Context-Aware Gaze-Based Interface for Smart Wheelchair

Tien Pham

CONTEXT-AWARE GAZE-BASED INTERFACE FOR SMART WHEELCHAIR

by

TIEN PHAM

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2023

To my mother Yen, my father Hong, my sister Tuyen, and my girlfriend Quyen.

who motivated me to achieve this milestone

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Huber for constantly motivating and encouraging me, and also for his invaluable advice during the time that I work on the thesis. I wish to thank my academic advisors Dr. Kamangar and Dr. Beksi for their interest in my research and for taking the time to serve on my dissertation committee.

I would also like to extend my appreciation to my labmate Khang and Tuan to help me conduct the experiment, and discuss with me about the project. I would also give a special thanks to Thomas Nguyen, my friend who helped me a lot with experiments and grades. I am especially grateful to people who have given me good or bad experiences during the time I study in grad school at UTA. I also want to express my gratefulness to the Department of Computer Science and Engineering at UTA who sponsor my GTA.

Finally, I would like to express my deep gratitude to my family and my girlfriend who have encouraged, inspired, and motivated me to overcome all the loneliness, stress, and irritation that I experienced while studying aboard.

May 4, 2023

ABSTRACT

CONTEXT-AWARE GAZE-BASED INTERFACE FOR SMART WHEELCHAIR

TIEN PHAM, M.Sc.

The University of Texas at Arlington, 2023

Supervising Professor: Manfred Huber

Human-Computer Interfaces (HCI) is an essential aspect of modern technology that has revolutionized the way we interact with machines. With the revolution of computers and smart devices and the advent of autonomous vehicles and other machines, there has been a significant advancement in this area that brings convenience to users to interact with technology intuitively and efficiently. However, the importance of HCI goes beyond the convenience of everyday technology. It has become crucial in the development of assistive technologies that empower people with disabilities to live more independently. Person with disabilities, who lack control of one or more parts of their physical body or who have mental limitation have to interact with the machine in a special often very custom way that match their individual capabilities. One common machine that many people with severe physical disabilities have to interact with every day is the wheelchair which has been used for decades to facilitate their lives. While many times the common simple interfaces which are usually available on wheelchairs, such as joysticks or sip and puff interfaces, are sufficient, they are difficult to use for persons with severe disabilities who do not have proper control of their hands or are inconvenient and hard to utilize. This need as well as the quest for more intuitive, less overhead control leads to research for other ways to interact with a wheelchair in the context of partially autonomous navigation. In this thesis, a context-aware gaze-based interface is developed to allow users to control the wheelchair naturally without translating the user's eye gaze input to specific commands. The system can estimate eye gaze directions and analyze the location users are looking at to obtain the context for inference of user navigation intention. A navigation detection model is also embedded into the system that can distinguish between users' navigation intention, navigation-related attention or non-navigation attention to serve as a driver of semi-autonomous smart wheelchair systems.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Motivation

How can a computer system understand what we want to do? This is always an interesting question in the research of Human-Computer Interaction (HCI), an area that has been developing over the few recent decades with the explosion of electronic smart devices and the advancements in computing systems. HCI has been deployed in various application domains, from smartphones to virtual assistants, from television to autonomous vehicles, to facilitate the convenience of our daily life. However, the effect on and importance for people with disabilities has often been underestimated. They account for a large segment of the human population andand can often significantly benefit from specialized ways to interact with assistive devices due to their physical limitations. An interesting example of this problem is the way disabled persons interact with the electrical wheelchair which has been investigated by various research efforts. The simplest way is to ask the user to give the machine distinct commands from various means such as joystick input or vocal commands so that the machine could understand the intention of users. However, those approaches are limited by transferring the intentions into commands which causes the users a certain amount of cognitive load and forces them to divert attention towards the command generation task. This raises a research problem of **how the smart wheelchair can understand the navigation intention of the user without requiring an additional cognitive task**. In order to achieve that, the smart wheelchair needs to (1) get the navigation signal from users and (2) understand the intention of those navigation activities directly from the normal user behavior. This leads to a research statement for this thesis: A navigation intention detection system applied on smart wheelchairs.

Attention detection is an emerging technology that has the potential to revolutionize the way we navigate through our daily lives. By incorporating attention detection into navigation systems, we can create more efficient and safer routes that take into account the user's level of engagement with their surroundings. For smart wheelchairs, this would allow users to operate the device more intuitively while keeping their attention on the surrounding, but its significance goes beyond this to all types of driving, as distracted driving is a leading cause of accidents on our roads. With attention detection, navigation systems can monitor the driver's level of focus and provide alerts or rerouting recommendations when necessary.

1.2 Related Work

A Smart Wheelchair is an advanced type of wheelchair that uses sensors, cameras, and other technologies to help individuals with disabilities navigate their surroundings with greater ease and independence [5]. These wheelchairs are equipped with a range of features and capabilities that make them different from traditional manual wheelchairs or motorized wheelchairs. Smart wheelchairs may have features such as obstacle detection, automatic braking, and advanced navigation capabilities. There are several interfaces that have been developed for smart wheelchair ranging from joysticks [6, 7, 8], air pressure [3], tongue-based [3, 9], physiological signal-based [3], voice-based [10], head movement and eye tracking [11]. This section discusses some of the existing Human-Wheelchair Interface types with their advantages and disadvantages.

