or divergence of the eyes as a visual target moves closer or further away [9]. This movement is slower than a saccade. Smooth pursuit is used when the eyes follow a moving target, maintaining alignment of the fovea with the target [9][19]. The vestibular ocular reflex is responsible for maintaining focus on a point of interest even when the head or body are in motion [19].

In addition to voluntary eye movements, there are also involuntary ocular motions [9]. For example, pupil diameter is modulated by the parasympathetic and sympathetic nervous systems [9]. The optokinetic response is a reflexive movement of the smooth pursuit of a point of interest as it moves through the environment, followed by a quick return of the eyes to their original position [9].

2.2 Theory Object Detection

This section aims to provide an introduction to the fundamental concepts in the field of object detection. Exploring these key concepts enhances the understanding of object detection technology and its relevance to the research conducted in this study.

2.2.1 Computer Vision

In the field of computer vision, digital computer techniques are used to extract, characterize, and interpret information from visual images of a three-dimensional world [22]. This involves the development of algorithms that enable computers to understand images and videos, in the same way human does. The computer is trained to identify patterns and objects in the data and use this knowledge to perform task such as facial recognition and object detection on visual data [23].

Computer vision involves the use visual data, machine learning and deep learning techniques to analyze images and videos [24]. These techniques are used to train the computer on a large amounts of data, enabling it to learn to identify patterns independently. An example of computer vision application is automated driving, where the computer is tasked to recognize objects such as other cars or pedestrians, and then perform an action, such as use the brake [24].

2.2.2 Machine Learning

Machine Learning and Deep learning are well known topics within the field of Artificial Intelligence. The definitions and connections between these concepts are illustrated in Figure 2.2.1, the definitions are also presented below.

Artificial Intelligence:

> *"A program that can sense, reason, act and adapt"*

Machine Learning:

> *"Algorithms whose performance improve as they are exposed to more data over time"*

Deep Learning:

> *"Subset of machine learning in which multilayered neural networks learn from vast amount of data"*

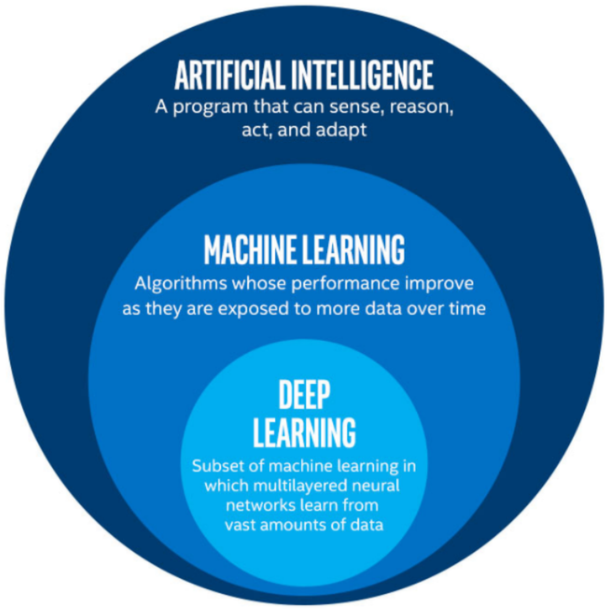
ARTIFICIAL INTELLIGENCE
A program that can sense, reason, act, and adapt
MACHINE LEARNING
Algorithms whose performance improve as they are exposed to more data over time
DEEP LEARNING
Subset of machine learning in which multilayered neural networks learn from vast amounts of data

Figure 2.2.1: Artificial Intelligence, Machine Learning and Deep Learning [25]

Machine learning, a subfield of artificial intelligence, has become an important tool in the development of advanced models capable of extracting insights from large and complex data [26]. Machine learning use algorithms that adapt and improve their performance based on the data, without any explicit programming [26]. These algorithms make it possible for machines to identify patterns, make predictions and solve problems across different fields, such as computer vision [26]. Supervised learning, unsupervised learning, and reinforcement learning are three primary categories of machine learning algorithms, each with distinct methodologies [27].

2.2.2.1 Supervised Learning

Supervised learning trains models using labeled input-output data pairs, enabling the model to make predictions for new, unseen inputs based on the learned relationship between inputs and outputs [27]. It is a frequently used method for making predictions. For instance, when predicting house prices, supervised learning employs features such as house area, number of bedrooms, and local facilities, to name a few, alongside the house price [28]. With this information, the model finds patterns in the features connected to the house price. Supervised learning addresses two categories of problems: classification and regression. In classification, data is categorized into distinct classes, for example, predicting a student's pass or fail outcome based on their past performance [28]. Regression, in contrast, deals with output variables that are real or continuous values, such as predicting the probability of a student's score [28].

2.2.2.2 Unsupervised Learning

In unsupervised learning, models are given unlabeled data and must identify patterns or structures without any guidance. Instead of being fed with labeled data and clear target outcomes, the unsupervised learning model examines the raw, unsorted data for underlying patterns and structures [28]. This method is particularly beneficial for projects where the desired output or specific data features aren't clearly identified, allowing for the detection of unknown similarities and differences in the data [28]. Unsupervised learning techniques, such as K-means clustering, neural networks, and principal component analysis, have been effectively used for various applications, like categorizing users based on their social media behavior [28].

2.2.2.3 Reinforcement learning

Reinforcement learning, on the other hand, enables algorithms to self-learn within an environment, independent of labeled data or training sets, distinguishing it from supervised or unsupervised learning [28]. This technique is widely used in gaming, robotics, and numerous other fields, using a learning agent that operates within start and end states, possibly traversing multiple paths [28]. An agent might attempt to influence its environment and move from one state to another. Successful actions get rewarded, while failures go unrewarded [28].

2.2.3 Deep Learning

Deep learning is a subfield of machine learning that aims to create and train neural networks with three or more layers, enabling them to learn and identify complex patterns and features within the data [29]. Figure 2.2.2 illustrates a deep neural network. Machine learning algorithms are known for relying more on structured, labeled data to make predictions, while deep learning algorithms can ingest and process unstructured data, such as text and images [29]. Deep learning automates feature extraction, removing some of the dependency on humans, while in machine learning, the hierarchy of features is established manually by a human [29]. Through gradient descent and backpropagation, the deep learning algorithm adjusts and fits itself for accuracy, allowing it to make predictions about new data with increased precision [29].

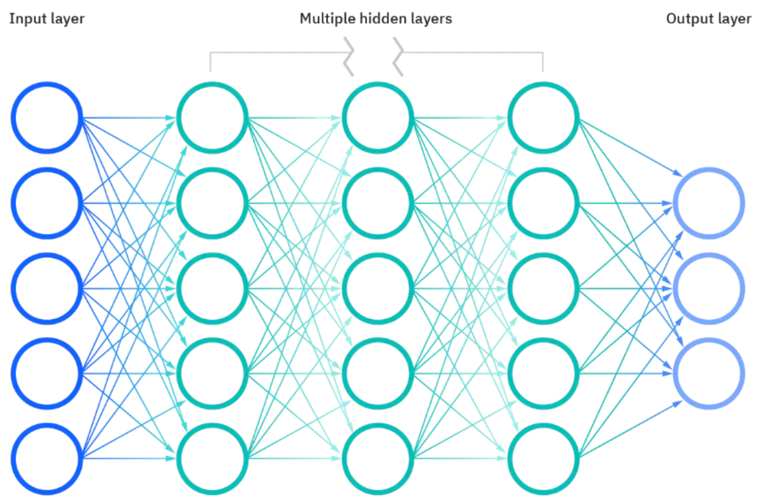

Figure 2.2.2: Deep neural network [30]

Deep learning neural networks attempt to replicate the functioning of the human brain using data inputs, weights, and bias [29]. The networks are made up of multiple layers of interconnected nodes that build upon each other to improve predictions or classifications [31]. The progression of computations through the network is called forward propagation, and the input and output layers are known as visible layers [29]. Backpropagation is another process that uses algorithms like gradient descent to calculate prediction errors and adjust weights and biases to train the model [31].

Deep learning algorithms are complex, and there are different types of neural networks designed for specific datasets [29]. For example, Convolutional Neural Networks (CNNs) detect features and patterns in images for tasks like object detection, while Recurrent Neural Networks (RNNs) process sequential or time series data for example in speech and natural language recognition applications [29].

2.2.4 CNN

Convolutional Neural Networks (CNNs) are a type of neural network used for analysing images and videos [32]. Unlike classical machine learning methods, CNN does not require any task-specific feature engineering and has been proven to produce high accuracy in image classification tasks [33]. It has become increasingly popular in the field of deep learning due to the ability to process data with grid-like topology, such as images or time-series data [34]. This method use a mathematical operation called convolution, which is a specialized form of linear operation, to analyse the data [31].

CNN architecture consists of two stages, hierarchical feature extraction and classification [35]. A typical CNN architecture, illustrated in Figure 2.2.3 , takes an input image and applies trainable filters (called kernels) to produce feature maps [35]. These feature maps are then divided into equal-sized, non-overlapping regions, and the maximum (or average) of each region is passed to the next layer (sub-sampling layer), resulting in smaller but still deep feature maps [35]. This process makes the network more robust to translations [35]. The convolution and subsampling steps are repeated for multiple iterations, and the resulting feature maps are then connected to a fully connected layer to perform classification [35].

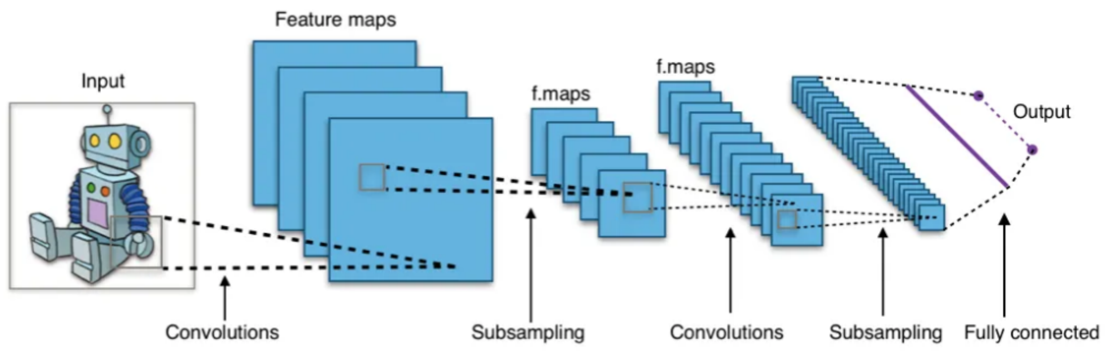

Figure 2.2.3: Convolutional Neural Network Architecture [36]

2.2.4.1 The Convolutional Layer

The convolutional layer is an important part of a CNN. It performs the operation of convolution between the input image and a set of learnable filters, also called kernels [35]. The filters are typically small, square matrices with weights that are adjusted during the training process to extract important features from the input image [35].

During the convolution operation, the filter is slid across the input image, and each position, the element-wise multiplication between the filter and the image is computed [35]. The resulting products are summed, and the result is stored in the output feature map [35]. By stacking multiple filters, the convolution layer can extract a variety of different features from the input image [35].

The convolutional layer extracts important features from the input image and produces an output feature map that is smaller than the input image. To maintain the size of the input image, zero-padding can be added to the edges [35]. During the training process, the weights of the filters are updated using back-propagation to learn the most important features [35]. This allows the CNN to perform tasks such as image classification, object detection, and segmentation. The convolutional layer is a critical component of CNNs and is essential for image processing and computer vision applications [35].

The convolution layer takes an input image of size $W_1 \times H_1 \times D_1$ from the previous layer, where W_1, H_1 and D_1 represent the image's height, width and depth [35]. K filters with a shape of $F \times F \times D_1$ are defined, where F is the kernel size. The convolution of the input volume and filters produces an output volume of size $W_2 \times H_2 \times K$, where the values of W_2 and H_2 depend on the filter size, stride, and pad settings of the convolution operation [35]. The formula for calculating the output size, W_2 (width) and H_2(height), is as follows:

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$

In these formulas, K is the filter size, P is the padding and S is the stride. An example of 2D convolution with a $7 \times 7 \times 1$ input volume convoluted with one 3×3, with 0 padding and 1 stride, produced a $5 \times 5 \times 1$ output volume [35], as shown in Figure 2.2.4.

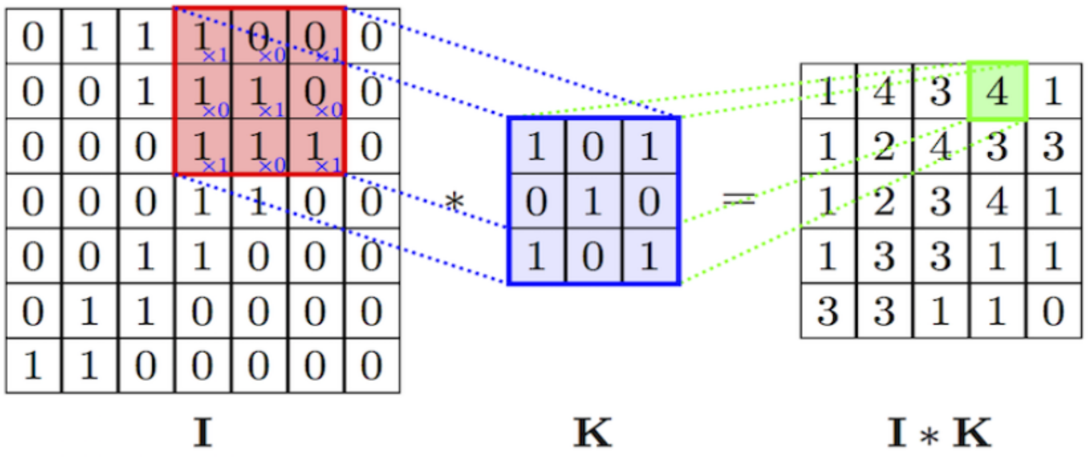

Figure 2.2.4: Convolution operation in 2D [35]

2.2.4.2 The Pooling Layer

The pooling layer in CNNs is an essential component that enables non-linear down-sampling of the input image [35]. The layer divides the image into non-overlapping rectangles and outputs the maximum value for each region (in max-pooling) [35]. This technique helps to reduce the dimension of the representation, which in turn reduces the number of parameters and computations required in the network, while also preventing overfitting [35]. The pooling layer can be implemented using various non-linear functions such as max-pooling, average-pooling, and stochastic-pooling [35]. This layer operates independently on each depth slice of the input and adjusts its dimensions, while keeping the structural layout of the original image. Additionally, the pooling layer performs a translation-invariance operation, making the network more robust to small translations in the input image [35].

The pooling layer in CNN takes an input volume of size $W_1 \times H_1 \times D_1$ and produces an output volume of size $W_2 \times H_2 \times D$ [35]. The dimensions of the output width and height, W_2 and H_2 respectively, are determined by the kernel size, stride and padding settings, as illustrated in Figure 2.2.5 . The output dimensions can be calculated as follows:

$$W_2 = \frac{W_1 - F}{S} + 1$$

$$H_2 = \frac{H_1 - F}{S} + 1$$

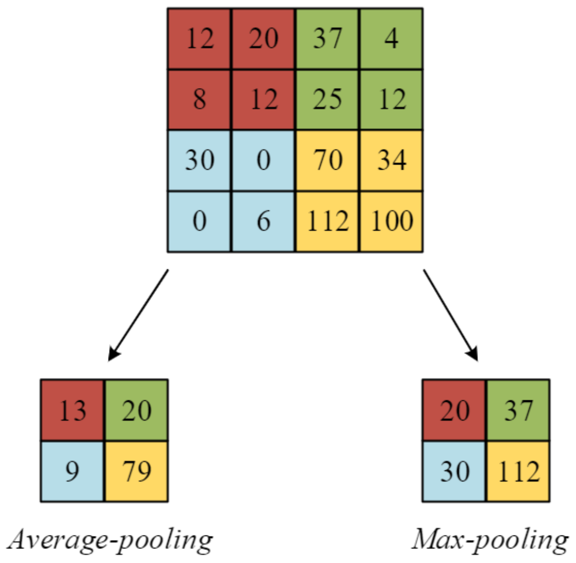

Figure 2.2.5: Pooling with 2 × 2 filer and a stride of 2 [35]

2.2.4.3 Fully Connected Layer

This layer take the output from the previous layer and flattens it into a 1D vector [37]. This vector is then passed through a fully connected layer, which applies a set of weights to the input vector and produces an output vector of arbitrary length [37]. This layer typically provides the final classification decision for the network [37].

During training, the weights of the filters in the convolutional layer and the fully connected layer are updated through backpropagation to minimize the loss function [37]. This process of adjusting the weights allows the CNN to learn the most important features of the input image and can be used for a variety of tasks such as image classification, object detection, and segmentation.

2.2.5 Object Detection

Object detection is a task in computer vision that involves identifying the categories and locations of object instances in an image or video [38]. It has various applications in fields such as surveillance, autonomous driving, and medical decision making [39]. The first generation of object detection methods relied on hand-crafted features and linear classifiers [39]. However, with deep learning current object detection methods use deep neural networks to extract features and classify objects [39]. This has led to significant improvements in performance compared to earlier methods.

There are two main approaches in object detection: top-down and bottom-up. The top-down approach tries to identify whole objects first, while the buttom-up approach focuses on smaller parts of objects and then puts them together to identyfy the whole object [40]. Most object detection methods use the top-down approach, which can be further divided into two-stage and one-stage methods [40]. Two stage methods try to reduce the number of objects to be identified, while one-stage methods try to identify objects directly from pictures [40]. YOLO is commonly used in object detection tasks, and are an example of the one-stage top-down approach.

2.2.6 YOLO

YOLO was published in 2016 by Joseph Redmon et al. [41]. The name YOLO stands for "You Only Look Once", reflecting to its ability to accomplish the detection task with a single pass of the network [42]. This is unlike previous approaches that either used sliding windows followed by a classifier that needed to run hundreds of times on each image, or the more advanced methods that divided the tasks into two steps [42]. In the two-step approach, the first step detects possible regions with objects or regions proposals, and the second step run a classifier on the proposals [42]. YOLO works by predicting the bounding boxes and class probabilities directly from full images in a single feedforward pass of a convolutional neural network [41]. This model was the first to perform object detection using a single neural network on the entire image in one evaluation [41]. YOLO partitions the input image into a grid of S $\times$ S cells [41]. Every grid cell predicts bounding boxes and confidence scores. The confidence scores measure both how confident the model is that the boxes contains an object and also the precision in predicting the boxes accurately, this process is illustrated in Figure 2.2.6. Since the first release of YOLO, it has been updated with newer versions, with the latest being YOLOv8 which was released in 2023.