1.2.1 Joystick Interface

Joystick interfaces are one of the most common types of input devices used in smart-wheelchair systems and have been commercialized into real products for decades. A joystick is a lever that can be moved in different directions, and its position is used to control the movement of the wheelchair. Due to its popularity, the joystick interface is easy to use, requiring minimal training and experience. This makes them an ideal input device for individuals with limited mobility or cognitive impairments. Furthermore, the joystick can be customized in terms of sensitivity, speed, and other parameters, providing users the ability to precisely control the wheelchair with safety assurance. However, users have to have relatively good control of their hands and have to always focus on controlling the wheelchair with the physical effort of holding the joystick, which often limits the maneuvering ability and imposes additional cognitive tasks. Figure 1.1 shows an example of a smart wheelchair with a joystick interface [1].

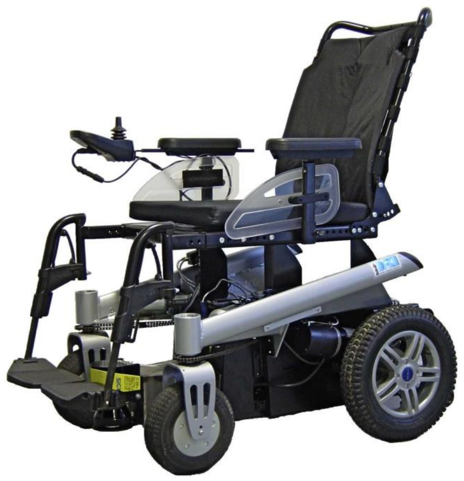


Figure 1.1. Joystick Interface for Smart Wheelchair. Ottobock XENO [1].

1.2.2 Voice-Based Interface

Due to the recent advanced achievement in Natural Language Processing (NPL), Voice-Based Interfaces have gained increasing popularity and have been widely used for a wide range of smart devices such as Alexa from Amazon, Google Home, Siri, and have been made available through multiple commercial and open source software systems to use voice command. Voice Command systems for smart wheelchairs have been developed and tested by multiple research groups [12, 13, 14].

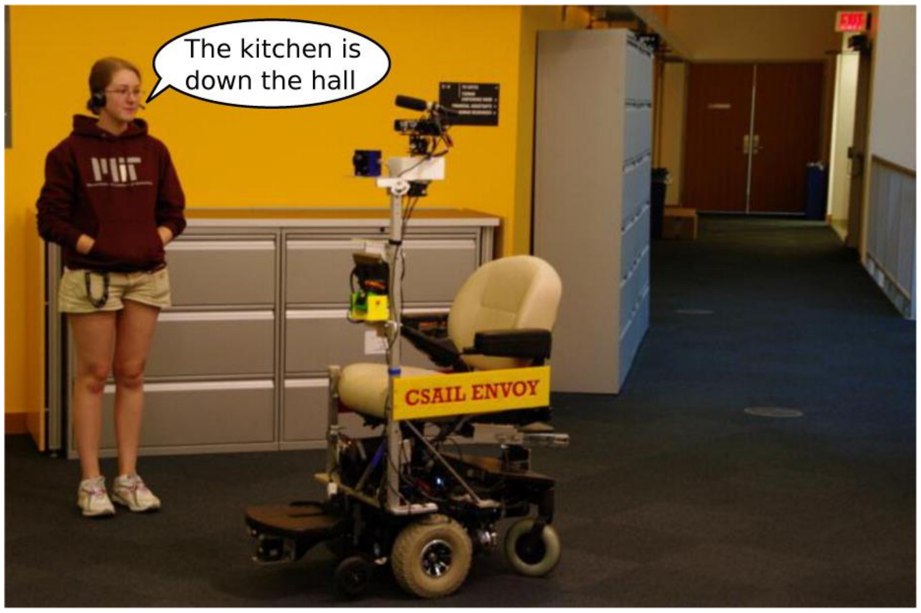


Figure 1.2. MIT Intelligent Wheelchair Project[2].

A voice-based interface for smart wheelchairs is a type of assistive technology that allows users to control their wheelchairs using their voice. This technology uses speech recognition software to understand the user's voice commands and convert them into actions that the wheelchair can perform. The voice-based interface can be used to control various features of the wheelchair, such as steering, speed, and braking. The development of a voice-based interface for smart wheelchairs involves several challenges. One of the biggest challenges is designing a speech recognition system that can accurately recognize a wide range of accents, dialects, and speech patterns. Another challenge is designing a natural language processing system that can understand the user's intent regardless of the ambient noise which is very common in real-life settings. In particular the latter has generally limited the deployment of such systems to very limited and controlled environments to avoid unintended movements due to mistakes of the recognition system. Figure 1.2 shows an example of an intelligent wheelchair with a voice interface [2].

1.2.3 Physiological Signal Interface

Physiological Signals are signals generated by the human body activities such as brain signals (Electroencephalogram - EEG), muscle signals (Electromyogran - EMG), and eye signals (Electrooculogram - EOG) which have been used for a significant number of experimental and research HCI applications recently [15, 16, 17]. One of the primary advantages of physiological signals is that they can provide more natural and intuitive interaction between humans and computers, which can lead to improved user experience and increased efficiency. For instance, instead of using a traditional keyboard and mouse, physiological signals such as muscle tension and eye movements can be used for gesture recognition or to control a cursor on a screen, allowing for more intuitive and efficient navigation. Another advantage of using physiological signals in HCI applications is their ability to provide real-time and objective feedback on the user's emotional and cognitive states. By measuring physiological signals such as heart rate, brain waves, and skin conductance, the interface can provide insight into the user's stress levels, attention, and engagement with the system. This information can be used to adapt the system's interface to the user's needs and preferences, leading to a more personalized and effective user experience. An example of Brain-Computer interface for a smart wheelchair is shown in Figure 1.3 [3].

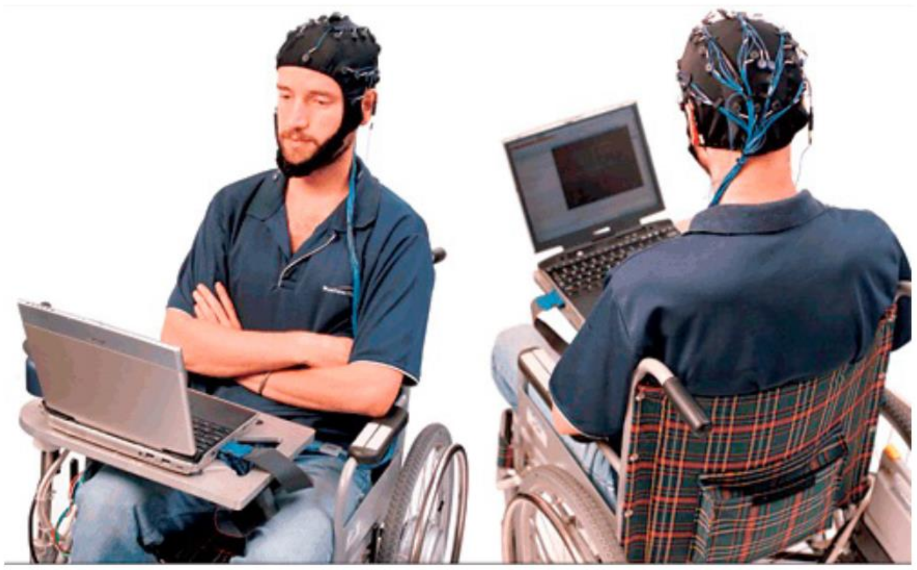


Figure 1.3. Brain-Computer Interface for Smart Wheelchair [3].

However, there are also some disadvantages to using physiological signals in HCI applications. One of the main challenges is the variability of physiological signals across individuals, which can make it difficult to develop a universal model for interpreting the signals. Additionally, physiological signals can be affected by various factors such as medication, illness, and environmental conditions, which can lead to noisy or unreliable data. This can make it challenging to accurately interpret the signals and use them as reliable inputs for HCI applications. Furthermore, physiological signals are relatively small (µV level) which makes the system very sensitive and prone to environmental noise. A final complication with the use of most physiological systems for interfaces is that they usually require specialized sensors to be worn by the user, thus increasing the overhead of activating the interface each time the user enters the wheelchair and frequently requiring a new calibration even for the same user to address the precise location of the sensors.

1.2.4 Head Movement and Eye Tracking Interface

A head movement and eye-tracking interface for a smart wheelchair is a human-computer interface (HCI) that enables individuals with limited mobility to control the movement of a wheelchair using their head and eye movements. This technology works by detecting the user's head movements and eye gaze direction through a combination of sensors and cameras and then translating these movements into commands for the wheelchair's movement. The head movement interface typically consists of a small sensor attached to the user's forehead or glasses, which measures the orientation and movement of the head in different directions. The eye-tracking interface, on the

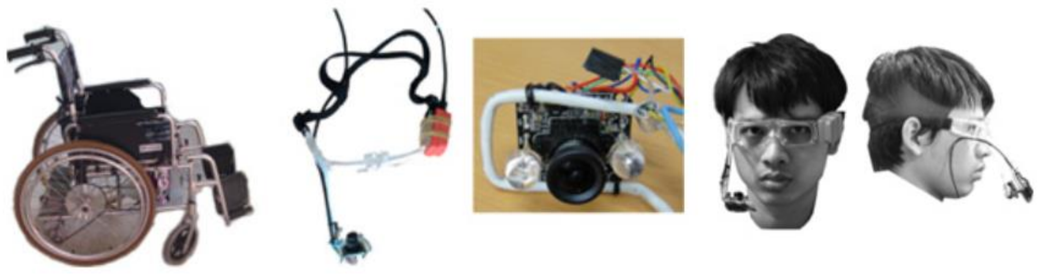


Figure 1.4. Gaze-Based Interface for Smart Wheelchair [4].

other hand, uses a camera to track the user's eye movements and gaze direction. By combining these two input modalities, users can control the direction and speed of the wheelchair's movement by simply moving their head or looking in the desired direction. For example, turning the head to the left could cause the wheelchair to turn left, while looking straight ahead could cause it to move forward. This type of interface can provide significant benefits for individuals with limited mobility, including those with spinal cord injuries, cerebral palsy, or other physical disabilities that limit their ability to use traditional joystick-based wheelchair controls. However, there are also some challenges to consider when using head movement and eye-tracking interfaces for smart wheelchairs. For example, users may experience fatigue or discomfort after prolonged use, as the constant movement of the head and eyes can be tiring. Additionally, the accuracy and reliability of the interface can be affected by various factors such as lighting conditions and the user's physical condition, which can lead to errors or misinterpretation of the user's movements. Moreover, as most of these interfaces define a set of specific movements or eye blink responses to identify commands, the use of these interfaces can interfere with and be influenced by common gaze and head movements that happen during common activities during navigation, such as the head movements or gaze changes when encountering a known person or when hearing an unexpected sound, causing additional cognitive load and requiring heavy concentration on the gaze-based navigation commands [18, 19].