2.2.6.1 YOLOv8

YOLOv8 was released by Ultralytics in January 2023 [43], and builds upon previous versions by incorporating several improvements. Its architecture includes a backbone, head, and neck. One of the key features of YOLOv8 is the use of Darknet-53 backbone architecture, a convolutional neural network with 53 layers that can classify images into 1000 object categories [44]. YOLOv8 divides an image into a grid of smaller regions and predicts a bounding box and class probabilities for each region [44]. Using a anchor-free detection head, YOLOv8 predicts bounding boxes with higher precision and speed, owning to the large feature map

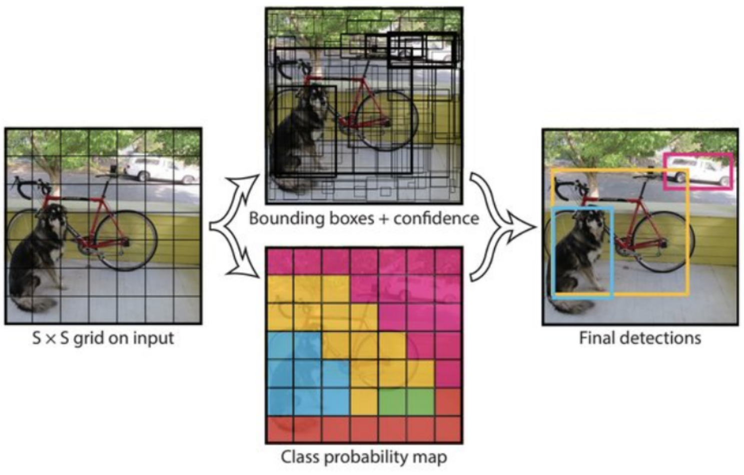

Figure 2.2.6: Bounding boxes and class prediction in YOLO [41]

and improved convolutional network [45]. To improve accuracy and robustness, YOLOv8 incorporates a technique called Pseudo Supervision that uses multiple models with different configurations during the training process [44]. These models are trained on the same dataset with different hyperparameters, leading to a more diverse set of predictions [44]. YOLOv8 provides five scaled versions, YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x (extra large) [42].

2.2.7 Person Re-Identification

Person re-identification (re-ID) is a challenging task in computer vision that aims to match people across different camera views. Re-ID is challenging because people can look different depending on the camera angle and the clothes they wear, which makes it hard for computers to match them [46]. There are two big problems in re-ID, the first one is that people can look different even if they are the same person (intraclass variations), and the second problem is that people can look similar even if they are different people (small interclass variations) [46]. To solve this problem, researchers need to find ways to recognize people based on different parts of their appearance, like their shoes or glasses, and different parts of their body, like their face or clothes [46]. One way to do this is by using deep learning, that can learn to recognize people by looking at lots of different images of them.

2.2.7.1 OSNet

OSNet (Omni-Scale Network) is a neural network architecture designed for person re-identification, which is the task of recognizing individuals across different frames in a video [46]. The OSNet architecture is designed to learn a representation of features that is able to handle changes in lighting, body position, and other factors that can impact how a person looks [46]. The OSNet architecture is made up of a network that extracts information from the input image, followed by a layer that combines this information into a fixed-length vector [46]. The features are then reduced in size and a classifier is added to differentiate between different people. Zhou et al. [46] who wrote the original paper show that it performs better than similar models in several benchmark datasets for person re-identification, including Market-1501, DukeMTMC-reID, and MSMT17 [46]. OsNet has been used in combination with other models such as YOLOv8, for real-time multi-object tracking.

2.3 Theory Gaze Prediction Model

This section goes into the theory needed to understand the work on the model predicting gaze points, aiming to provide a thorough understanding of the concepts involved. Through the exploring of these theoretical aspects, valuable insights are gained into the techniques used in predicting gaze points. Topics like machine learning, supervised learning, and deep learning are also relevant for this section, but since they were discussed in the previous section, they will not be further addressed.

2.3.1 Recurrent Neural Network

The Recurrent Neural Network (RNN) is a popular type of deep learning that is well-suited for handling sequential data [47]. Unlike traditional deep learning networks, RNNs integrate recurrent neurons that allow the output signal to be fed back to the input [47]. This unique architecture make it possible for RNNs to remember past information and learn long-term dependencies within the sequential data [47]. Unlike other algorithms, recurrent neural networks can develop a deeper understanding of the context and patterns within a sequence, making it possible for more informed predictions [48].

To understand the concept of RNNs, it is helpful to have a basic understanding of the nature of sequential data. Sequential data refers to ordered information where elements are arranged in a specific sequence [48]. One of the most common types of sequential data is time series data, which consists of a sequence of data points arranged in chronological order [48].

In a usual feed-forward neural network, information flows in one direction, moving from the input layer through the hidden layers and finally the output layer [48]. The network processes the information in a straightforward manner, without any memory of previous inputs. As a result, feed-forward networks lack the ability to predict future events and do not consider the sequential order of the data [48].

On the other hand, RNNs use a cycles structure that allows information to loop back within the network [48]. This looping gives RNNs the capability to consider both the current input and the knowledge from previous inputs when making decisions [48]. By maintaining an internal memory, RNNs can capture temporal dependencies and effectively model sequential data [48]. Figure 2.3.1 illustrates the difference between a recurrent neural network on the left, and a feed-forward neural network on the right.

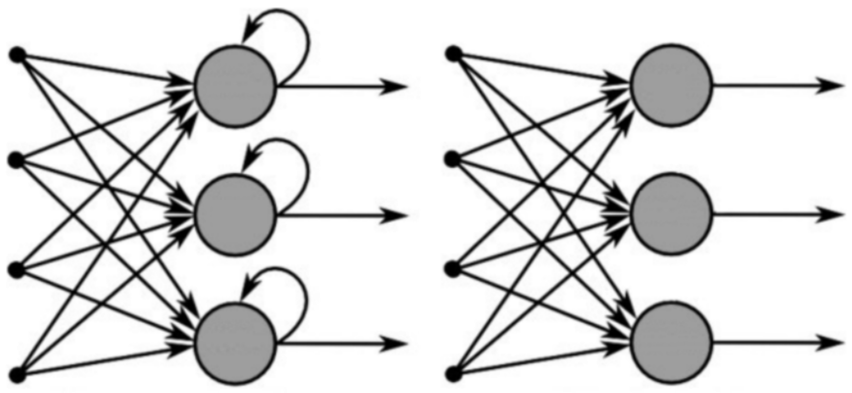

Figure 2.3.1: Recurrent Neural Network and Feed-Forward Neural Network [48]

While RNN offer a good solution for sequential data, one common issue with traditional RNN architectures is the challenge of training, primarily due to problems such as gradient explosion or vanishing gradients [49]. To handle these challenges, the long short-term memory (LSTM) network was introduced.

2.3.2 Long Short-Term Memory Network

Long Short-Term Memory (LSTM) network is a type of recurrent neural networks (RNNs), which are characterized by having an internal loop in the neuronal connection architecture [50]. The network was introduced by Hochreiter and Schmidhuber [49] in 1997, and have continually been refined over the years. LSTM are designed to remember patterns over longer periodes, and they solve the common problems with standard RNNs, the problems known as “vanishing and exploding gradients” [50].

The LSTM network is build up by an input layer, one or more memory cells (a type of neuron in the hidden layer), and an output layer [50]. These memory cells store data, while different types of gates manipulate this data. LSMT have three types of these gates, being forget gate, input gate and output gate [50].

The forget gate, input gate, and output gate all serve essential roles in the functioning of an LSTM cell. The forget gate identifies and remove irrelevant information from the cell state, ensuring only meaningful data is retained [51]. The input gate then evaluate new incoming information and decides what is important to add to the cell [51]. Lastly, the output gate determines the next output based on the data currently stored in the cell state [51].

The control gates and memory cell facilitate the flow of information in a LSTM model, which can get into, stay in or be read from the cell [52]. This functionality is determined by the following equations.

$$\text{Input gate: } i_t = \sigma(W_{ih}h_{t-1} + U_{ix}x_t + b_i) \tag{2.1}$$

$$\text{Forget gate: } f_t = \sigma(W_{fh}h_{t-1} + U_{fx}x_t + b_f) \tag{2.2}$$

$$\text{Cell state: } C_t = f_t \times C_{t-1} + i_t \times \tanh(W_{ch}h_{t-1} + U_{cx}x_t + b_c) \tag{2.3}$$

$$\text{Output gate: } o_t = \sigma(W_{oh}h_{t-1} + U_{ox}x_t + b_o) \tag{2.4}$$

$$\text{Output vector: } h_t = o_t \times \tanh(C_t) \tag{2.5}$$

The σ is the activation function, the $\times$ operator represents element-wise vector multiplication, x_t is the input vector (including forcings and static attributes) for time step t, W represents the network weights, b is the bias parameter, y is the output to be compared with observations, h is the hidden state, and s represents the cell state of memory cells, which is a distinctive feature of LSTM [50].

LSTM architectures can be implemented with a single hidden layer or multiple hidden layers [52]. While an LSTM network with multiple hidden layers may improve forecasting performance under specific conditions, it also adds complexity, becomes more challenging to track, and may lack interpretability [52]. The architecture of both RNN and the original LSTM network are illustrated in Figure 2.3.2.

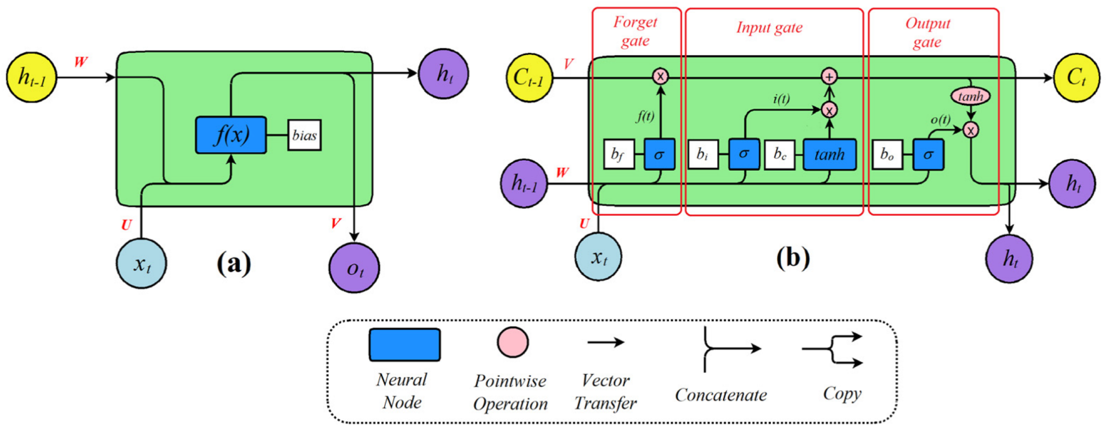

Figure 2.3.2: a) RNN and b) orginal LSTM architecture [52]

2.3.3 Loss Function

Loss functions quantify the difference between the predicted output and the true target values, providing a measure of how well the model is performing [53]. The choice of loss function can significantly impact the training process and the model's performance.

In supervised learning problems, the training set consist of n labeled samples $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where x_i represents the features of sample i and y_i represents the corresponding label [54]. The objective is to develop a model that accurately predicts the labels based on the features [54].

To measure the difference between the model's predictions and the true labels, a loss function is used. The loss function calculate the prediction error for an individual sample by comparing the model's predicted value ($\hat{y}$) with the true value (y) [54]. During the model training the loss is calculated and from this, the weights are adjusted in the network to minimize the loss [54]. The goal is to find the optimal parameter values that gives the smallest value from the loss function [54].

In regression problems, where both the target label and the model's prediction are continuous values, different loss functions can be used to measure the difference between the predicted and true values [54]. Among the commonly used loss functions in these problems, there are the mean squared error. Mean squared error (MSE) is defined as the squared difference between the target label and its predicted value, and the function is given below [54].

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2.6}$$

Mean squared error calculates the average squared difference between the predicted and true values [54]. The loss function is well-suited for problems where the goal is to minimize the overall squared error [54].

2.3.4 Optimization Algorithms

Optimization algorithms is important when it comes to improving the performance of deep learning models. These algorithms widely impact both the accuracy and speed training of the models [55]. During the training process of a deep learning model, optimizers are responsible for adjusting the weights and minimizing the loss function at each epoch [55]. An optimizer is an algorithm that modifies the attributes of the neural network, such as weights and learning rates [55]. To select appropriate weights for a deep learning model can be challenging, considering the large number of different parameters involved [55]. Therefore, choosing the right optimization algorithm becomes important for each specific model [55].

"Adam" is an optimization algorithm used for improving deep learning models. It is specifically designed for optimizing stochastic objective functions by using first-order gradients [56]. "Adam" can handle large-scale problems with a large amount of data and parameters, and is well-suited for scenarios with dynamic objectives or noisy gradients [56]. It is an optimization method that adapts learning rates for different parameters using estimates of the first and second moments of the gradients [56]. "Adam" combines the benefits of "AdaGrad", which handles sparse gradients effectively, and "RMSProp", which performs well in online and non-stationary scenarios [56]. These two are other examples of optimization algorithms.

2.3.5 Dropout

Deep neural networks that consist of a large number of parameters are highly effective within the field of machine learning. However, overfitting is a common problem. This is a concept where the model learns the training data too well, and performs poorly on unseen data [57]. Essentially, the model becomes too complex as it tries to capture all the noise or variations in the training set, causing it to perform poorly when presented with new data [57]. Dropout was first introduced by Srivastava et al. [58], and is a strategy implemented to handle this issue.

The idea behind dropout is to randomly eliminate units, along this their connections, from the network during the training process [58]. This technique handle the issue of units becoming overly reliant on one another [58]. Throughout the training phase, dropout essentially draws samples from an exponential range of different “thinned” networks [58]. During the testing phase, it becomes straightforward to mimic the effect of averaging predictions from all these ’thinned’ networks, by merely using a single “unthinned” network that has smaller weights [58]. This practice helps to prevent overfitting.

2.3.6 Evaluation Metrics

Evaluation metrics are a crucial part of any machine learning model. They provide an insight into the performance of the model, helping to identify the strengths and weaknesses of the model, which can help to do further improvements [59].

Different machine learning models require different types of evaluation metrics. In classification problems metrics like accuracy, precision, recall, or F1 score are used [60]. However, the problem in this study is a regression problem, where the goal is to predict a continuous output. The four most common metrics for problems like this are presented below.

2.3.6.1 Mean Squared Error

Mean Squared Error (MSE), calculates the average of the squared differences between the original and predicted values [60]. An important advantage of MSE is that it simplifies the computation of the gradient [60]. Since the error is squared in MSE, larger errors have more significant impact compared to smaller error [60]. MSE is calculated using the following formula:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2.7}$$

Where n is the number of observations, Y_i is the actual observation, $\hat{Y}_i$ is the predicted observation, and $\sum_{i=1}^{n}$ denotes the summation over all observations from $i = 1$ to $i = n$ [60].

2.3.6.2 Mean Absolute Error

Mean Absolute Error (MAE) represent the average of the absolute differences between the original and predicted values [60]. It provides a measure of how much the predictions deviate from the actual results [60]. However, MAE does not indicate the direction of the error, meaning it doesn't inform us whether the predictions are consistently overestimating or underestimating the actual values [60]. MAE is calculated using the formula below.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i| \tag{2.8}$$

where n is the number of observations, Y_i is the actual value of the observation, $\hat{Y}_i$ is the predicted value for the observation, $|Y_i - \hat{Y}_i|$ represent the absolute value av the difference between the actual and the predicted value, and $\sum_{i=1}^{n}$ denotes the summation over all observations from $i = 1$ to $i = n$ [60].

2.3.6.3 Root Mean Squared Error

Root Mean Squared Error (RMSE) is computed as the square root of the average of the squares of all the errors [61]. RMSE is widely used and is considered an exceptional general purpose error metric for numerical predictions [61]. The formula for RMSE is given below.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} \tag{2.9}$$

In this formula, n represents the total count of observations, Y_i corresponds to the actual observation, while $\hat{Y}_i$ is its predicted value, $\sum_{i=1}^{n}$ indicates the summation operation applied across all observations, ranging from $i = 1$ to $i = n$ [61].

2.3.6.4 The R^2 Score

The R^2 Score, also known as the coefficient of determination, is a crucial evaluation metric for regression-based models [62]. It measures how well the model's prediction align with the actual values by evaluation the discrepancy between the dataset samples and the model's predictions [62]. A score of 1 signifies a perfect fit, while a score of 0 suggest that the model's predictions are bad on unseen data [62]. The R^2 score are calculated as shown below.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} \tag{2.10}$$

Where n is the total number of observations, Y_i is the actual value for the observations i, while $\hat{Y}_i$ is the predicted value for the observations, $\bar{Y}$ is the average of the actual values, and $\sum_{i=1}^{n}$ indicates the summation operation applied across all observations, ranging from $i = 1$ to $i = n$ [62].

CHAPTER

THREE

DATA COLLECTION

In order to conduct the data collection, an experiment is designed. In this experiment participants is presented with a video and their eye movements will be tracked and recorded using an eye-tracker.

3.1 Apparatus

The hardware included in the experiment setup are listed below, with specifications.

- Graphics Card: NVIDIA GeForce GTX 760
- Tv Screen: Samsung 65" Q70A QLED 4K Smart TV
- Monitor 1: Philips 241S4 is a 24-inch IPS monitor with a resolution of 1920 $\times$ 1080 pixels. It features a 60 Hz refresh rate.
- Monitor 2 and 3: BenQ G2420HD monitor features a 24-inch display with a resolution of 1920 $\times$ 1080 pixels and a refresh rate of 60 Hz.
- Eye-tracker: Pupil Core, with a world camera featuring a resolution of 1280x720 pixels and a frame rate of 30 Hz, and an eye camera with a resolution of 192x192 and a frame rate of 120 Hz. The Pupil Core has a gaze accuracy of 0.6 degrees and a gaze precision of 0.02 degrees.

3.1.1 Pupil Core

Pupil Core is an eye-tracking headset that uses one world camera and two infrared spectrum eye cameras for dark pupil detection [63]. The cameras are connected to a host monitor via USB, and the video streams are read using the Pupil Capture software for real-time pupil detection, gaze mapping, recording, and other functions [63]. Pupil Core provides gaze data within the field of view of the world camera, allowing the wearer to move and look freely within their environment while still enabling accurate gaze analysis [64]. The headset is shown in Figure 3.1.1.

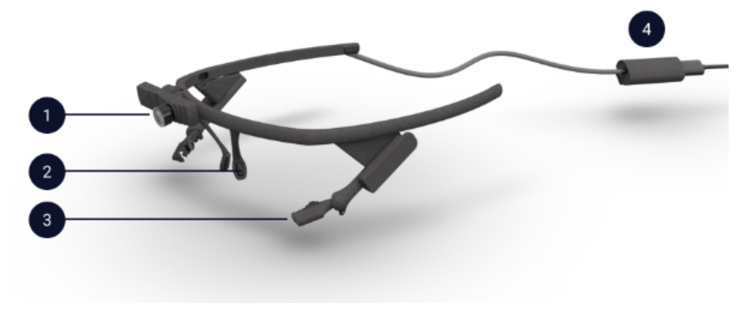

Figure 3.1.1: Pupil Core Eye-tracker [65]

The parts of Pupil Core shown in Figure 3.1.1 are listed below:

1. World camera
2. Nose support
3. Eye cameras, adjustable
4. USB-C connector clip: Connect Pupil Core headset to the computer through USB-C to USB-A cable [65].

All eye-tracking data was collected using Pupil Capture (Version 3.5.1). Pupil Capture runs in real-time to capture and process images from the camera video streams [63]. Pupil Player (version 3.5.1) was used to playback and visualize video and gaze data recorded with Pupil Capture [63]. In order to do an screen-based eye-tracking experiment, PsychoPy will be used. PsychoPy is an application that allows researchers to run a wide range of experiments in the behavioral sciences [66].

3.2 Participants

The participants in this study were recruited from the Department of Mechanical and Industrial Engineering at the Norwegian University of Science and Technology. The group consisted 14 participants, 5 male and 9 female adults between the ages of 23 and 26, all of whom were in their final year of their master's degree program. None of the participants had any prior experience with eye-tracking, and the only inclusion criteria was that they did not use glasses, as this could potentially have an impact on the data quality.

3.3 Designing Eye-Tracking Experiment

Before performing the experiment with the Pupil Core eye-tracker, the stimuli that the participant would be presented had to be created. To make the experiment, PsychoPy will be used. PsychoPy is an application that allows researchers to run a wide range of experiments in the behavioral sciences [66]. Figure 3.3.1 illustrates the flow of the experiment. First, the participant will see a welcome message, which is followed by calibration and validation of the eye-tracker. The validation is performed after the calibration to check the accuracy. After this, the participant receives instructions informing that they are going to watch a video, and that they should look as they feel natural, and also that they can move their head freely. During the presentation of the video, data from the eye-tracker will be recorded. Finally, a concluding message will be shown to the participant, allowing them to remove the eye-tracker and ask any questions they may have.

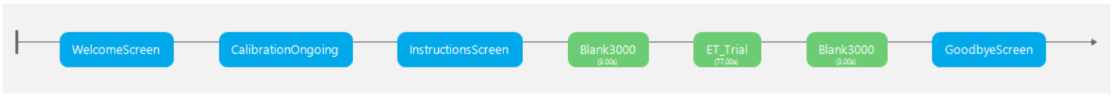

Figure 3.3.1: The flow of the experiment

To check if the experiment routine worked properly and if the instructions were clear, pilot experiments were conducted. These pilot experiments aimed to determine if participants understood the instructions as intended and enable the examiner to identify any necessary additions or modifications to the procedure.

3.3.1 Experiment Setup

The experiment will be conducted in an experiment box build for eye-tracking experiments. This is a box with a large TV screen on one of the walls. Inside of the box there is a table, a chair, and an monitor the examiner can use while placing the eye-tracker, to make sure it is placed correct. This monitor show the view of both eye cameras, and make it possible to adjust the eye-tracker for each participant without leaving the box. It is important that these cameras are placed correctly, and adjusted for each participant, to get good data quality. The table hides this monitor, so it won't be a distraction for the participant during the experiment. Figure 3.3.2 shows the inside of the experiment box.

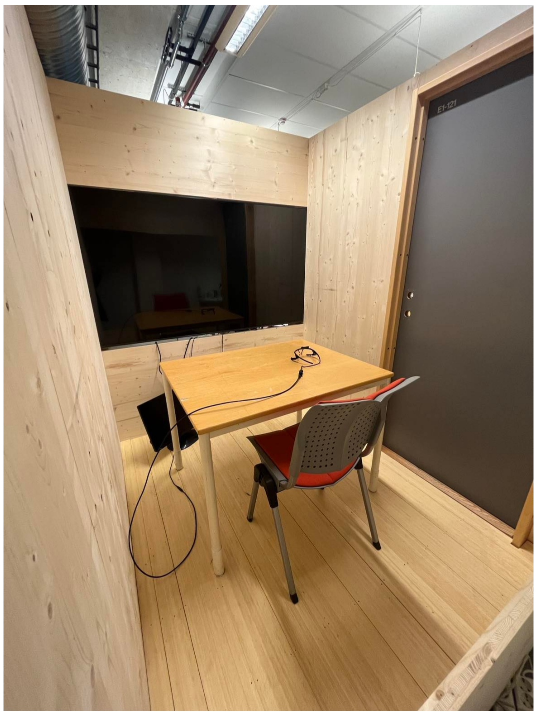

Figure 3.3.2: The experiment box

The chair is positioned at a specific distance to ensure that the entire TV screen remains within the participant's field of view at all times. This is necessary because all four corners of the screen must remain visible within the field of view that is being recorded by the eye-tracker.

The experimental box is designed to eliminate any distraction within the participant's field of view, ensuring them to focus on the TV screen only. Additionally, the use of a large screen is advantageous, as the video stimulus will occupy a grater portion of the participant's visual field, causing the eyes to move over larger distances. This can be helpful in identifying any patterns or trends within the data. The box is also placed in a room where there are no sound, which also could have been an distraction. As a result, the experimental box was carefully constructed to ensure that the participant's attention is fully directed toward the TV screen during the experiment.

A visual representation of the entire experimental setup, including both inside and outside of the box, are shown in Figure 3.3.3.

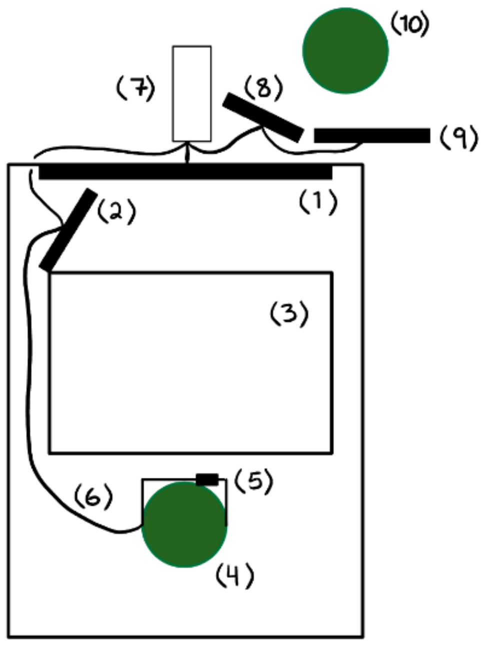

Figure 3.3.3: The setup for the experiment

The different components of the setup are listed below. Component 1-6 are placed inside the experiment box, while components 7-10 are placed outside of the box.

1. TV screen
2. Monitor 1 to see eye-tracker placement
3. Table
4. Participant
5. Pupil Core eye-tracker
6. USB to connect eye-tracker and host monitor
7. Workstation, with graphics card
8. Monitor 2 for examiner to observe
9. Monitor 3 for examiner to observe
10. Examiner

3.3.2 Video for stimili

The video chosen for the stimuli is a video made by Massey University - University of New Zealand found on YouTube [67]. The video is in 360 view, which means that it is recorded using a special type of camera that captures a full 360-degree view of the surrounding environment. This allows the viewer to look around and see everything in the video, as if they were present in the environment where the video was recorded. It was not possible to use a 360-view in this experiment, and therefore a screen recording of the desired view of the video was used instead. Specifically, the screen recording is 1 minute and 17 seconds in duration, and extracted from the original video, starting at the 1 minute and 48 seconds mark. The duration was selected to provide a natural beginning and end to the scene, and also ensuring that the video's length would sustain the participant's attention.

The focus of this experiment lies in observing eye responses to visual stimuli. To maintain focus, the experiment design includes video stimuli without any sound, which is eliminating any potential reactions based on sounds.

The objective of this study is to add eye movement functionality into the manikin. The selected video captures a patient's perspective from a bed, with a person conducting an examination. This particular video was selected for its relevance to this study. Situations such as the one in the video is an example where it is wanted for the manikin to have the ability to determine where to direct its gaze. By analyzing and studying such scenario, valuable insights can be gained to enhance the realism and effectiveness of the eye movement implementation in the manikin. Figure 3.3.4 shows a frame from the video, providing an illustration of the captured point of view.

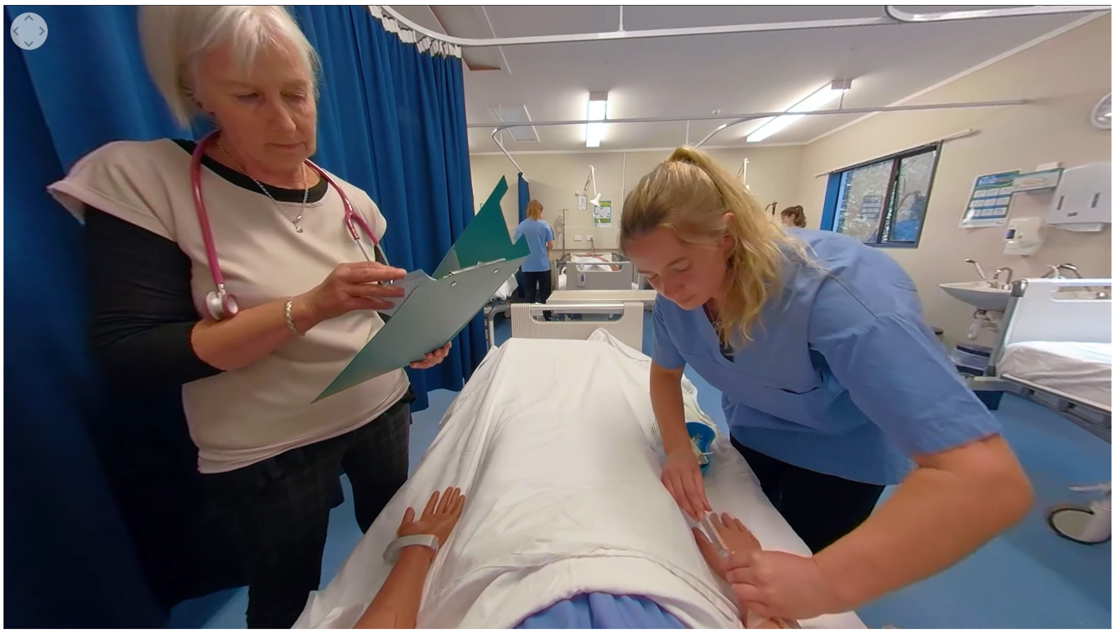

Figure 3.3.4: An illustration of the view

The image shows that there are other individuals and objects present in the video, in addition to the person conducting the examinations. This presents an opportunity to investigate the visual focus of the various participants and gain insights into their viewing patterns. Specifically, we want to understand whether salient objects, such as the person conducting the examination and interacts with the viewer through eye contact and talk, attract more visual attention compared to the other people in the video who are not engaged in direct interaction.

3.3.3 Define Area of Interest

To collect data about how the participants looked at the TV screen, an area of interest must be defined. The eye-tracker will be in different heights and angles depending on the participants, and therefore the coordinates of the gaze points on the screen will differ. Because of this, it is important to define an area of interest to be able to compare the results. Most eye-trackers allows the user to specify areas of interest prior to the start of an experiment. In Pupil Capture, this function is called "Surface Tracking" and allows the user to define planar surfaces within the environment to track areas of interest [9]. In this experiment, the area of interest is the TV screen where the video is shown.

To define a surface with Pupil Capture, AprilTag markers can be used. AprilTag is a visual fiducial system that can be used for a wide range of tasks across different discipline [68]. AprilTags are similar to regular QR codes and are designed to encode small data loads (between 4 and 12 bits), allowing for robust detection from longer distances [68]. Figure 3.3.5 shows the four AprilTags used in the experiment.

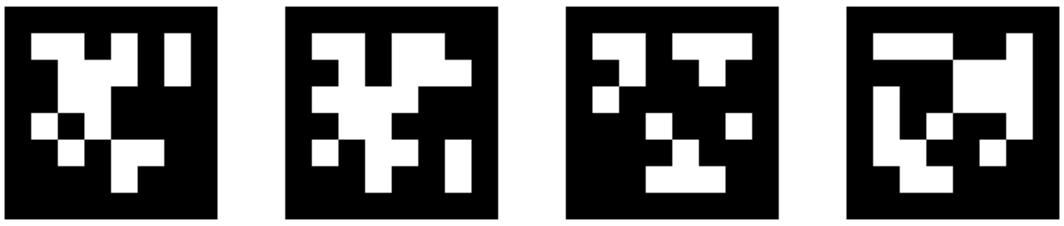

Figure 3.3.5: Four AprilTag markers [69].

A surface can be based on one or more markers. The marker should be placed near each other, and within the desired area of interest [69]. If the area of interest is a TV screen, as in this experiment, the markers should be placed in all four corners of the screen. This is done in PsychPy, so that they will be visible in the corners of the TV screen while the video is showing. To be able to define a surface in Pupil Capture, the "Surface Tracker" plugin must be turned on. Figure 3.3.6 shows how the AprilTags are placed on the TV.

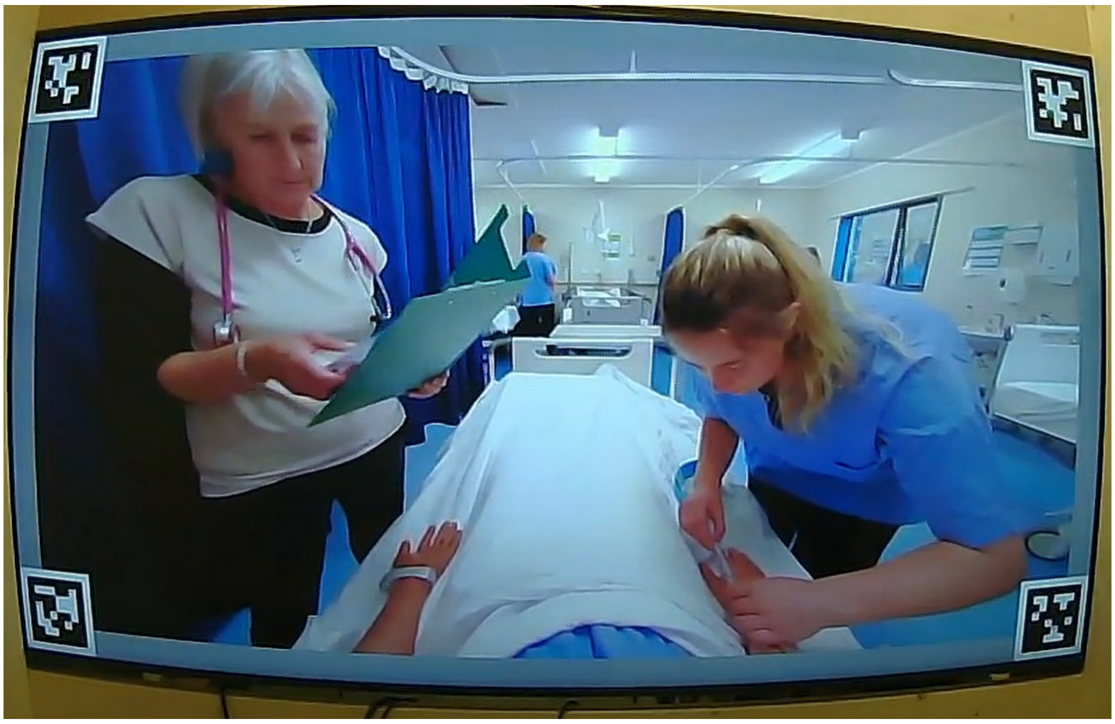

Figure 3.3.6: TV Screen with AprilTags

3.4 Conducting Eye-Tracking Experiment

When conducting the experiment a checklist will be followed, to ensure that the same procedure is consistently followed for all participants and that no steps are forgotten. The following is the list of what to remember before the participant arrive for the experiment.

- Consent form and participant ID must be ready
- Eye-tracker must be ready and connected
- Close other programs on host monitor and turn on "Do Not Disturb"
- Start Pupil Capture
- Start PsychoPy
- Check monitors, TV screen and sitting arrangement
- Check dual screen working

When participants arrives, the following steps will be taken:

- Read the first part of the brief
- Give the consent form
- Control signature
- Read the second part of the brief
- Place eye-tracker on the subject

The first part of the brief includes information about the context of the experiment, and the duration. The participant is told to remove disturbing items like chewing gum and mobile phone. The participant is also requested to provide permission for the use of the collected data. This is achieved by presenting a consent form, and get signatures form all participants to confirm their consent. By signing this consent form, participants agree to allow the data collected during the experiment to be used for analysis purposes, on the condition that the data remains anonymous. The experiment only collects data from the eye-tracker, ensuring that no personal information is obtained. Consequently, the data remains anonymous and cannot be traced back to individual participants.

During the second part of the brief, participants are informed about the placement and purpose of the eye-tracker, and instructed to remain as calm as possible while maintaining their natural head and eye movements during the experiment.

Before starting the experiment in PsychoPy, is it important to check the eye camera resolution and ensure the eye-tracker is placed correctly on the participant. Some parameters are resolution dependent, so this step can ensure good data quality.

The following is a list of tasks that will be performed during the experiment:

- Perform Calibration
- Perform Verification
- Start the experiment in PsychoPy

The calibration is done by following the Pupil Core manufactures "Best Practices" [15], taking into account the distance of the stimuli from the participant. Since it is a fixed distance, the calibration points should only be at this distance [15]. The calibration points should cover the outermost bounds of the gaze of the participant [15]. When the experiment is conducted on a screen, the calibration points should cover the surface of the screen. In this experiment this is done by placing points in the corners and in the middle of the TV screen. Figure 3.4.1 shows a visualisation of how the calibration points at displayed, one at a time.

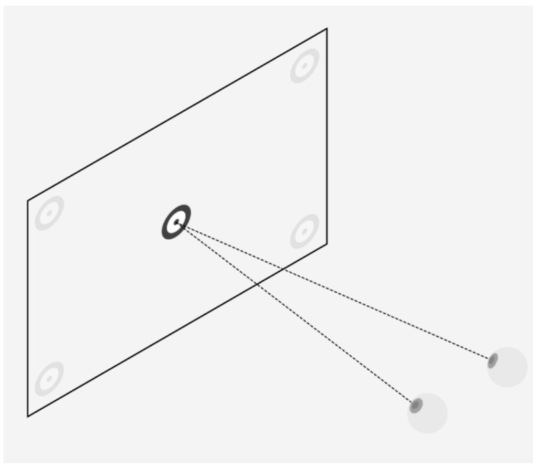

Figure 3.4.1: Visualisation of the calibration [69]

When the calibration is complete, the windows in Pupil Core that show the eyes should show a well-fit model as a blue circle surrounding the modelled eyeball [69], like in Figure 3.4.2. If the circle is dark blue, the model is within physiological bounds [69]. However, if the circle is light blue, the person in charge of the experiment should check the placement of the eye-tracker on the participant and re-calibrate. Calibration can be performed multiple times. The red circle and dot mark in Figure 3.4.2 mark the bounds and center of the pupil, respectively, and are used to track changes in the pupil diameter.