1.2.5 Analysis

Each type of interface has its own advantages and disadvantages which raise the idea of using a hybrid approach with a priority on an intuitive, effortless, and reliable interface. Within this it is important that such a system does not generate incorrect or additional navigation instructions and that it can correctly and reliably identify the navigation instructions or intentions of the user.

- **Trigger interface**. A smart wheelchair is a vehicle which needs to be safe for the user to use. The trigger mechanism of the wheelchair that determines whether to run or be idle needs to be reliable and comfortable for the user to use. Electromyogram (EMG) seems to be the most suitable option when it relates to the lightweight cognitive, conscious task of the user, and can be embedded into multiple places on the user's body depending on the user's preference and availability of the muscle. EMG is the strongest and most reliable physiological signal which can be easily captured from the outer skin. The user can easily squeeze the leg, calfor muscles in the shoulders to generate a strong signal. While this still requires the wearing of special purpose sensors, its signal tends to be relatively robust with respect to small placement variations, at least if only a small number of commands are extracted.

- **Navigation Interface**. In addition to determining when to move and when not, the system also needs to identify the user navigation intention to move accordingly. Most of the prior works translate the inputs from the user into a new set of commands for a joystick interface following pre-defined rules. Take a hand gesture interface [20, 21] as an example. Here 4 gestures of the hand are set to 4 navigation commands such as *turn left*, *turn right*, *forward* and *stop*. While the signal variation between gestures can make the classification task easier and the system more reliable, the translation of gestures into commands places a significant cognitive burden on the user as this use of gestures for continuous navigation is not intuitive. This can make long-term usage of the system more difficult and may require users to undergo a significant training process to become familiar with the gesture-to-command translation rule. In the research presented in this thesis, eye gaze, one of the least effort signals for navigation, is chosen due to its intuitive relation to and use in navigation behaviors of humans. Users who look for longer periods to the left side will normally not want to turn to the right side and vice versa. However, eye gazing is not always used for navigation. Users may look at some distraction factors such as their friendsor a picture on the sidewall while continuing to go straight. That challenge raises an important issue which is the main scientific problem of this thesis: Could the system differentiate the pattern of navigation and non-navigation related eye gazing behavior? This will be discussed in the following sections and throughout the remainder of this thesis.

1.3 Human Visual Attention Detection and Navigation

The human visual system is the complex network of structures and processes that allows us to perceive and interpret visual information from our environment. At a high level, the human visual system can be divided into two main components: the eyes and the brain. The eyes are responsible for capturing light and converting it into neural signals that can be sent to the brain for processing. The brain, in turn, is responsible for interpreting these signals and generating visual perceptions. as one of the most salient senses, humans, as well as other creatures in the world, use eye gazing for a wide range of purposes, including navigation. This leads to the complicating factor for gaze-based navigation that humans will not always focus on the destination while moving. Instead, they can look around to see if anything happens that can affect the navigation intention, or if something attracts their attention. In order words, eye gazing, on the one hand, is one of the most intuitive factors representing navigation intention; on the other hand, it is difficult but imperative to distinguish navigation intention from other attention activities.

To solve this problem, the second part of the human visual system - ”the brain” - needs to be analyzed in order to understand the gaze activities of users. However, it is very difficult to capture the brain signal from the user due to the weak and prone-to-noise characteristics of brain signal sensing technologies as mentioned in the previous section. However, the intention could often be indirectly inferred from gazing behaviors as well as the environment settings. For example, it is unlikely a navigation intention if the user is looking upward or if they are looking at a wall. It is probably not a navigation intention if the users quickly look to the left or right side. Those logical interpretations give a promising motivation for navigation intention detection by analyzing the eye-gazing directions along with environmental perceptions representing the context of the gaze.

1.4 Context-Aware Interfaces

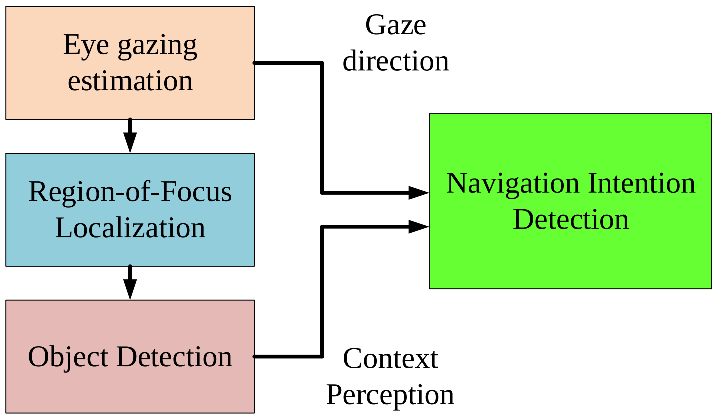


Figure 1.5. Context-Aware Gaze-Based Interface.

To capture the previously mentioned intentions, the smart wheelchair needs to be able to capture the eye-gazing signals from users and has to have the capability to interpret those signals. Firstly, the eye-gazing direction has to be measured as the basic input of the interface. There are several eye tracker systems in both wearable and non-wearable form factors such as Eyetribe [22], Tobii Pro Glass 3 [23], etc. They have the ability to detect eye pupils using RBG or Infrared Cameras and estimate the gaze direction. In this thesis, an RBG-based eye tracker component is implemented to estimate the eye-gazing direction of users, which is discussed in Section 2.2 in detail. Secondly, the system has to be able to capture the environmental context and analyze the position and objects that users are looking at. The objects, such as a person, a chair, or a car, to which the user is paying attention might have information regarding the navigation intention. From the eye gaze direction the region where the user is looking is identified and the system only needs to analyze the objects around that region. A feasible approach to solve this problem is to use a camera that has a similar view to the user and infer the region of user attention in the image captured by this camera. Using this concept, the context-ware interface used in this thesis has been built which facilitates the capability of interpreting the user navigation intention by measuring the gaze direction, localizing the region of user focus, and detecting the object in that region. An overview of the proposed context-aware gaze-based interface is shown in Figure 1.5.

CHAPTER 2

SYSTEM IMPLEMENTATION

2.1 Overview

The system proposed in this thesis was built in order to track the user's eye gaze as well as to detect if there is an object that they are looking at and what type of object it is. The system includes (1) one RGB camera placed towards the user's face, which plays a role as an eye tracker system, (2) one camera that has a relatively similar viewing angle to the user view to identify the region they are looking at and detect whether there is an object in that region. The two cameras are connected to (3) a computing device that receives the input from both cameras and runs the processing logic. In this project, a wheelchair is utilized as a use case for the system, although it is not limited to this particular device and can be applied to other machines as well. One of the reasons for this setup rather than the use of a wearable eye tracker is the goal to have a system that requires minimal activity from the user to operate as target users will often have limited dexterity and might not be able or forget to put on any wearable devices. The constructed wheelchair prototype is shown on the left in Figure 2.1 with a high level software architecture overview shown on the right of the figure.

Due to the intensive computational load of both the eye tracker and object detector, a processing pipeline is developed that leverages the multi-processing capabilities of the computing device by developing the eye tracker and object detector in different ROS nodes. The eye tracker node will publish the face topic, including head pose and eye gaze vector messages, while the object detector node will subscribe to

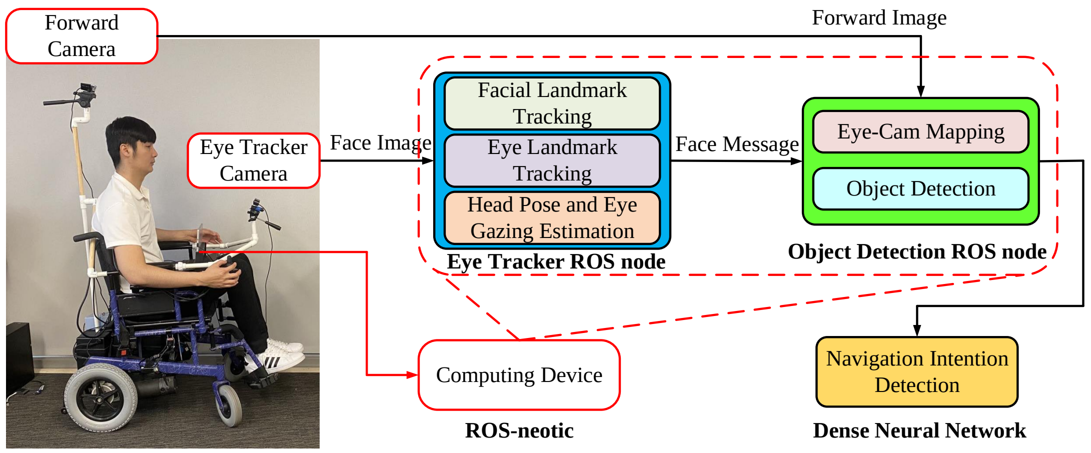


Figure 2.1. Overview of the System.

that topic and perform the detection algorithm. For the sake of being portable and easy to use, a Microsoft Surface Pro 4 is used as the main computing component. It has a relatively strong computing capability (4th Gen Intel Core i5-4300U processor, 2 physical CPUs 1.9GHz) compared to other tablet form factor devices. However, ROS is not well supported on Windows OS, while Linux OS is not officially supported by Microsoft Surface Pro 4. A customized Linux Kernel is developed to enable the peripherals as well as touch screen and camera of the Surface Pro 4 that follows the tutorials in [24]. We also adapt the system architecture at [25] to run different software components which serve different hardware devices and synchronize them using ROS messages. This architecture helps the system work with multiple communication protocols like USB or Ethernet, allowing high connectivity and synchronization [26].

2.2 Eye Tracker Component

The eye tracker component consists of an RBG camera that connects to a Linux machine running ROS-noetic and runs a series of computer vision processes to detect the head orientation and eye gaze. A PVC-piped prototype addition to an electric wheelchair is manufactured that can mount the camera in front of the user and capture the user's face. The distance between the camera and the user's face needs to be reasonable to balance stability which would favor short mounts and the need to have sufficient angle and distance to capture the face and both eye when the user turns the head. If the camera is too close, it can not see both eyes when users turn their head, which makes the following processing difficult. On the other hand, mounting the camera far from the face would lead to a smaller face and eye image in the camera frame, which also degrades the performance of the system. After capturing the face image, a series of facial analyses using computer vision are implemented, including: *facial landmark location estimation, head pose estimation, and eye gaze estimation.* The overall gaze estimation system works as a semi-pipeline process that gives us the result of eye gaze as shown in Figure: 2.2.