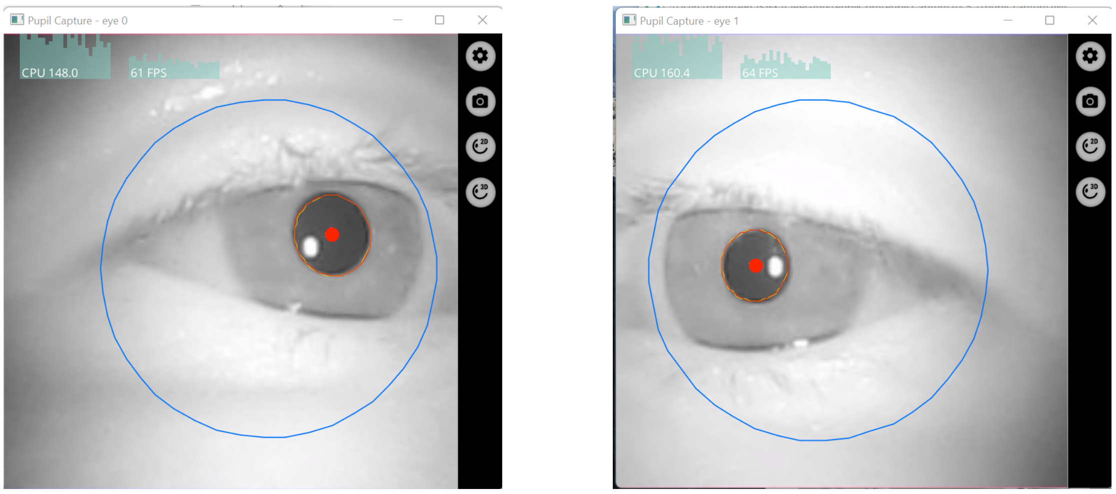

Figure 3.4.2: Eyes after calibration

After calibration, it is essential to conduct a validation check to verify its accuracy.

If the validation results in a high accuracy error, another calibration is required. It is therefore important that the validation points are different than the ones used for calibration [69].

The final step to do during the experiment is to start the experiment in PsychoPy, after this the routine will go automatically. Pupil Capture will automatically start recording as soon as the video start showing on the screen. While the experiment is running the person in charge should monitor that everything goes according to plan, from the outside of the box.

Following the completion of the experiment, a debrief is read to the participant, and the eye-tracker is removed. During the debrief the participant is instructed not to discuss or disclose any information related to the study, as this could potentially impact the results for the test persons who have not yet participated. After the participant has departed, the following steps are carried out:

- Check for leftovers from last person
- Clean equipment
- Check data recordings in Pupil Player
- Set up the experiment for new test subject

3.5 Data Cleaning

After each participant it is important to go through the data to check if the quality is good. This is done by watching the data recorded from the eye-tracker in Pupil Player. The raw eye-tracking data collected from Pupil Core contains various types of noise, such as blinks. Before analysing the data, it is necessary to preprocess and clean it. First, the data is looked through to see if the quality is good enough to include in the analysis. It is important to check that the eye-tracker was able to track gaze points through the whole duration of the video. The data is also checked for any outliers and artifacts that could influence the result. After this, if the data is good enough to include in the study, blinks are detected and removed from the data, as they can cause inaccuracies in the gaze positions. Finally, the data is segmented into fixations, which represent periods of stable gaze, and saccades, which represent rapid eye movements between the fixation points. The resulting cleaned data can then be used for further analysis.

CHAPTER

FOUR

IMPLEMENTATION OF OBJECT DETECTION

For this study, the YOLOv8 convolutional neural network was selected as the object detection model. When using an algorithm created by someone else, it is rarely a straightforward process. There are customization and configuration parameters that needs to be applied, and adjusted. The upcoming sections aim to detail the tools and methods used to implement and customize the object detector to suit the specific needs of this study.

4.1 Python

When considering a programming language for a machine learning project, it is important to weigh various factors, including personal experience, community support, and compatibility with the chosen framework. In the case of this project, Python was selected as the programming language due to its common use in machine learning and the fact that YOLOv8, the chosen object detection model, is implemented in Python using the PyTorch framework. This choice is expected to simplify the problem-solving during implementation and adjustments in the code, as solutions for any issues are more likely to be available in Python on forums and other resources.

4.2 Google Colaboratory

Google Colaboratory, also known as Google Colab, is a cloud-based Jupiter Notebook environment developed by Google Research [70]. It provides an accessible and user-friendly way to write and execute Python code directly in the browser, with no more setup required. Google Colab is particularly well-suited for machine learning and data analysis [70]. One of the key features of Google Colab is that it proivdes access to computing resources, including GPUs, free of charge [70]. The GPU the user get access to is NVIDIA Tesla T4 graphics card, which is designed for use in high-performance computing and artificial intelligence applications [71]. It features 16GB and GDDR6 memory and 320 Turing Tensor Cores, providing high-speed computing power for deep learning, machine learning and other complex computing tasks [71]. It was because of the access to additional computational power, Google Colab was chosen for this study.

4.3 PyTorch

PyTorch is an open-source machine learning library developed by Facebook. It is a flexible and modular open-source deep learning framework that is designed for both research and production deployment [72]. It offers a Python package for high-level features such as tensor computation, which are similar to NumPy, with powerful GPU acceleration, as well as TorchScript for easy switching between eager mode and graph mode [72]. PyTorch also offers pre-trained models, including those for object detection, segmentation, and classification. It is used in various machine learning and deep learning applications, including the state-of-the-art object detection model YOLOv8 used in this study.

4.4 Building the Model

In order to perform a good object detection on the video recorded with the eye-tracker, YOLOv8 was chosen. This is the newest version of the YOLO-models, and the accuracy has been improved from previous versions. Additionally, the latest implementation of YOLOv8 introduces a range of new features, including a user-friendly command-line interface (CLI) and a comprehensive GitHub repository, which simplifies usage and integration [73]. When implementing the model necessary libaries like OpenCV, PyTorch and NumPy were imported, and the YOLOv8 weights and necessary files were downloaded from their source [74]. This is not the original code from the developers of YOLOv8 [43], but a modified version witch combine both object detection and person re-identification.

The model was initiated by running the "track.py" file downloaded, while referencing the downloaded YOLOv8 weights. The model was then optimized for this specific study by adjusting parameters such as image frame and size, and making minor code modifications to ensure optimal performance. Testing was done on different videos, and the parameters were adjusted as needed.

To be able to use the data generated by the model, it was not enough to plot the bounding boxes and class onto the video, as was the default output in the code. It was necessary to get a file containing information about bounding box coordinates, and its width and height, as well as the class label for each object detected. The code was improved to generate bounding box coordinates, and a function was implemented to write this information to a file. In addition to the coordinates and class label, the file also stored the frame number in which the object was detected, as well as the width and height of the corresponding bounding box. By implementing these adjustments, the generated data from object detection became more accessible for analysis purposes. Table 4.4.1 is an example of the file generated from the model, this is the first eight rows of one of the participants.

Object	Confidence	Height	Width	X-mid	Y-mid	frame_idx
1bed	0.72	270	487	646.5	476	0
2person	0.73	91	36	593	278.5	0
4person	0.75	44	36	842	245	0
5person	0.78	118	89	495.5	549	0
5person	0.74	83	85	789.5	540.5	1
1bed	0.72	267	489	645.5	477.5	1
2person	0.73	91	37	594.5	278.5	1
4person	0.75	44	36	842	245	1

Table 4.4.1: Data Generated from Object Detection Model

4.5 Use the Model

Once the model was adjusted, a systematic approach was taken to test the different models available in YOLOv8. YOLOv8 offers five different models, namely YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x, representing Nano, Small, Medium, Large, and Extra Large. These models vary in terms of speed and accuracy, with smaller models being faster but less accurate, while larger models provide higher accuracy at a slower pace. In order to determine the most suitable model, all the available models were tested, ultimately leading to the decision of using YOLOv8l for the object detection.

After selecting the object detection model, the next step was to choose an appropriate person re-identification model. To make this choice, a testing approach was again used to evaluate different models. After careful consideration, the "osnet_ain_x1_0_msmt17.pt" model seemed as the preferred option for person re-identification in this study. This model has shown better cross-doimain performance than similar models[74], which means that it is adapting and handling variations in image quality, lighting conditions, camera angles, and other factors that may differ across different environments or sources [75]. Given that the individuals in the video move around and appear from various angles, this characteristic proves to be advantageous in this context.

Following the selection of the object detection and person re-identification models, they were applied to the set of videos collected from the different participants in the experiment. The results produced a set of separate documents for each participant, containing valuable information, as shown in Table 4.4.1. The collected information played a crucial role in further analyses and in the development of the gaze point prediction model.

4.6 Validating the Model

When using a pre-trained model, the accuracy has already been tested and calculated by the creators. These models are trained on large datasets and are evaluated to ensure their effectiveness before they are made available for use. However, a pre-trained model's accuracy may not be the same for all specific use cases. The model's performance may vary depending on factors such as the domain, data distribution, or the specific task being addressed.

To validate the accuracy of the object detection model used on the videos generated from the eye-tracking experiment, a visual inspection was used. This involved verifying the correct placement of bounding boxes and ensuring that the assigned class labels were appropriate. The purpose of this validation was to ensure high-quality results, and to ensure consistency in object detection across the videos for all participants. This is important to establish the reliability of the model for further analysis.

During the validation process, certain differences and errors in the detection were identified. The model occasionally wrongly labeled objects, such as mistakenly detecting a surfboard where it was not present. Furthermore, inconsistencies were observed in object detection across the videos for different participants. As result of this findings, data cleaning was done.

4.7 Data Cleaning

In order to improve the quality and reliability of the dataset for further analysis, specific actions were taken during the process of data cleaning. First, objects that were known to be absent in the videos were removed from the dataset, as they would not contribute any meaningful information to the analysis. An example was objects like the surfboard mentioned earlier. This step made the dataset cleaner by removing unnecessary information.

Furthermore, objects that were not consistently detected across all videos were also removed. This ensured a more accurate and reliable comparison of results among different participants. For instance, the model detected a sink, which was consistently present throughout the video, but it was only detected in some specific frames and not for all participants. Since this inconsistency would not provide valuable information for the analysis, objects with such patterns were removed from the dataset.

The last step in the data cleaning process was to assign a single name to each unique object. Although person re-identification was used in the model, it did not consistently assign the same identifier to the objects throughout the video. Therefore, manual adjustments were made to ensure that each object had only one name. This also ensured that all participants' data used the same identifier for each object, which was crucial for the analysis to be able to compare the results.

CHAPTER

FIVE

GAZE POINT PREDICTION MODEL

In order to predict future gaze points, a model was designed using a Sequential Long Short-Term Memory (LSTM) architecture. The objective of the model was to learn form historical gaze coordinates and characteristics of objects within the frame, in order to predict the next gaze point. This model serves at a starting point in predicting the gaze location for the manikin, which ideally in the future will be based on just the objects in the room.

5.1 Feature Extraction

The dataset used in this model was obtained from the combination of the eye-tracking experiment and YOLOv8 object detection model outputs. Data preparation was performed on the raw data generated from these two sources, to extract and structure the features used in the building of the model. Feature extraction is a critical step in preparing the dataset for the predictive model. It is during this process key attributes are identified, that can help improve the model's ability to learn from the data.

In this dataset, each instance corresponds to a gaze point in the frame, and include the following features: normalized gaze coordinates, attributes of the objects within the frame, such as width, height, and center point coordinates, as well as the identity of the objects present in the frames. The gaze coordinates represent where someone was looking in a standard two-dimensional space, like a screen or an image. Meanwhile, the characteristics of the objects in each frame provide additional details about where the object is placed at that specific moment, as well as the size of the object.

Historical gaze coordinates were also extracted from the data. They were arranged in a lagged order, which means that for each gaze coordinate in the data set, the model also considers the gaze coordinates from one or more previous time steps. In this dataset, the gaze coordinated from previous 1,2, and 3 time units. For prediction tasks, it is often the case that future states are influenced by past and current states, and this allows the model to learn temporal patterns in gaze movements.

The process of preparing the dataset also involved the encoding of the categorical variable, "Object in Frame". This variable is an identifier for the objects present within the frames. Neural networks models, being mathematical models, perform calculations on numerical input data. Therefore, they are not able to understand categorical data directly. Because of this, categorical data must be encoded into a numerical format that can be used by the model. As such, the "Object In Frame" variable was transformed from a categorical variable to a numeric format. This transformation was performed using the LabelEncoder utility from sklearn, which converts each unique string value into a unique integer [76].

During the final stages of data preparation, two operations were carried out to make sure the data is accurate and fits the chosen machine learning model. Firstly, any data entries containing missing values were eliminated. After this, the data was transformed into a three-dimensional array. This transformation was necessary to meet the input requirement for the LSTM model, resulting in a data shape of (samples, timesteps, features).

5.2 Model Architecture and Training

The chosen model for this study is a Sequential Long Short-Term Memory (LSTM) network. LSTM was chosen for its ability to learn from sequences of data, and also remember patterns over time, making it suitable for the task of predicting gaze points basted on past information. The code for the model predicting gaze point is located in Appendix B.

The network is structured with two LSTM layers, each followed by a dropout layer. The first LSTM layer contains 128 cells and returns sequences, passing the outputs along to the next layer. The purpose of this layer is to process the input sequence and pass on valuable temporal information. The dropout layer, with a rate of 0.2, is used to prevent overfitting during training by randomly setting input units to 0 at each update during training time [58]. The specific rate of

0.2 was determined through testing to identify the value that gave the best results.

The second LSTM layer contains 64 cells and does not return sequences. This means that it processes the features extracted by the previous layer and provides the final output of the sequence to the next layer in the model. Another dropout layer with a rate of 0.2 follows this LSTM layer, further helping to prevent overfitting.

The last layer of the network is a dense layer with two units, which correspond to the next predicted x and y gaze positions. The use of a dense layer here enables the model to output continuous values, fitting the nature of the gaze prediction task [77].

The model is also compiled with the Adam optimizer and Mean Squared Error as the loss function. Adam was chosen for its efficient computation and the ability to handle sparse gradients [56], while MSE was used due to its effectiveness for regression tasks, which fits the nature of this gaze prediction problem [53].

For the training process, the model was set to be trained for 50 epochs with a batch size of 32. To avoid overfitting and unnecessary training time, an EarlyStopping callback was used, which stops the training when the model's performance on the validation set stops improving for a specified number of epochs, in this case, 10 epochs. This approach helps to ensure that the model generalizes well, prevent overfitting, and performs effectively on unseen data [78].

A crucial part of the model training process is splitting the data into separate sets for training and testing. In this study, the data was divided based on an 80/20 percentage split. This means that the first 80% of the data, in chronological order, was used for training the LSTM model. The remaining 20% of the data was then used as a testing set. Since this is data sequential data, it is important that the training and testing sets are not divided in a random order. This approach ensures that the model is evaluated on unseen data, providing a more reliable measure of its predictive performance and ability to generalize.

5.3 Model Evaluation

The evaluation of the LSTM model performance was primarily measured on its ability to accurately predict gaze points. To measure this accuracy, a suite of error metrics were used.

The first evaluation metric used was Mean Squared Error (MSE), which provided a measure of the average squared difference between the predicted and actual gaze points. This metric treats larger errors more harshly due to its squared characteristic.

Secondly, Mean Absolute Error (MAE) was calculated. This metric provides the average absolute difference between predicted and actual gaze points, offering a linear measure of the error magnitude. Root Mean Error (RMSE), a derivative of MSE, was also used. This metric provides the square root of average of squared differences between prediction and actual observation.

For the last evaluation metric, the coefficient of determination, known as R^2 score was implemented. The R^2 score provides a measure of how well future samples are likely to be predicted by the model. Best possible score is 1.0, and negative scores indicating a model performing worse than a constant baseline.

The choice to include four different evaluation metrics – Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the R^2 score – is because of their ability to provide different perspectives on the model's performance. By implementing all these metrics, it is possible to gain a more detailed overview of the model's performance, taking into account the impact of the errors, the constancy of errors, and the model's ability to predict gaze points.

The evaluation process also included visually evaluation of the model's performance. To evaluate the model visually, the predicted gaze points from the model were plotted directly on the video. These predicted points were compared with the actual points where the participant was looking. This process included converting the gaze points from their normalized form back to their original position within each video frame. The model's predicted gaze points were marked in green, while the actual gaze points were marked in red. By overlaying these points on the video frames, it was possible to visually compare the model's predictions against the actual gaze points and see where and when the model accurately predicted the gaze. This evaluation method was included in the study to provide a deeper understanding of the models' prediction accuracy, beyond the abstract numerical values.

CHAPTER SIX

RESULTS

6.1 Eye Tracking Experiment

In the eye-tracking experiment, a total of 14 participates contributed. However, it was found that the data collected from two of the participants were very noisy, making it difficult to extract meaningful insights from their gaze patterns. It was discovered that the primary reason for the noisy data on these participants were the use of dark mascara on the eyelashes. The dark mascara led to confusion for the eye tracking equipment, which struggled to accurately track the pupils. Although the calibration and validation process appeared to be successful initially, the issue with the noisy data became clear while the actual experiment was taking place.

Furthermore, two other participants also had noisy data. Upon reviewing the recordings, it was observed that these participants squinted, leading to a temporary loss of pupil tracking by the eye-tracker. As a result, the eye-tracker was unable to accurately track the gaze point during that time period.

As a result, the decision was made to exclude these four data sets from further analysis to ensure the integrity of the findings. Consequently, the final dataset used for the analysis included data from the remaining 10 participants. This decision allowed for a more robust and reliable comparison of the gaze patterns for the different participants and the relationship between the participants' focus and the objects detected. By focusing on the data from the 10 valid sets of data, the study aimed to obtain accurate and relevant insights into human gaze behavior.

6.2 Object Detection On Eye-Tracking Videos

This section presents the results from the process of applying object detection on the videos from the eye-tracking experiments. The videos capture the field of view for each participant. In order to identify and analyze the objects in the recorded videos, a pre-trained YOLOv8 model, with Person Re-Identification model OSNet, were used. Although the video shown in the experiment were identical for each participant, the outcome of object detection varied. The overview of the number of bounding boxes and the number of unique objects detected for each participant are shown in Table 6.2.1.

Participant	Bounding boxes detected	Unique objects detected
P1	8058	124
P2	8258	140
P3	8347	120
P4	9057	132
P5	7123	147
P6	9731	129
P7	7882	130
P8	8918	161
P9	9017	147
P10	8551	135

Table 6.2.1: Bounding boxes and unique objects detected for each participant

The object detection model generated a large number of bounding boxes for every video, assigning a new box to each object in each frame. Throughout the object detection process, the videos were divided into approximately 1900-2200 frames (images), with the specific number varying for each video. This resulted in a significant amount of data that needed to be processed and cleaned.

Even though a person re-identification model was used, many objects were assigned multiple identifiers, making it challenging to accurately track each unique object throughout the video. The first step in addressing this issue was to identify all the different labels assigned to each object and manually change them so that each object would have a single identifier. By changing the object identifiers, it became simpler to accurately follow and examine the objects in the video. This made the foundation for further analysis more reliable.

Despite using the same video in the experiment for each participant, the object detection results were not consistent across the recordings. As we see in Table 6.2.1, the number of detected bounding boxes and the number of detected unique objects are different for all participants. Some objects were consistently detected in every video, while others were occasionally missed or misidentified. Furthermore, there were instances of false detections, such as a surfboard being detected when none was present in the video. These inconsistencies required a thorough review and manual correction to ensure the accuracy of the object detection results. This involved removing any false detections and ensuring that each object was correctly identified and represented in the data. The cleaned data was then used for further analysis, enabling the analysis of participants' gaze patterns and the relationship between their attention and the objects detected in the videos. This was also used as input data for the gaze point prediction model.

6.2.1 Object Detection Data after Data Cleaning

In Table 6.2.2 the results for each participant after the data was cleaned are shown, where each participant's data consists of seven different objects. These objects are identified as 1person, 2person, 3person, 4person, 5person, 6person and 1bed. All of them are shown in Figure 6.2.1, 6.2.2, 6.2.3 and 6.2.4.

Participant	Bounding boxes	Unique objects
P1	6473	7
P2	7795	7
P3	6718	7
P4	7418	7
P5	5710	7
P6	9059	7
P7	5970	7
P8	6945	7
P9	7430	7
P10	6735	7

Table 6.2.2: Bounding Boxes and Unique Objects Detected after data clean

The number of bounding boxes after data cleaning, also varies across participants. One possible reason for this variation is the difference in the number of frames captured in each video. Even though all videos have the same duration, the number of frames can still differ. As bounding boxes are placed on each frame, this difference in the number of frames directly affects the number of bounding boxes. However, given that all participants are viewing the same video, the number of bounding boxes should hopefully not influence the outcome of the analysis. This expectation relies on the condition that the objects are detected for consistent duration across the participants.

Another reason why the number of bounding boxes is assumed not to impact the analysis is because the focus is on the objects themselves rather than the quantity of the boxes. The bounding boxes provide information about the objects' positions within each frame, allowing for further analysis and understanding of their movements and interactions. As long as there are a sufficient number of bounding boxes to provide this necessary information, the analysis can still be carried out effectively regardless of the variation in the number of bounding boxes.

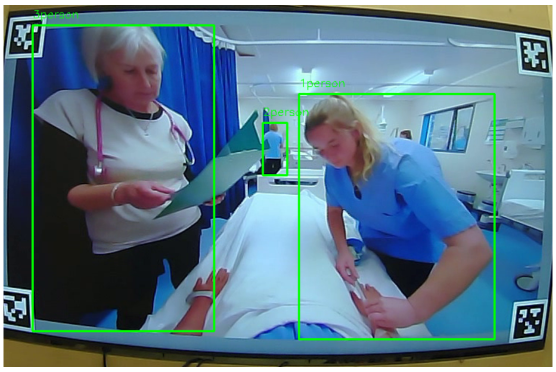

Figure 6.2.1: 3/2/1person

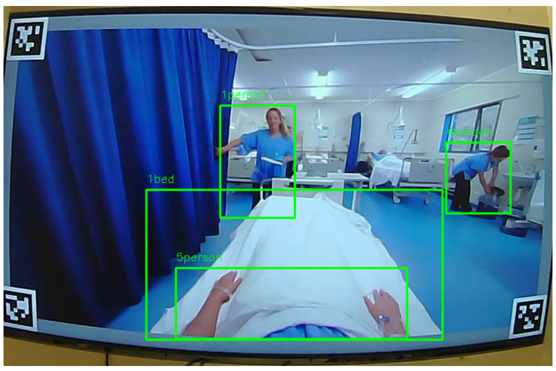

Figure 6.2.2: 1bed, 1/5/4person

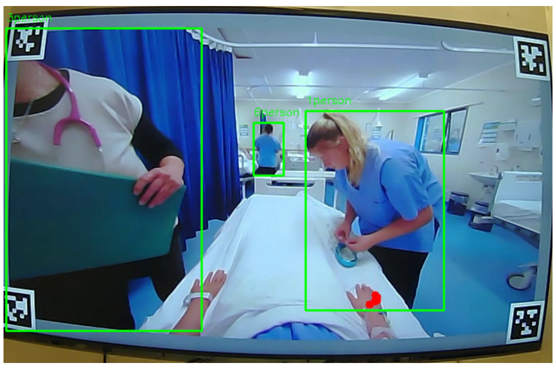

Figure 6.2.3: 3/6/1person

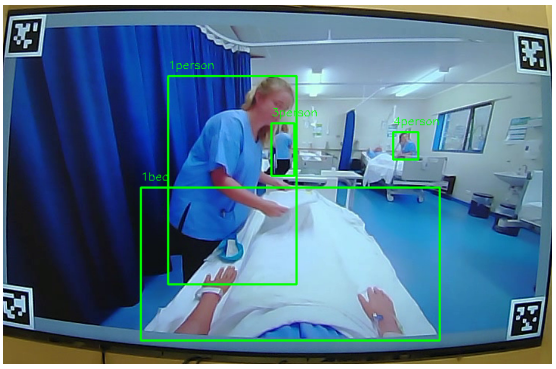

Figure 6.2.4: 1bed, 1/3/4person

6.3 Combining Eye-tracking and Object detection

In this section, the results obtained from combining the eye-tracking and object detection data are presented. The goal was to identify what the participants were looking at in the video, and to explore any patterns in their gaze behavior, offering valuable insights into the visual attention of the participants. By merging these two sets of data, the aim was also to extract features that could be used in the gaze point prediction model.

By mapping gaze points onto the bounding boxes of the detected objects, it was possible to identify which objects the participants were focusing on during the video. This involved finding whether each gaze point was located within the boundaries of a specific bounding box. If the participants were not looking at any of the bounding boxes, the gaze points where categorized as looking at the background.

The analysis of the combined data revealed trends in participants' visual attention. Certain objects consistently captured participants' attention more than others, indicating their significance in the scene. Notably, the salient objects, specifically the person interacting with the camera with eye contact and conversation, attracted the most focus. Furthermore, common gaze patterns emerged among the participants, revealing shared visual behavior. These findings contribute to a deeper understanding of visual behavior. By integrating eye-tracking and object detection data, valuable insights are gained into the objects that attract attention and how attention shifts among different objects.

Furthermore, the analysis of dwell time, discussed in detail below, provides additional insights into participants' visual attention. This analysis quantifies the duration participants spent fixating on each object, further supporting and extending the findings obtained from the integration of eye-tracking and object detection data.

6.3.1 Dwell time

In this study, the dwell time for all the objects identified by the object detection model was analysed and visualized by using bar charts. The charts for all the participants are shown in Figure 6.3.1, 6.3.2, 6.3.3, 6.3.4, 6.3.5, 6.3.6, 6.3.7, 6.3.8, 6.3.9 and 6.3.10. The bar charts are also illustrated in a larger size in Appendix A. The data is also presented in Table 6.3.1, which shows the dwell times (in seconds) of 10 participants (P1 to P10) for each object. Dwell time provides information about how much time participants spent looking at a given area of interest. The dwell time for each object varied among the participants, indicating differences in their focus and engagement with the objects in the video.

To calculate the dwell times, the duration of each fixation on an object was measured before the gaze shifted to another location. Dwell time is a valuable metric for understanding participants' visual attention as it captures the temporal aspect of their engagement with specific objects.

Object	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
1bed	5.0	6.0	8.9	4.8	5.8	13.1	5.3	5.7	3.7	14.6
1person	39.5	40.0	56.1	33.1	51.8	50.3	48.4	38.6	40.5	58.8
2person	4.4	1.8	3.2	0.7	3.6	1.5	0.3	2.3	2.2	0.1
3person	7.9	6.7	0.2	7.5	4.8	6.7	3.8	7.7	3.0	4.3
4person	0.8	0.2	2.0	0.0	2.0	1.5	0.0	0.2	0.7	1.0
5person	0.5	0.5	0.5	0.8	0.3	0.0	0.9	0.3	0.8	0.3
6person	1.5	0.4	0.0	0.2	0.0	0.0	0.2	0.0	0.6	0.2
Background	11.1	8.1	6.0	7.5	3.6	4.9	9.5	18.7	14.9	2.4

Table 6.3.1: Dwell Time for each Object and Participant

The analysis of dwell times gave interesting findings and patterns. For example, the dwell time on the 1person object was notably higher than for the other objects, with participants spending an average of 40.82 seconds fixating on it. This finding suggests that the 1person object was particularly engaging and captured the attention of viewers. The bar charts, in Figure 6.3.1-6.3.10, also show that all participants mainly focused on the 1person object during their dwell time. On the other hand, the 6person object had the least dwell time, with an average of just 0.31 seconds. This implies that it was less interesting or relevant to the viewers. It's important to note that the 6person object was only visible in the video for a few seconds, which explains the minimal dwell time. In contrast, the 1person object appeared throughout the entire duration of the video, which could contributing to the long dwell time.

These findings provide insights that align with the goals of the study, which aim to understand how people's visual attention responds to different objects, identifying what is of interest in an environment with multiple objects. For instance, the longer dwell time on the 1person object suggests that it captured participants' attention more effectively, indicating that it had a stronger visual impact. On the other hand, the shorter dwell time on the 6person object suggests that it received less attention, indicating its lower visual impact. These findings support the assumption that salient objects tend to attract more visual attention, while less salient objects receive comparatively less attention.

The dwell times for other objects, such as 2person, 3person, and 5person, showed moderate engagement, with average values ranging from 2.12 to 3.56 seconds. These objects also showed notable variations in dwell time among the participants, suggesting that individual differences can play a role in the viewers' engagement with the different objects.

Additionally, the Background object had an average dwell time of 6.35 seconds, which could indicate that viewers occasionally shifted their attention from the primary focus to explore the background elements. This shift in focus might be because of a temporary loss of concentration, or an attempt to search for more context and interesting features within the background.

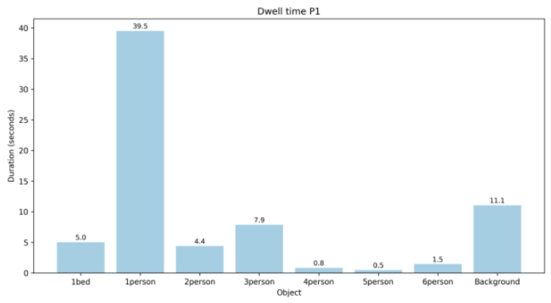

Figure 6.3.1: Participant 1

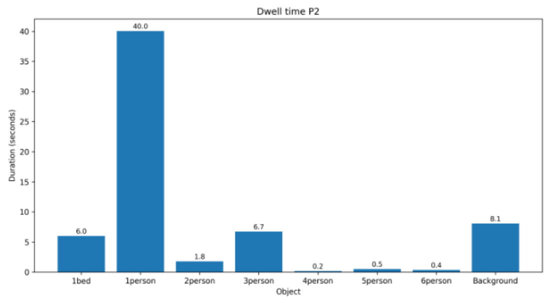

Figure 6.3.2: Participant 2

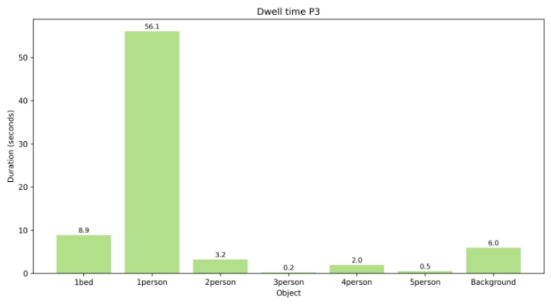

Figure 6.3.3: Participant 3

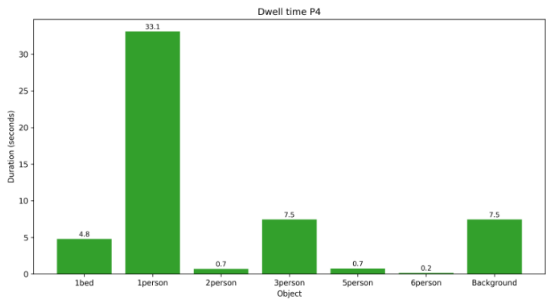

Figure 6.3.4: Participant 4

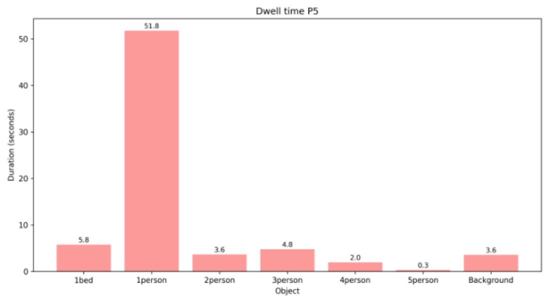

Figure 6.3.5: Participant 5

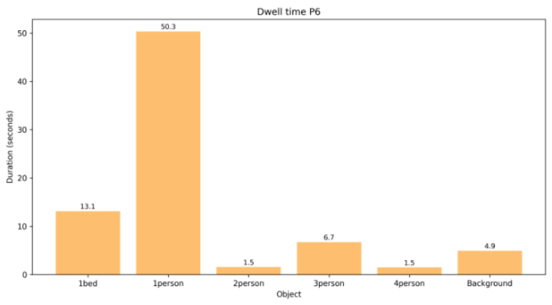

Figure 6.3.6: Participant 6

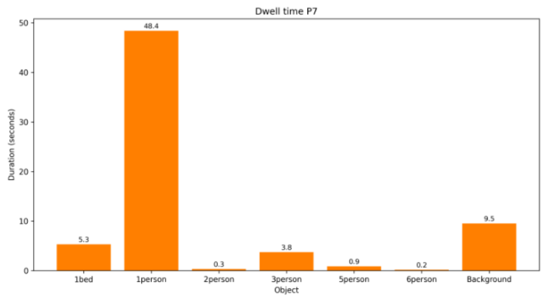

Figure 6.3.7: Participant 7

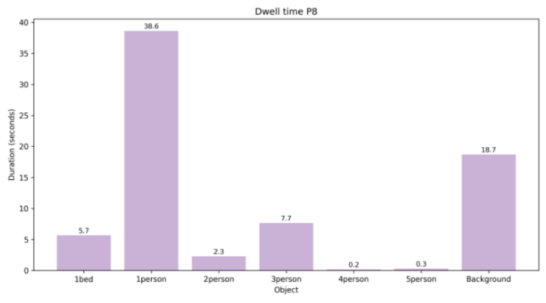

Figure 6.3.8: Participant 8

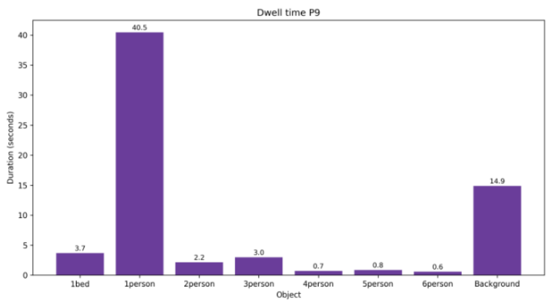

Figure 6.3.9: Participant 9

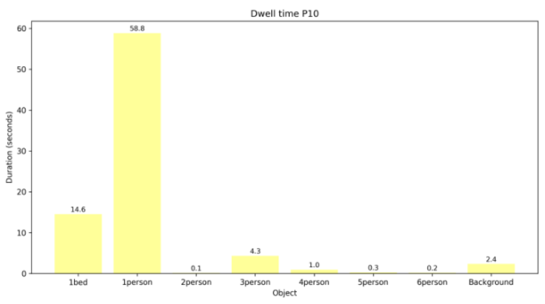

Figure 6.3.10: Participant 10

6.4 Exploring the Gaze Point Prediction Model

The initial strategy for the model predicting gaze points was to merge the data from all ten participants into a single large dataset, and use this as input. However, this proved to be a more complicated task than initially expected. The assumption was that merging datasets based on frame index would be possible, given that all videos had the same starting and stopping points, thereby likely resulting in same number of frames. This assumption did not turn out to be true.

The inconsistency in the number of frames can be caused by several factors. A participant's visual attention could impact the frame count significantly. For instance, blinking or looking away from the screen could impact the total number of frames, since the video is recorded with an eye-tracker. Further, the sampling rate may vary among participants, even when the software settings are consistent across the experiment for all participants. Uncontrollable factors such as blinking might influence the sampling rate, indicating that the software's control over the situation is not complete.

Since the frame number strategy was not successful, an attempt was made to use the "world_timestamp" feature to compare the participants. Given the identical start and end points for each video, the expectation was that the timestamps would match. However, timestamps are logged with millisecond precision for each gaze point. As a result, since the gaze points vary for all participants, the timestamp values ended up being inconsistent.

Despite the lack of a method to synchronize participant data to a specific point in the video, an effort was still made to merge the datasets. The assumption was that any inherent patterns in the gaze data might still be identified by the model. It is important to note that machine learning models often do well in finding patterns and making predictions, even in seemingly complex or noisy data. Therefore, despite the synchronization challenges, it's possible that meaningful analysis can still be obtained from the combined dataset. The model was tested with various configurations, combining data from all ten participants, from five participants, and even running the model with data from a single participant. The results obtained from these evaluations are presented below.

6.4.1 Evaluation Metrics - Ten participants

The model predicting gaze point was evaluated using combined data from all ten participants. The code took approximately 17 hours to complete, underscoring the complexity of the task and the computational resources required when handling larger datasets. The performance metrics for this larger dataset are shown in Table 6.4.1.

Metric	**Value**
Mean Squared Error	0.0041
Mean Absolute Error	0.0565
Root Mean Squared Error	0.0639
R^2 Score	-0.2429

Table 6.4.1: Performance Metrics for Ten Participants

In this study, the model's performance was evaluated using several metrics, including Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the R^2 Score.

The computed MSE was 0.0041, indicating a relatively small difference between the predicted and actual gaze points. This is a promising result as it suggests that the model's predictions are fairly accurate. The MAE value of 0.0565 indicates the absolute difference between the predicted and actual gaze points, also showing the model's relative accuracy.

The RMSE value, calculated as 0.0639, measures the differences between values predicted by the model and the values observed. This value being relatively low also demonstrates the accuracy of the model's predictions.

The R^2 score for the model, however, was -0.2429. This negative value indicates that the model's predictions do not fit the actual data, for this set of ten participants. This result indicate challenges in model generalization when dealing with a larger and potentially more diverse dataset. Therefore, even though the model appears to accurately predict gaze points when considering error metrics, the negative R^2 score shows that the model might not be suitable for reliably predicting gaze points in this case.

This outcome could be due to the complexity when dealing with data from a larger number of participants. Variations in how each participant views the video can create inconsistencies in the data, making it more difficult for the model to find common patterns and accurately predict gaze points. An other reason could be the difficulty of synchronizing the data between different participants. Each frame in a participant's data does not necessarily correspond to the same point in the video for another participant. Consequently, finding consistent patterns across different participants becomes challenging.

6.4.2 Evaluation Metrics - Five participants

The model predicting gaze point was additionally tested using combined data from five participants. The process took approximately four hours to complete, demonstrating the computational requirements, although significant less time-consuming than when dealing with the full set of ten participants. The performance metrics for this smaller dataset are shown in Table 6.4.2.

Metric	**Value**
Mean Squared Error	0.0038
Mean Absolute Error	0.0423
Root Mean Squared Error	0.0617
R^2 Score	0.6483

Table 6.4.2: Performance Metrics for Five Participants

The performance of the model was evaluated using the same metrics as before, including Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the R^2 Score.

The computed MSE was 0.0038, indicating a slightly smaller difference between the predicted and actual gaze points when compared to the ten-participant dataset. This suggests an improvement in the model's predictive accuracy. The MAE of 0.0423 further supports this conclusion, as it also shows a decrease in the absolute difference between the predicted and actual gaze points.

The RMSE was calculated to be 0.0617, which is a lower value than obtained when using the larger dataset. This result is promising as it indicates the model is better able to accurately predict gaze points for this subset of five participants.

The R^2 score for the model was 0.6483, a significant improvement from the negative value observed in the ten-participant dataset. This positive R^2 score indicates a good fit between the model's predictions and the actual data for this subset of participants. The results indicate a better model generalization capability when working with a smaller and potentially less diverse dataset.