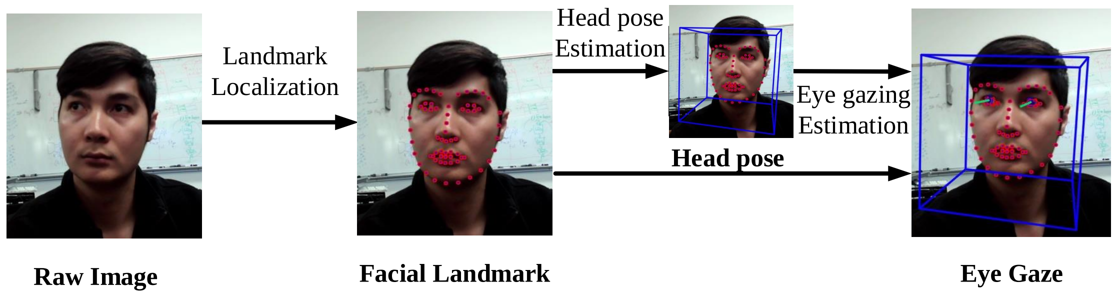


Figure 2.2. Eye Gaze Estimation Process.

2.2.1 Facial Landmark and Head Pose Estimation

Facial Landmark Detection. Facial landmarks are located using the Convolutional Experts Constrained Local Model (CE-CLM) to robustly detect the facial landmarks with the pre-defined constraints on the alignment of landmarks and the shape of the face using a distribution model [27]. The CE-CLM uses a Convolutional Experts Network to compute a response map that is used to localize the landmarks with the probability for each individual pixel [27]. While updating parameters, a Point Distribution Model with Regularized Landmark Mean Shift is used to update the positions of all landmarks and penalize misaligned or irregular shapes.

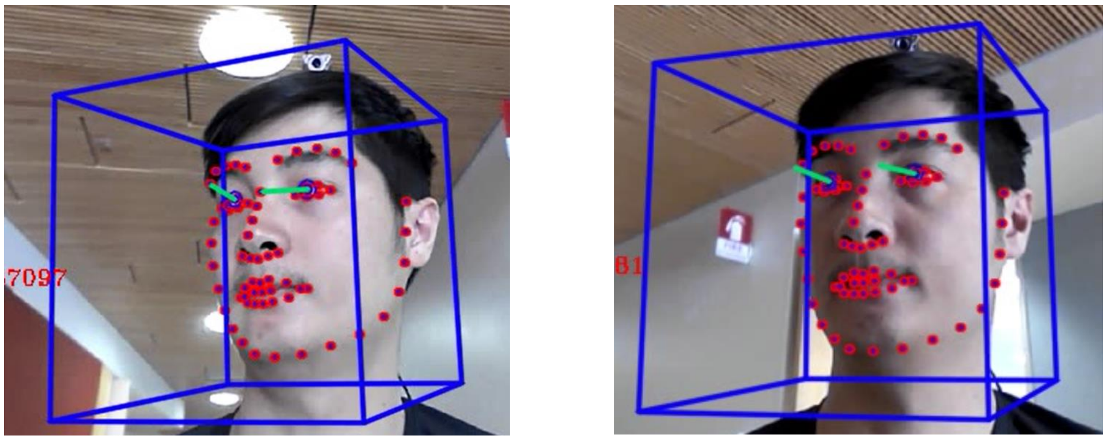
Convolutional Experts Network (CEN). The given image is used to extract a Region of Interest (ROI) based on the estimated position of the landmark. This ROI goes through a Contrast Normalizing Convolutional layer which performs Z-score normalization before the correlation operation, which outputs a response map. Next, the response maps are fed into a convolutional layer with ReLU units, followed by a Mixture of Expert Layer (ME-layer) to learn an ensemble that captures ROI variations. The ME-layer uses a convolutional layer with sigmoid probability decision kernels. The output response map is a non-negative and non-linear combination of neurons in the ME-layer using a sigmoid activation [27, 28].

Point Distribution Model. Active Appearance Model [29] and Deformable Model Fitting by Regularized Landmark Mean-Shift [30] are used to regularize the shape of the face constructed by the landmark positions. This stage will help to avoid local optima, to reduce the complexity of evaluation and to enhance the outlier detection by penalizing irregular shapes [27]. Additionally, the landmark detector relies on an initial parameter p_0 which is usually the result of the face detector. The optimal solution p then will need to have an update $\delta\mathbf{p}$ which can be solved by Regularised Landmark Mean Shift [30].

Head Pose Estimation. From the landmark locations, a 3D representation of facial landmarks is established in the projection of the camera using the Direct Least-Squares (DLS) approach [31]. In this approach, an n point perspective problem is formulated as the constrained non-linear least square minimization problem and solved by directly computing the local minima with the use of the Macaulay matrix [32]. The head pose, therefore, can be accurately estimated in the camera perspective efficiently.