This improved outcome could be caused by the lower complexity associated with handling a smaller number of participants. With fewer individuals, the model may find it easier to identify common gaze patterns and accurately predict gaze points. The decreased difficulty in synchronizing data across fewer participants may also play a role, as corresponding frames between different participants are likely to match more closely for five participants, then for 10 participants.

6.4.3 Evaluation Metrics - One participant

The model predicting gaze point was further tested using data from a single participant. The process took approximately one and a half hour to complete, demonstrating the model's efficiency when handling smaller datasets. The performance metrics for this individual dataset are displayed in Table 6.4.3.

Metric	**Value**
Mean Squared Error	0.0019
Mean Absolute Error	0.0332
Root Mean Squared Error	0.0431
R^2 Score	0.8666

Table 6.4.3: Performance Metrics for a Single Participant

As in the previous cases, the model's performance was evaluated using Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the R^2 Score.

The computed MSE was 0.0019, marking a significant decrease from the five- and ten-participant datasets. This result suggests a significant improvement in the accuracy of the model's predictions. The MAE value of 0.0332 further supports this finding, indicating a smaller absolute difference between the predicted and actual gaze points.

The RMSE was determined to be 0.0431, the lowest value among the three datasets. This suggests that the model was most accurate in predicting gaze points when using data from a single participant.

The R^2 score for the model was 0.8666, a notable increase from the five-participant dataset. This indicates a strong fit between the model's predictions and the actual data when dealing with an individual participant. The result confirms a significantly better model generalization capability when working with a less diverse dataset.

These improved metrics can be because of the lower complexity of the data obtained from a single participant. The model can more easily identify patterns and make accurate predictions when there are fewer variations in the data, like individual differences in visual attention. Furthermore, the absence of synchronization issues, that are relevant when combining datasets from multiple participants, are assumed to improve the model's performance.

6.4.4 Visual Evaluation - One participant

In addition to the traditional evaluation metrics, a visual method has been used to demonstrate the performance of the model predicting gaze point. This visualization approach was most effectively applied when using the dataset from a single participant. When dealing with data from multiple participants, visualizing the results became more challenging due to complexities associated with frame numbers and variations in gaze points among participants. Thus, the visualization was only conducted on the single-participant dataset.

This visualization provided a detailed insight into the performance of the gaze prediction model. A strong visual correlation between the green (predicted) and red (actual) gaze points would provide visual confirmation of the model's effectiveness. The advantage of such an approach is that it offers an intuitive understanding of the model's prediction accuracy, beyond the abstract numbers presented by evaluation metrics.

In Figures 6.4.1, 6.4.2, 6.4.3 and 6.4.4, a set of snapshots from the video is presented to show the relationship between the red and green dots. These dots represent the actual and predicted gaze points, respectively. As confirmed in the snapshots from the video, the green dot, despite not always landing directly on the exact point of the actual gaze (the red dot), consistently falls within the same general area. Notably, it also manages to replicate movements like saccades and fixations with a high degree of accuracy and timing. This pattern confirms that the gaze prediction model effectively tracks the path of the actual gaze throughout the video.

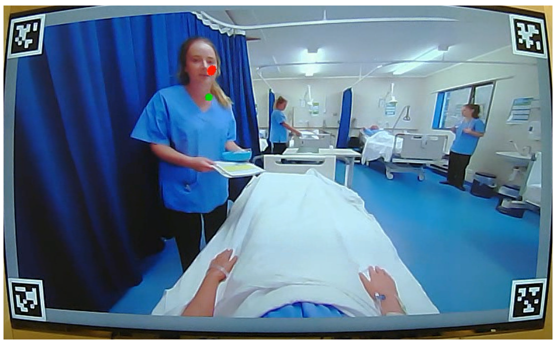
Figure 6.4.1: Gaze Point Example 1

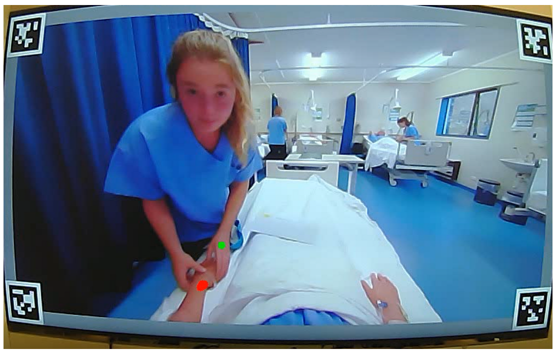
Figure 6.4.2: Gaze Point Example 2

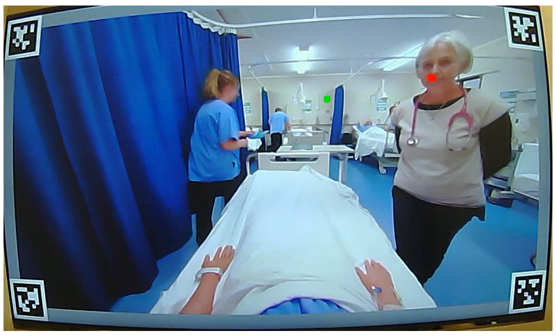
Figure 6.4.3: Gaze Point Example 3

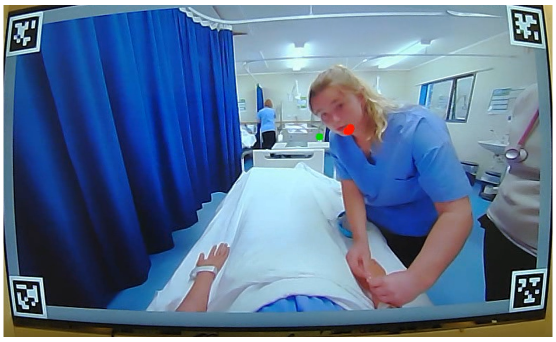
Figure 6.4.4: Gaze Point Example 4

The snapshots show in Figure 6.4.1 - 6.4.4 are from the video, and data, from the participant used in training of the model. To further test the model's performance, snapshots from another participant's data are presented in Figures 6.4.5, 6.4.6, 6.4.7, and 6.4.8. A visual evaluation indicates that the model performs effectively. Attempts were made to capture snapshots from similar situations. Notably, the model trained on one participant's data can be applied to data from other participants

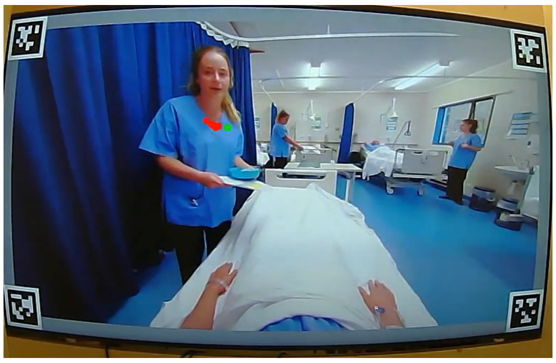

Figure 6.4.5: Gaze Point Example 1

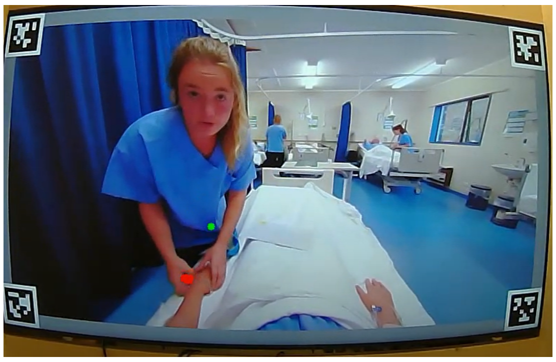

Figure 6.4.6: Gaze Point Example 2

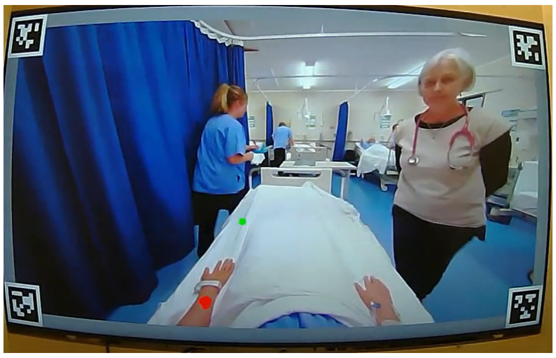

Figure 6.4.7: Gaze Point Example 3

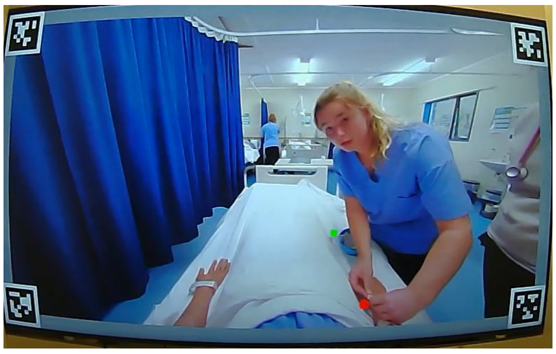

Figure 6.4.8: Gaze Point Example 4

CHAPTER

SEVEN

DISCUSSION

7.1 Understanding the Visual Attention

The analysis of the combined data from eye-tracking and object detection data presented in the results section provides valuable insights into participants' visual attention and gaze behavior. The integration of these two datasets aimed to extract features to use in the gaze point prediction model, and also provide valuable insights into participants' visual attention. The findings offers valuable information for implementation of eye movement in a full-body patient simulator, specifically a manikin that replicates human anatomy and physiology.

The analysis of dwell times, a measure of the duration participants spent fixating on each object, revealed interesting patterns and variations among the objects. The significantly higher dwell time on the object named 1person indicates that it had a stronger visual impact and effectively captured participants' attention. This finding supports the assumption that salient objects tend to attract more visual attention. On the other hand, the lower dwell time on the object named 6person implies that it had less visual significance and received significantly less attention. This means that the gaze point prediction model should be able to understand that salient objects should attract more visual attention. This is something to consider in the implementation of eye movement in the manikin's eyes.

The variation in dwell times among the participants also highlights the role of individual differences in interest in the different objects. Objects such as 2person, 3person, and 5person got moderate engagement, with notable variations in dwell time. This indicates that participants had varying levels of interest in these objects, further highlighting the importance of considering individual differences in visual attention. For a simulator to be truly realistic, it may need to be capable to adjust for these variations in its eye movement patterns.

Additionally, the average dwell time on the Background object indicates that participants occasionally changed their attention from the primary focus to explore the background elements. This shift in focus could be because of factors such as a temporary loss of concentration or the search for additional context or interesting features within the background.

These findings underscore the potential of integrating eye-tracking and object detection data for understanding and comparing visual attention. By understanding these insights, it becomes possible to enhance the realism of a manikin-based patient simulator by incorporating eye movement behavior. The combined analysis of eye-tracking and object detection data provides crucial groundwork for understanding participants' visual attention and establish the basis for integration of eye movement tracking in a manikin-based patient simulator.

Additionally, the combination of eye-tracking data and object detection data, in addition to valuable insights into the patterns of visual attention, also provided a solid foundation for the dataset used in the building of the gaze point prediction model. The gaze point prediction model developed in this study serves as the starting point for a future implementation of eye movement in such a manikin, facilitating more accurate and realistic patient simulations.

7.2 Evaluation of Prediction Model Performance

A clear trend observed across all three sets of results is that the accuracy of the model predicting gaze point improved significantly as the number of participants in the dataset decreased. The MSE, MAE, and RMSE metrics all shows an improvement as the datasets became less diverse, moving from ten to five, and then to one participant. This improvement suggests that the model was more capable of making accurate gaze point predictions when dealing with less diverse data.

One interesting finding was the negative R^2 score when dealing with the ten participant dataset. This result indicates that the model's predictions did not fit the actual data well when handling a larger and more diverse dataset. The R^2 score improved considerably when the number of participants was reduced, suggesting better model generalization capability when dealing with less diverse datasets.

This trend could be caused by several factors. First, data derived from a larger number of participants is naturally more complex, with potentially more variation in gaze patterns. Second, synchronizing data across multiple participants poses a significant challenge, as the correspondence between frames is not always precise across different participants.

Despite these challenges, it's important to note that, in practical applications, a model's ability to handle diverse data is essential. Our real-world environment consists of individuals with unique viewing patterns and habits, meaning a gaze prediction model should ideally be able to handle this variability effectively.

Therefore, while the model performed best with a single participant's data, this scenario might not reflect real-world conditions where data from multiple individuals need to be analyzed concurrently. Moreover, the model's computational efficiency also needs to be discussed. While it was more time-efficient to compute the gaze predictions for a single participant, scaling up to larger datasets significantly increased the computation time.

As the model's ability to handle diverse data is essential in practical applications, it is necessary to consider how this will impact its implementation in real-world simulators. A model that performs well with less diverse data may not fully capture this real-world diversity. This represents a key challenge in transferring the research findings into a practical simulator setting. In order to achieve realistic eye movements, it is believed that the movement should be based on patterns observed across multiple participants, rather than relying solely on data from a single participant.

In an attempt to improve the model's suitability for real-world scenarios, a novel approach was explored. The aim was to create a model that determines the next gaze point solely based on the objects present in the room. However, this proved to be a complex task given the available data. The dataset exhibited imbalances, which posed challenges for accurate gaze prediction. The model developed in this study represents an initial step in predicting gaze location for the manikin. Ideally, future work will refine the model to rely exclusively on the objects in the room, enhancing its applicability to real-world scenarios.

7.3 Limitations

Like all studies, this research has its limitations. The analysis relies on the data from the eye-tracking technology and object detection algorithms available during the research period, which may not cover all elements of visual attention. It is also crucial to acknowledge that the bounding boxes generated by the object detection model are not precise and cover a larger area than just the objects. This limitation in the study could potentially get incorrect results since participants may fixate on areas nearby an object, which could mistakenly be registered as fixation on the object itself.

The analysis, and the dataset used in the model predicting gaze point, is based on a limited dataset size of just 10 participants. The small size of this dataset may not provide an accurate representation of the diverse visual behaviors in a larger, more varied population. Although patterns are identified among these 10 participants, the group is too small to draw definitive conclusions from these findings.

The research is based on the visual attention on a video, raising questions about the relevance of these results for a dynamic, real-world scenario. Additionally, the model predicting gaze point in its current form operates on pre-collected data, and a non-real-time setting. This limitation limit the use in situations requiring real-time predictions, as for the manikin. Attempts were made to make the model's gaze point prediction only dependent on objects within the frame. However, this approach did not get good results with the data collected in this study. The dataset obtained in the study was significantly imbalanced, which also is an limitation. There were a higher number of registered gaze points on the object labeled 1person, which were present for the longest duration in the video, and therefore also had the most detected bounding boxes. Consequently, the model may have difficulties predicting other objects, making it challenging to determine the model's accuracy. To address this, it is necessary to generate a dataset with better balance for a model of this kind

Data cleaning and preprocessing were necessary aspects of this study, both during the development and training of the machine learning model. However, these processes could be more challenging in a real-time operational setting, potentially influencing the model's performance. This limitation, while recognized, provides directions for future research and improvements in the field. Additionally, implementing the model predicting gaze point in an actual simulator could reveal further challenges not identified in the research setting, such as the need for more robust error handling or real-time adjustments to changing conditions.

7.4 Future work

The research conducted in this study lays a foundation for developing eye movement simulations that can improve the realism of Laerdal Medical's simulators. However, to achieve the end goal of providing real-time object identification and eye movement based on identified objects, there are several important aspects that need to be addressed in future work.

7.4.1 Improving Real-time Object Detection Performance

The current system uses the YOLOv8 algorithm for object detection, which is a widely used method that provide a balance between speed and accuracy. However, as the demands of a simulation environment require real-time performance, further improvement of the object detection component is crucial.

Future work should explore more efficient and faster object detection methods to ensure the system can run effectively in real-time. Potential areas of exploration includes newer versions of object detection models that have been optimized for improved speed and accuracy. Alternatively, customizing the object detection model to better suit the specific situations for the manikin might provide further improvements. This may involve adding objects that the manikin should recognize into the object detection model. Additionally, it is crucial to note that for object detection with the manikin to be possible, cameras must be implemented in the manikin's eyes.

7.4.2 Developing a Real-time Version of the Model

The current LSTM model used for predicting eye movements was designed to work with pre-collected data. To adapt this for real-time use, the model needs to be capable of handling a continuous stream of incoming data.

This will involve reconstructing the data processing pipeline and possibly the model architecture to work in a real-time setting. This is a non-trivial task, as it requires maintaining the model's predictive accuracy while ensuring that computations can be performed within the time constraints of the simulation. The model should be capable of determining where to focus based on the objects present in the surroundings, discerning their salience. This decision-making process may not solely rely on visual input but could also consider sound and touch, which were not considered in this study. These additional sensory inputs have the potential to influence where to look.

7.4.3 Implementation of the Model in Medical Simulators

Beyond improving real-time object detection performance and developing a real-time version of the gaze point prediction model, future work could also explore how the model could be adapted to reflect individual differences in gaze behavior. This could involve developing gaze point prediction models that are able to find patterns across different participant, and use this to predict gaze movement based on these patterns. Incorporating user feedback to continuously update and improve the model's accuracy in predicting gaze behavior should also be included in future work.

Furthermore, more detailed work could be done on how the model could be implemented in Laerdal Medical's patient simulators to improve realism. This could involve working closely with simulator developers to integrate the model into the simulator's existing architecture, as well as testing the model in real-world simulation scenarios to evaluate its performance and make necessary refinements.

These steps could further improve the realism and educational value of Laerdal Medical's simulators, ultimately contributing to improved training and patient outcomes in real-life medical scenarios.

CHAPTER

EIGHT

CONCLUSIONS

This study forms a solid groundwork for the future development of a realistic patient simulator. The integration of eye-tracking and object detection data, offers a valuable insights into visual attention patterns, which have been used to develop a gaze point prediction model. The model has demonstrated promising performance and results, but further work is required to enhance its accuracy, improve real-time object detection, and adapt the model for real-time applications. Continued research and development in these areas can aim to create patient simulators with improved levels of realism.

REFERENCES

[1] Laerdal Medical. *Our story.* n.d. URL: `https://laerdal.com/about-us/`. (accessed: 01.12.2022).

[2] Unknown. "Laerdal Medical - saving lives through innovation". In: 150.6 (2005), pp. 31–31. ISSN: 0307-1847.

[3] Truls Nygaard et al. "From the eyes of the patient: real time gaze control of medical training mannequins". In: *ACM International Conference Proceeding Series*. NordiCHI '18. ACM, 2018, pp. 932–935. ISBN: 9781450364379.

[4] Laerdal Medical. *Advanced Care and Maintenance Courses.* n.d. URL: `https://laerdal.com/services-and-programs/educational-services/advanced-care-and-maintenance-courses/`. (accessed: 27.05.2023).

[5] Laerdal Medical. *The importance of realism in simulation-based learning for healthcare education.* URL: `https://laerdal.com/gb/learn/resource-library/the-importance-of-realism-in-simulation/`. (accessed: 01.12.2022).

[6] Thomas A. D'Agostino and Carma L. Bylund. "Nonverbal Accommodation in Health Care Communication". In: *Health communication* 29.6 (2014), pp. 563–573. ISSN: 1041-0236.

[7] Truls Hjertnes Nygaard. *Low Cost Human - Machine Interaction Platform for Bridging the Uncanny Valley, Using Humans as the Control System Input.* 2017. URL: `http://hdl.handle.net/11250/2615446`.

[8] Shensheng Wang, Scott O Lilienfeld, and Philippe Rochat. "The Uncanny Valley: Existence and Explanations". In: *Review of general psychology* 19.4 (2015), pp. 393–407. ISSN: 1089-2680.

[9] Benjamin T. Carter and Steven G. Luke. "Best practices in eye tracking research". In: *International journal of psychophysiology* 155 (2020), pp. 49–62. ISSN: 0167-8760.

[10] Aga Boiko. *Eye tracking the user experience : a practical guide to research.* eng. Brooklyn, N.Y: Rosenfeld Media, 2013. ISBN: 9781933820101.

[11] Jennifer Romano Bergstrom and Andrew Schall. *Eye Tracking in User Experience Design.* eng. San Francisco: Elsevier Science Technology, 2014. ISBN: 9780124081383.

[12] Pupil Dev Team Pupil Labs. *What is eye tracking?* Dec. 2020. URL: `https://pupil-labs.com/blog/news/what-is-eye-tracking/`. (accessed: 19.11.2022).

[13] Laurent Itti. "New Eye-Tracking Techniques May Revolutionize Mental Health Screening". eng. In: *Neuron (Cambridge, Mass.)* 88.3 (2015), pp. 442–444. ISSN: 0896-6273.

[14] Richard Andersson, Marcus Nystrom, and Kenneth Holmqvist. "Sampling frequency and eye-tracking measures: how speed affects durations, latencies, and more". eng. In: *Journal of eye movement research* 3.3 (2010), p. 1. ISSN: 1995-8692.

[15] Pupil Labs. *Best Practices.* n.d. URL: `https://docs.pupil-labs.com/core/best-practices/`. (accessed: 15.11.2022).

[16] Marcus Nyström et al. "The influence of calibration method and eye physiology on eyetracking data quality". eng. In: *Behavior research methods* 45.1 (2013), pp. 272–288. ISSN: 1554-351X.

[17] H. Kolb. *Facts and Figures Concerning the Human Retina.* Ed. by H. Kolb, E. Fernandez, and R. Nelson. University of Utah Health Sciences Center, 2005.

[18] Richard S Koplin et al. *The Scrub's Bible: How to Assist at Cataract and Corneal Surgery with a Primer on the Anatomy of the Human Eye and Self Assessment.* eng. 2013th ed. New York, NY: Springer New York, 2013. ISBN: 9781461456438.

[19] Tobii Connect. *Types of eye movements.* Sept. 2022. URL: `https://connect.tobii.com/s/article/types-of-eye-movements?language=en_US`. (accessed: 14.11.2022).

[20] Simon Liversedge, Iain Gilchrist, and Stefan Everling. *The Oxford Handbook of Eye Movements.* eng. 1st ed. Oxford Library of Psychology. Oxford: Oxford University Press, 2011. ISBN: 9780199539789.

[21] Robert Krueger, Steffen Koch, and Thomas Ertl. *SaccadeLenses: Interactive Exploratory Fltering of Eye Tracking Trajectories.* Oct. 2016. DOI: `10.1109/ETVIS.2016.7851162`.

[22] *Computer Vision.* eng. IntechOpen, 2008. ISBN: 9789537619213.

[23] Amazon Web Services. *What Is Computer Vision?* n.d. URL: https://aws.amazon.com/what-is/computer-vision/. Accessed: 27.05.2023.

[24] IBM. *What is computer vision?* n.d. URL: https://www.ibm.com/topics/computer-vision. Accessed: 27.05.2023.

[25] Himani Bansal. *How to get the Perfect start in AI ML as Newbie?? Learn the Art in just 5 mins!* July 2019. URL: https://becominghuman.ai/how-to-get-the-perfect-start-in-ai-ml-as-newbie-learn-the-art-in-just-5-mins-cba28d2705e4. Accessed: 19.04.2023.

[26] Vijay Kanade. *What Is Machine Learning? Definition, Types, Applications, and Trends for 2022.* Aug. 2022. URL: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-ml/. Accessed: 27.05.2023.

[27] Allan Bond. *Supervised, unsupervised, and reinforcement learning. The three pillars of machine learning.* July 2021. URL: https://medium.com/mlearning-ai/supervised-unsupervised-and-reinforcement-learning-the-three-pillars-of-machine-learning-fd898f0c7c81. Accessed: 27.05.2023.

[28] Alyshai Nadeem. *Machine learning 101: Supervised, unsupervised, reinforcement learning explained.* Sept. 2022. URL: https://datasciencedojo.com/blog/machine-learning-101/. Accessed: 27.05.2023.

[29] IBM. *What is deep learning?* n.d. URL: https://www.ibm.com/topics/deep-learning. Accessed: 19.04.2023.

[30] Eda Kavlakoglu. *AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference?* May 2020. URL: https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks. Accessed: 19.04.2023.

[31] Ian Goodfellow. *Deep learning.* eng. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016. ISBN: 9780262337373.

[32] Manav Mandal. *Introduction to Convolutional Neural Networks (CNN).* May 2021. URL: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/. Accessed: 27.05.2023.

[33] Christopher D Rodin et al. "Object Classification in Thermal Images using Convolutional Neural Networks for Search and Rescue Missions with Unmanned Aerial Systems". eng. In: (2018). ISSN: 2161-4393. URL: http://hdl.handle.net/11250/2579466.

[34] Nilesh Barla. *What Is Deep Learning? A Guide to Deep Learning Use Cases, Applications, and Benefits*. Feb. 2021. URL: https://clear.ml/blog/what-is-deep-learning/. Accessed: 27.05.2023.

[35] Ihab S. Mohamed. "Detection and Tracking of Pallets using a Laser Rangefinder and Machine Learning Techniques". PhD thesis. Sept. 2017. DOI: 10.13140/RG.2.2.30795.69926.

[36] Himadri Sankar Chatterjee. *A Basic Introduction to Convolutional Neural Network*. July 2019. URL: https://medium.com/@himadrisankarchatterjee/a-basic-introduction-to-convolutional-neural-network-8e39019b27c4. Accessed: 09.05.2023.

[37] Rikiya Yamashita et al. "Convolutional neural networks: an overview and application in radiology". eng. In: *Insights into imaging* 9.4 (2018), pp. 611–629. ISSN: 1869-4101.

[38] Jason Brownlee. *A Gentle Introduction to Object Recognition With Deep Learning*. May 2019. URL: https://machinelearningmastery.com/object-recognition-with-deep-learning/. Accessed: 09.05.2023.

[39] Kemal Oksuz et al. "Imbalance Problems in Object Detection: A Review". eng. In: *IEEE transactions on pattern analysis and machine intelligence* 43.10 (2021), pp. 3388–3415. ISSN: 0162-8828.

[40] Zhengxia Zou et al. "Object Detection in 20 Years: A Survey". eng. In: *Proceedings of the IEEE* 111.3 (2023), pp. 257–276. ISSN: 0018-9219.

[41] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". eng. In: *arXiv.org* (2016). ISSN: 2331-8422.

[42] Juan Terven and Diana Cordova-Esparza. "A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond". eng. In: (2023).

[43] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *YOLO by Ultralytics*. Version 8.0.0. Jan. 2023. URL: https://github.com/ultralytics/ultralytics.

[44] Dawood Ahmed et al. "Machine Vision-Based Crop-Load Estimation Using YOLOv8". eng. In: (2023).

[45] Armstrong Aboah et al. "Real-time Multi-Class Helmet Violation Detection Using Few-Shot Data Sampling Technique and YOLOv8". eng. In: (2023).

[46] Kaiyang Zhou et al. "Omni-Scale Feature Learning for Person Re-Identification". eng. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Vol. 2019-. IEEE, 2019, pp. 3701–3711. ISBN: 9781728148038.

[47] Jiechao Yang et al. "Combination of Convolutional Neural Networks and Recurrent Neural Networks for predicting soil properties using Vis–NIR spectroscopy". eng. In: *Geoderma* 380 (2020), p. 114616. ISSN: 0016-7061.

[48] Niklas Donges. *A Guide to Recurrent Neural Networks: Understanding RNN and LSTM Networks.* Feb. 2023. URL: `https://builtin.com/data-science/recurrent-neural-networks-and-lstm`. Accessed: 23.05.2023.

[49] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". eng. In: *Neural computation* 9.8 (1997), pp. 1735–1780. ISSN: 0899-7667.

[50] Caihong Hu et al. "Deep learning with a long short-term memory networks approach for rainfall-runoff simulation". eng. In: *Water (Basel)* 10.11 (2018), p. 1543. ISSN: 2073-4441.

[51] Malti Bansal, Apoorva Goyal, and Apoorva Choudhary. "A comparative analysis of K-Nearest Neighbor, Genetic, Support Vector Machine, Decision Tree, and Long Short Term Memory algorithms in machine learning". eng. In: *Decision analytics journal* 3 (2022), p. 100071. ISSN: 2772-6622.

[52] Lam Van Nguyen et al. "Predicting Discharges in Sewer Pipes Using an Integrated Long Short-Term Memory and Entropy A-TOPSIS Modeling Framework". eng. In: *Water (Basel)* 14.3 (2022), p. 300. ISSN: 2073-4441.

[53] Shankar297. *Understanding Loss Function in Deep Learning.* June 2022. URL: `https://www.analyticsvidhya.com/blog/2022/06/understanding-loss-function-in-deep-learning/`. Accessed: 23.05.2023.

[54] Roi Yehoshua. *Loss Functions in Machine Learning.* May 2023. URL: `https://towardsdatascience.com/loss-functions-in-machine-learning-9977e810ac02`. Accessed: 23.05.2023.

[55] Ayush Gupta. *A Comprehensive Guide on Optimizers in Deep Learning.* Oct. 2021. URL: `https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/#What_Are_Optimizers_in_Deep_Learning?`. Accessed: 23.05.2023.

[56] Diederik P Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". eng. In: (2014).

[57] Jason Brownlee. *Overfitting and Underfitting With Machine Learning Algorithms.* Mar. 2016. URL: `https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/`. Accessed: 28.05.2023.

[58] Nitish Srivastava et al. “Dropout: A simple way to prevent neural networks from overfitting”. eng. In: *Journal of machine learning research* 15 (2014), pp. 1929–1958. ISSN: 1532-4435.

[59] Tavish Srivastava. *12 Important Model Evaluation Metrics for Machine Learning Everyone Should Know (Updated 2023).* Aug. 2019. URL: `https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/`. Accessed: 22.05.2023.

[60] Aditya Mishra. *Metrics To Evaluate Your Machine Learning Algorithm.* Feb. 2018. URL: `https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234`. Accessed: 22.05.2023.

[61] David Christie and Simon P. Neill. “8.09 - Measuring and Observing the Ocean Renewable Energy Resource”. In: *Comprehensive Renewable Energy (Second Edition).* Ed. by Trevor M. Letcher. Second Edition. Oxford: Elsevier, 2022, pp. 149–175. ISBN: 978-0-12-819734-9.

[62] Aman Kharwal. *R2 Score in Machine Learning.* June 2021. URL: `https://thecleverprogrammer.com/2021/06/22/r2-score-in-machine-learning/`. Accessed: 22.05.2023.

[63] Moritz Kassner, William Patera, and Andreas Bulling. “Pupil: an open source platform for pervasive eye tracking and mobile gaze-based interaction”. eng. In: *Proceedings of the 2014 ACM International Joint Conference on pervasive and ubiquitous computing.* UbiComp ’14 Adjunct. ACM, 2014, pp. 1151–1160. ISBN: 9781450330473.

[64] PsychoPy. *Pupil Labs - Core.* Feb. 2022. URL: `https://psychopy.org/api/iohub/device/eyetracker_interface/PupilLabs_Core_Implementation_Notes.html#pupil-labs-core`. Accessed: 22.11.2022.

[65] Pupil Labs. *Hardware.* n.d. URL: `https://docs.pupil-labs.com/core/hardware/`. Accessed: 15.11.2022.

[66] Jonathan Peirce et al. “PsychoPy2: Experiments in behavior made easy”. eng. In: *Behavior research methods* 51.1 (2019), pp. 195–203.

[67] Massey University - University of New Zealand. *VR Patient Experience.* YouTube. Dec. 2018. URL: `https://www.youtube.com/watch?v=A7L8RGULwxs`. Accessed: 05.05.2023.

[68] APRIL Robotics Laboratory. *AprilTag.* n.d. URL: `https://april.eecs.umich.edu/software/apriltag.html`. Accessed: 19.11.2022.

[69] Pupil Labs. *Pupil Capture.* n.d. URL: `https://docs.pupil-labs.com/core/software/pupil-capture/`. Accessed: 19.11.2022.

[70] Google Inc. *Colaboratory: Frequently Asked Questions.* URL: https://research.google.com/colaboratory/faq.html. Accessed: 06.05.2023.

[71] NVIDIA Corporation. *NVIDIA T4 70W LOW PROFILE PCIe GPU ACCELERATOR Product Brief.* Apr. 2020. Accessed: 06.05.2023.

[72] Meta AI. *PyTorch.* 2023. URL: https://ai.facebook.com/tools/pytorch/. Accessed: 06.05.2023.

[73] Nikolaj Buhl. *YOLO models for Object Detection Explained [YOLOv8 Updated].* Feb. 2023. URL: https://encord.com/blog/yolo-object-detection-guide/. Accessed: 22.05.2023.

[74] Mikel Broström. *Real-time multi-object, segmentation and pose tracking using Yolov8 with DeepOCSORT and LightMBN.* Version 8.0. DOI: https://zenodo.org/record/7629840. URL: https://github.com/mikel-brostrom/yolov8_tracking.

[75] Qilian Liang et al. "Cross-Domain Person Re-identification: A Review". eng. In: *Artificial Intelligence in China.* Vol. 653. Lecture Notes in Electrical Engineering. Singapore: Springer Singapore Pte. Limited, 2021, pp. 153–160. ISBN: 9789811585982.

[76] Scikit-learn Developers. *LabelEncoder - scikit-learn documentation.* n.d. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html. Accessed: 24.05.2023.

[77] Yugesh Verma. *A Complete Understanding of Dense Layers in Neural Networks.* Sept. 2021. URL: https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/. Accessed: 24.05.2023.

[78] Jason Brownlee. *Use Early Stopping to Halt the Training of Neural Networks At the Right Time.* Dec. 2018. URL: https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/. Accessed: 24.05.2023.