2.2.2 Eye gazing estimation

In order to estimate the gaze vector, the eye landmarks, including the iris, eyelids, and pupils, are localized in 3D using a Constrained Local Neural Field (CLNF) [33]. The eyeball center location is calculated from the eyelid landmarks for each eye and considered as the origin of the eye gaze vector. The intersection between the image plane and the eye-ball sphere that contains the center of the pupil is calculated and the vector from the eyeball center to the pupil center that lies on the intersection is the estimated eye gaze vector. As a result, each eye gaze vector is estimated individually in 3D with respect to the camera coordinates. This is an efficient approach for eye gaze estimation that is proven to work accurately in screen-based settings [34]. However,



a) Unreliable Case b) Reliable Case

Figure 2.3. Example of Reliability and Unreliable of Eye Tracker.

due to the rotation of the head which distorts the landmark locations, especially with the constraint on the pupil-eyelid alignment, the eye gaze vector is not reliable n the presence of large head rotations with respect to the camera. An example of the errors that can occur in the context of larger head rotations is shown in Figure 2.3. As such, a compensation technique is suggested here to mitigate the effects of head rotation on eye gaze estimation accuracy. This involves applying a rotation matrix, corresponding to the head pose, to the estimated eye gaze vector. This intuitive and efficient error correction method aims to reduce the impact of head rotation on the estimated eye gaze vector. The final eye gaze vector of the user is computed by averaging the two gaze vectors for the two eyes in order to improve accuracy.

There is another major problem that can be observed in the context of individual estimates of the gaze for each eye in that sometimes only one vector is reliable while the other one is too far off (see Fig. 2.3 for an example). This problem is caused by errors in eye landmark localization (in this case the upper eyelid) which results in the error in gaze direction. Naturally, human eyes normally are not controlled differently, which can be applied as an additional constraint for eye gazing estimation. The rule of final eye gaze estimation is illustrated in Algorithm 1, which performs the angle corrections outlined above.

2.2.3 Eye Tracker ROS Node Performance

The eye tracker ROS node can fully utilize one virtual core of the CPU. Its computational load varies depending on the number of faces that appear on the image frame. The more faces appear in the frame, the more processing time needs to be spent, degrading the performance of the whole system. Furthermore, the resolution of the image is also a factor that affects the performance. In order to speed up the eye tracker performance, the region of interest is calibrated in both hardware (by

Algorithm 1 Correction Algorithm for Eye Tracker

```
1: Input: $L$, $R$, $H$                    ▷ Input left gaze, right gaze vector and head angle
2: Output: $gaze\_angle$                                                          ▷ Output
3: if $|L.x - R.x < 0.15|$ and $|L.y - R.y < 0.1|$ then
4:     $gaze\_angle \leftarrow \frac{L+R}{2}$
5: else
6:     if $H.x > 0$ then
7:         $gaze\_angle \leftarrow R$
8:     else
9:         $gaze\_angle \leftarrow L$
10: return $gaze\_angle$                               ▷ Return the final gaze angle
```

adjusting the camera angle so the user face position is n the center of the frame) and software (by cropping the image so it contains only the face with a resolution of 640x480). With this configuration, the eye tracker ROS node can operate at around 12fps which is adequate for this application.

2.3 Object Detection Component

The information about the object that the user is looking at contains important context information whether it is a distracting object or something related to navigation. For example, it is frequently a distraction if the users are looking at a person, a television, or a chair while they are moving. By contrast, if the users are looking at a door or a person moving toward their path, there is a high chance that their gaze is related to the navigation activity. Therefore, object detection in the direction of the gazing orientation is very important. An Object Detection Component will receive the gaze information from the Eye Tracker Component via ROS messages and analyze the region of interest that the user is focusing on. Thus, a second camera in the system that has a similar view as the user needs to be designed in order to capture the frontal view and analyze the user attention region.

Without the use of wearables, it is generally impossible to place the camera so it has the exactly same viewing angle as the user. This introduces a problem of image plane mapping between the user and camera perception addressed in Section 3.2.4. In a practical application, the camera can be either mounted at the same height as the user's eyes, which mitigates the distortion on the up-down angle, or above the user's head to reduce the left-right angle error. However, it is not possible to align it in both angles and thus at least one orientation alignment needs to be sacrificed and the angle difference needs to be compensated for. As in the context of wheelchair navigation horizontal accuracy is generally more important than vertical one where the viewing range and the operational range are limited, the camera is mounted above the user's head as closely as possible. A PVC-pipe/Wooden prototype is fabricated that attaches to the back of the wheelchair and has the Tripod mount on top to adjust the angle.

2.3.1 Object Detection Model

The Object Detection Component needs to recognize whether there is an object in side the user ROI. There are several available object detection models that can work reliably, namely You Only Look Once (YOLO) [35, 36, 37], Region-Based Convolutional Neural Network (R-CNN) [38, 39], Single Shot Detection (SSD) [40], MobileNet [41] and so on. Due to the constraint on computing resources, MobileNet is used here due to its efficient, lightweight model, as well as its ability to work well on mobile devices. MobileNet is based on SSD with the optimization on resource constraint devices and has comparable performance with other models [41]. The model is trained on the COCO dataset [42] as it contains a rich variety of labels (90 common objects)and can thus provide good context information for navigation intention identification.