APPENDICES

A - DWELL TIME BAR CHARTS

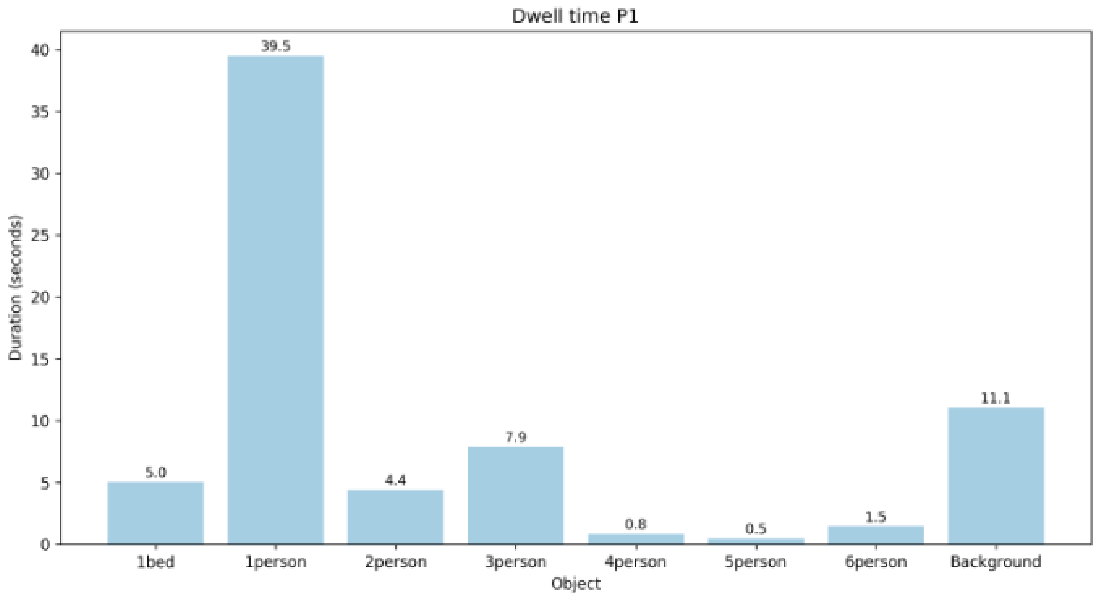

Figure .0.1: Dwell time Participant 1

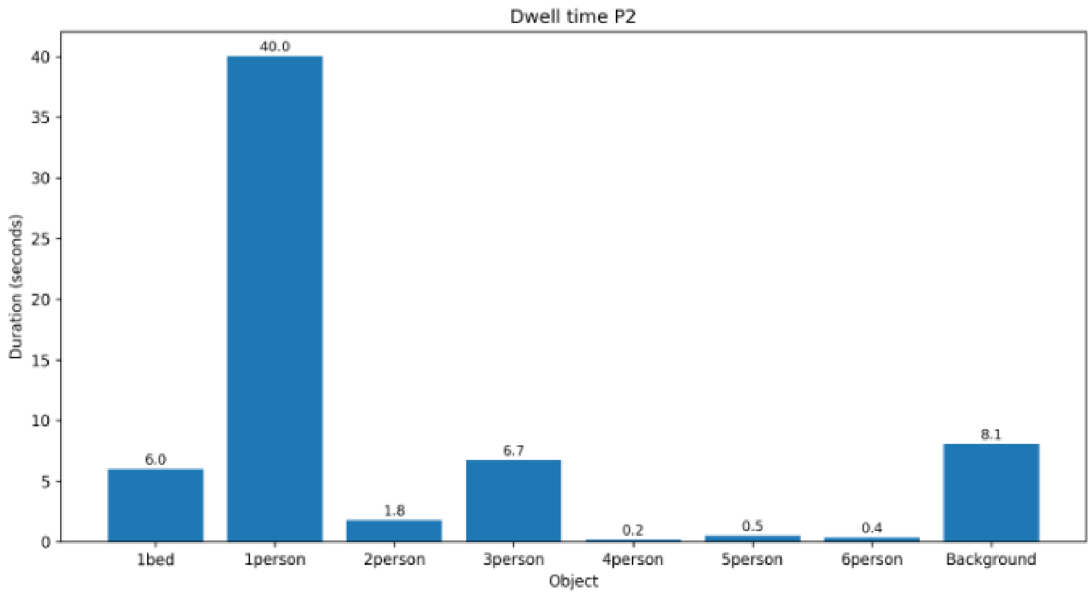

Figure .0.2: Dwell time Participant 2

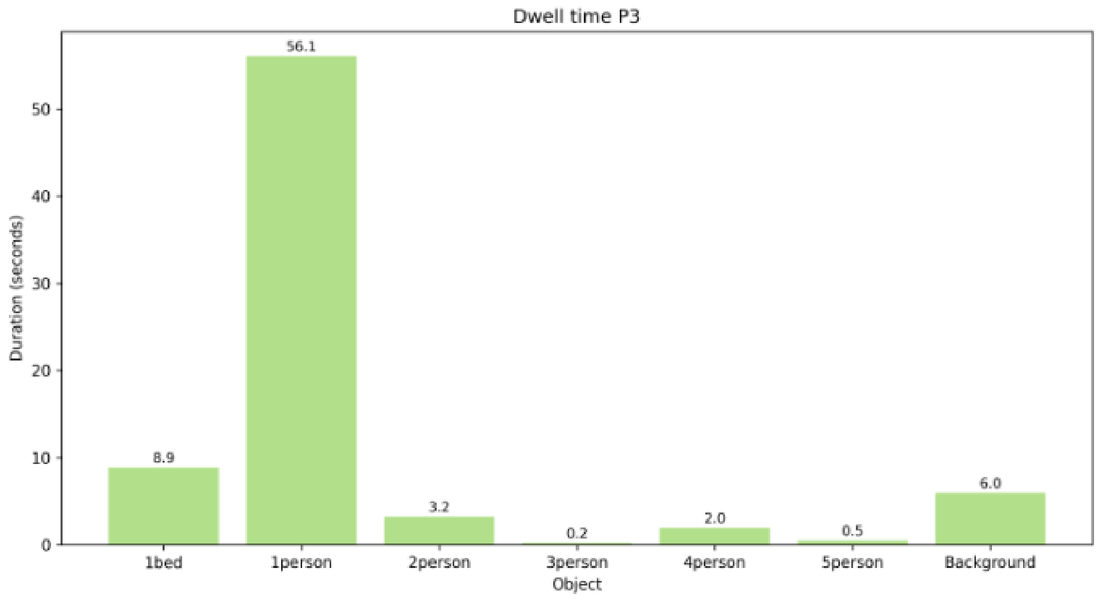

Figure .0.3: Dwell time Participant 3

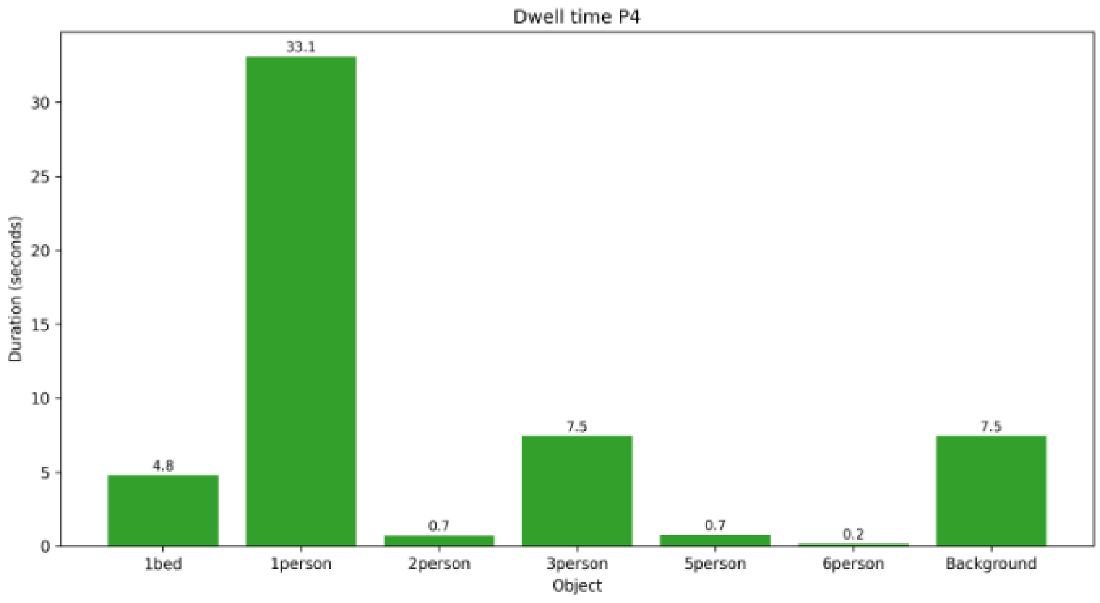

Figure .0.4: Dwell time Participant 4

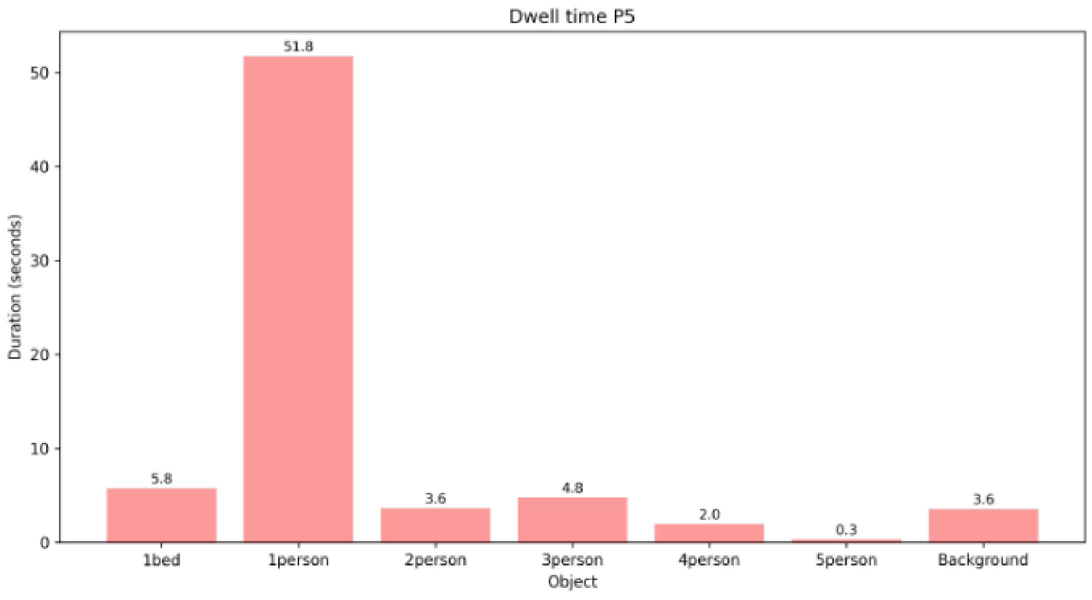

Figure .0.5: Dwell time Participant 5

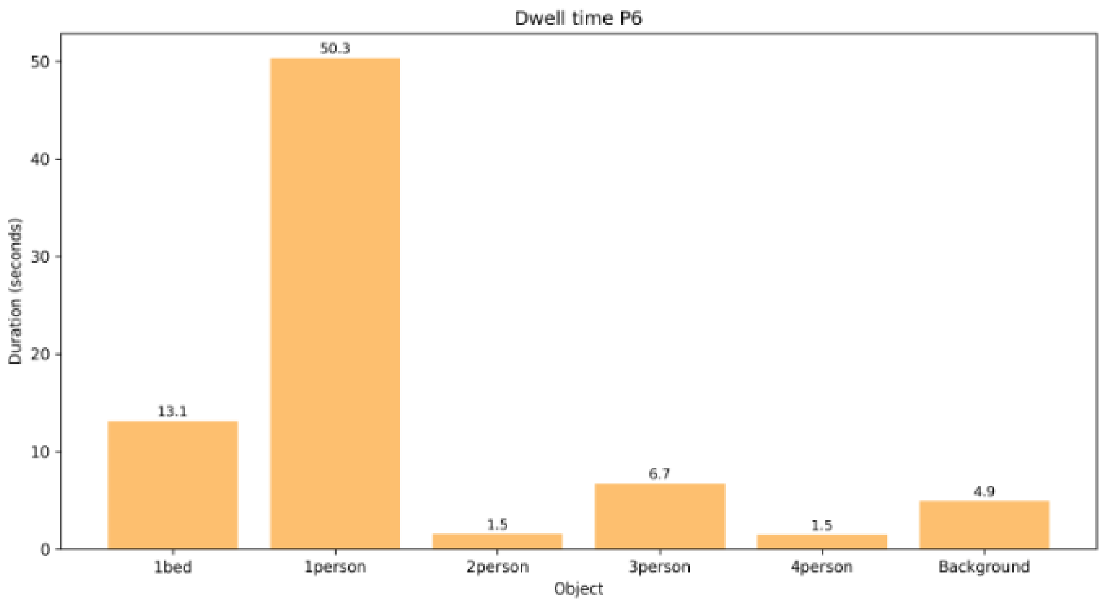

Figure .0.6: Dwell time Participant 6

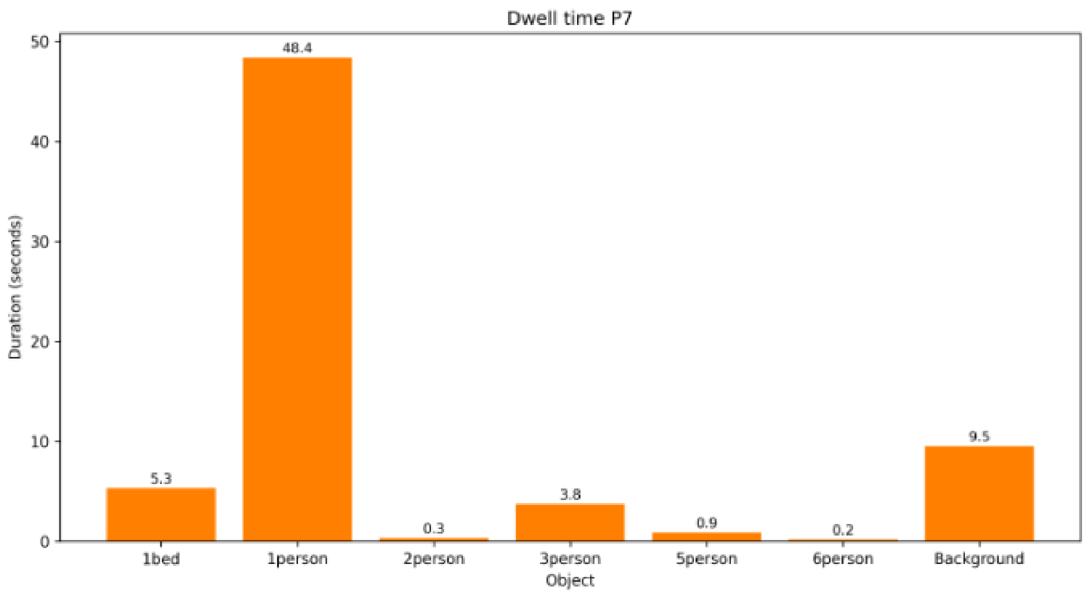

Figure .0.7: Dwell time Participant 7

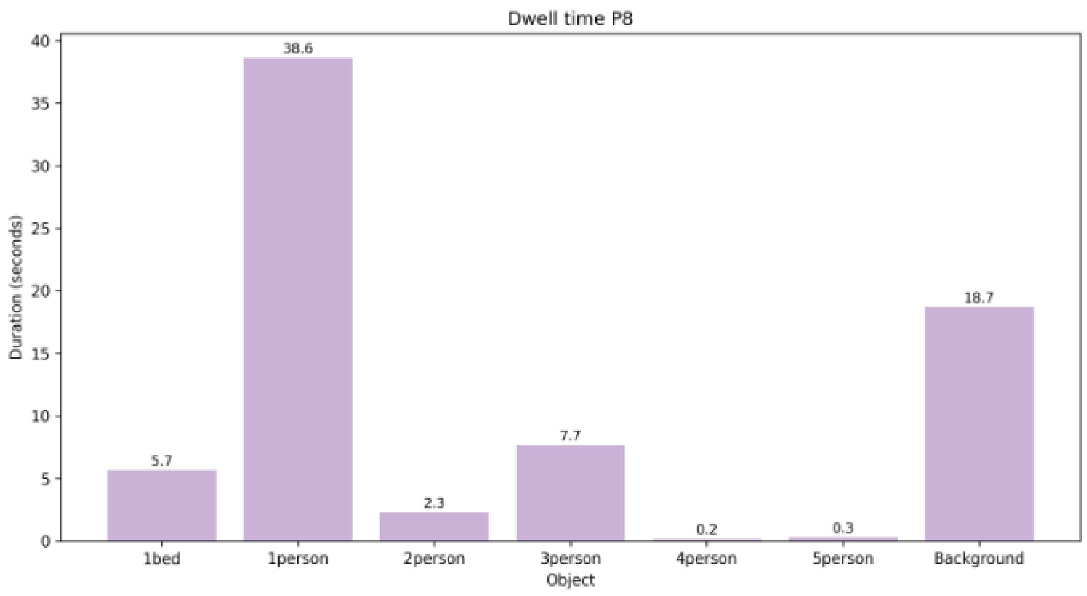

Figure .0.8: Dwell time Participant 8

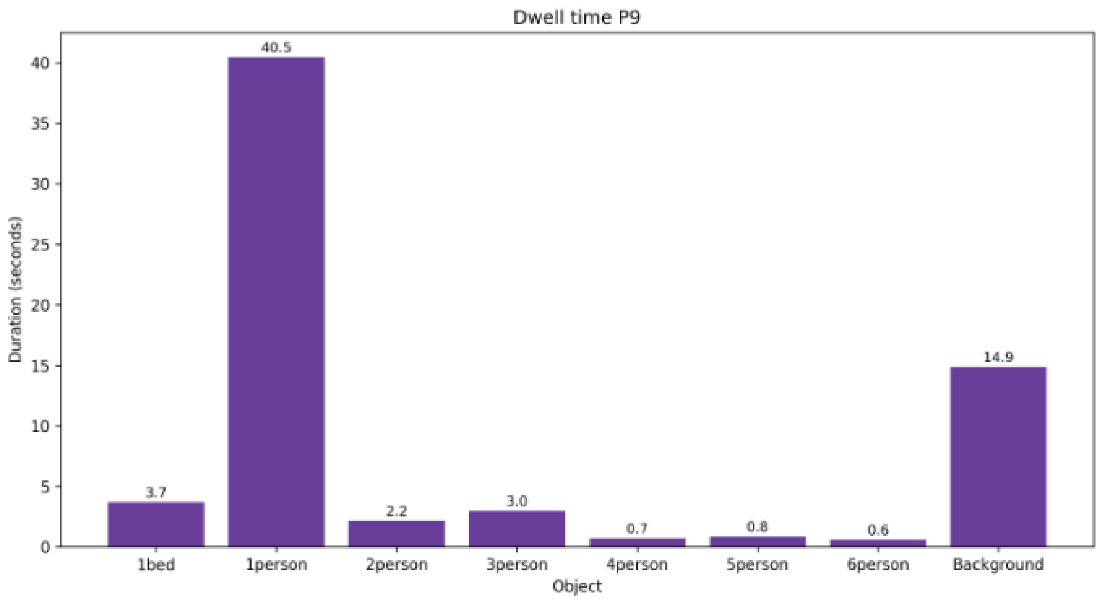

Figure .0.9: Dwell time Participant 9

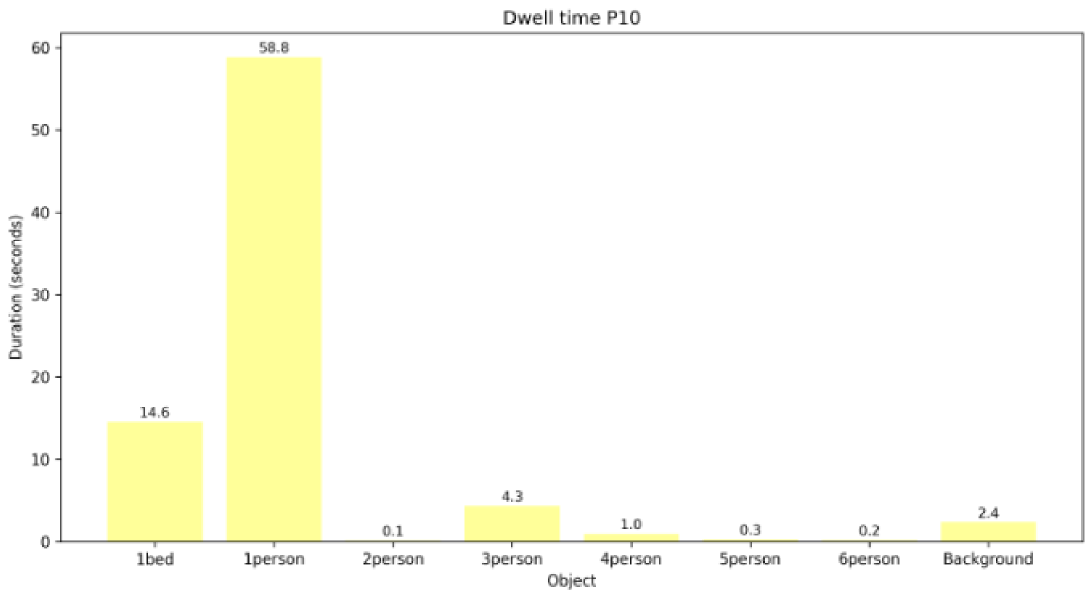

Figure .0.10: Dwell time Participant 10

B - GAZE PREDICTION MODEL

```python
import pandas as pd
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.callbacks import EarlyStopping

# Load data
df = pd.read_csv('final_dataset.csv')

# Prepare data
features = ['norm_pos_x', 'norm_pos_y', 'norm_pos_x_lag_1', 'norm_pos_y_lag_1',
            'norm_pos_x_lag_2', 'norm_pos_y_lag_2',
            'norm_pos_x_lag_3', 'norm_pos_y_lag_3',
            'Width', 'Height', 'X-mid', 'Y-mid']

# The "Object In Frame" column is categorical, so convert it to numerical
le = LabelEncoder()
df['Object In Frame'] = le.fit_transform(df['Object In Frame'])
features.append('Object In Frame')

# The next position to be looked at is our target
next_pos_x = df['norm_pos_x'].shift(-1)  # shifting data up by 1
next_pos_y = df['norm_pos_y'].shift(-1)  # shifting data up by 1

df['next_pos_x'] = next_pos_x
df['next_pos_y'] = next_pos_y

df.dropna(inplace=True)  # remove the last row which is NaN after shifting
```

```
X = df[features]
y = df[['next_pos_x', 'next_pos_y']]

# Changed shape to (samples, timesteps, features)
X = X.values.reshape(X.shape[0], 1, X.shape[1])

# Split data into train and test sets
split_index = int(len(X) * 0.8)
X_train, X_test = X[:split_index], X[split_index:]
y_train, y_test = y[:split_index], y[split_index:]

# Define LSTM model
model = Sequential()
model.add(LSTM(128, return_sequences=True,
        input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))
model.add(LSTM(64))
model.add(Dropout(0.2))
model.add(Dense(2))  # 2 output features: next_pos_x and next_pos_y

# Compile model
model.compile(loss='mse', optimizer='adam', run_eagerly=True)



# Definer EarlyStopping-kriterier
early_stopping = EarlyStopping(monitor='val_loss', patience=10)

# Fit model
model.fit(X_train, y_train, epochs=50, batch_size=32,
    validation_data=(X_test, y_test), callbacks=[early_stopping])

# Evaluate model
mse = model.evaluate(X_test, y_test)
print(f'Mean Squared Error: {mse}')
```

NTNU
Norwegian University of
Science and Technology