2.3.2 Object Detection ROS Node

OpenVINO (Open Visual Inference and Neural Network Optimization) is an open-source toolkit developed by Intel for optimizing and deploying deep learning models for computer vision applications [43]. It is designed to accelerate the inference (or prediction) phase of deep learning models on a wide range of Intel hardware, including CPUs, GPUs, FPGAs, and VPUs (Vision Processing Units). A ROS node that uses the OpenVINO toolkit [44] has been built to capture the image in a forward camera and analyze a region of interest that the system identifies based on the gaze direction.

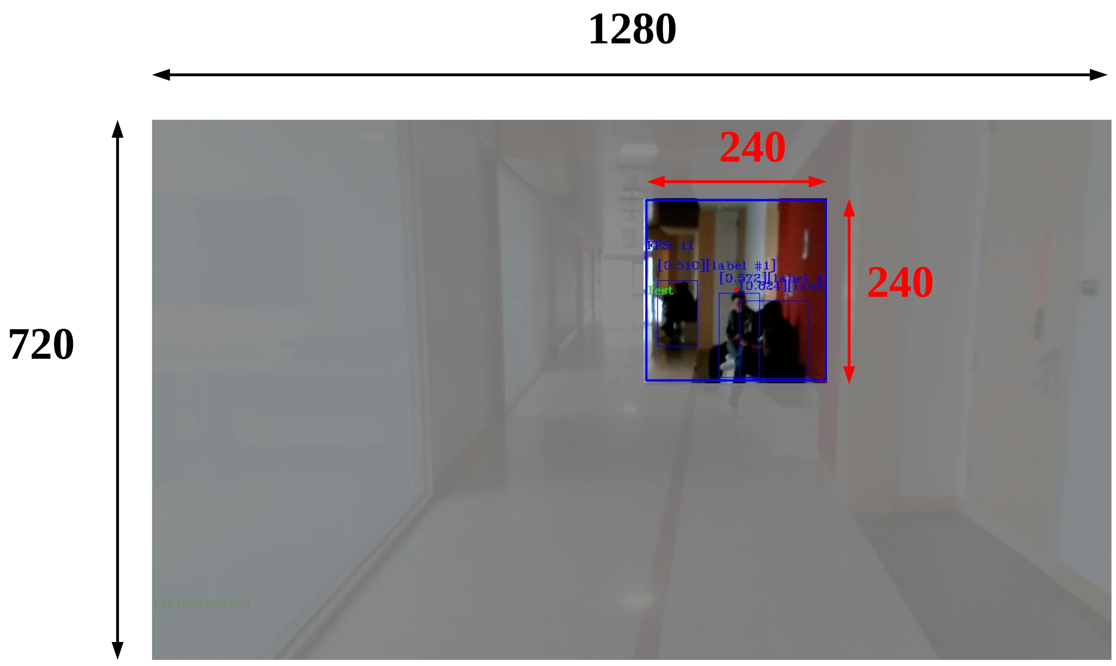


Figure 2.4. Configuration of Images in the Object Detection ROS Node.

The ROS node will operate based on the input of the face message received from the eye tracker component, then execute an eye-camera mapping algorithm to localize the region of interest that the user is looking at. A etailed description of the eye-camera mapping algorithm will be presented in Section 3. The expected processing rate of the ROS node would be the same as the Eye Tracker Component ROS node which is around 10fps. There are two main factors that can affect the performance of the system, namely the image size of the forward camera and the size of the ROI. The maximum resolution of the camera used is 1920x1080 which is relatively high and requires a remarkable amount of processing time, reducing the processing rate to around 5fps. The second highest resolution (1280x720) of the camera is chosen to balance resolution and processing time. The question of the size of the ROI is more complex, as it depends on the angle range of the camera and the size of the object that appears in the image, and its choice is directly related to the operational range of the system. A big ROI will result in less directional information but can tolerate a larger error of the eye-camera mapping function, while a smaller ROI provides better directional information but is more sensitive to gaze direction errors. After testing with various configurations, a ROI size of 240x240 is found as the balance trade-off between the aforementioned criteria.

CHAPTER 3

EYE-CAMERA MAPPING

3.1 Eye Gaze Estimation For Out-Of-The-Screen Usage

The performance of eye gaze estimation has previously been evaluated on MPI-IGaze data [34, 45] which is collected based on a screen-based application. This, however, might not be a correct estimate in the context of the wheelchair navigation application and turns out to be problematic when the user has a large head rotation relative to the camera, which distorts the landmarks that appear in the image. In evaluations in this application we observe that when the user turns the head too much, the landmarks detected have a higher error, which leads to wrong head pose and eye gazing estimation in some cases (see Figure 2.3). Therefore, a performance test on eye gaze estimation under large screen usage has been conducted. A big screen is placed in front of the user with a defined distance to displace multiple objects in fixed positions. A user is asked to sequentially look at the objects 10 times to get corresponding eye gaze estimates. The averages of gaze angles compared to the ground truth are shown in Figure 3.1.

Parameter	Description	Value
H_{image}	Height of image from camera	720 pixel
y_c	Height of camera	1.7 m
y_e	Height of user eye	1.28 m
x_0	Nearest Proximity of Camera	3 m
x_1	Furthest Proximity of Camera	50 m
x_{calib}	Distance of calibration plane	1.1 m
L	Length of calibration plan in camera view	1.05 m

Table 3.1. Configuration of Wheelchair System

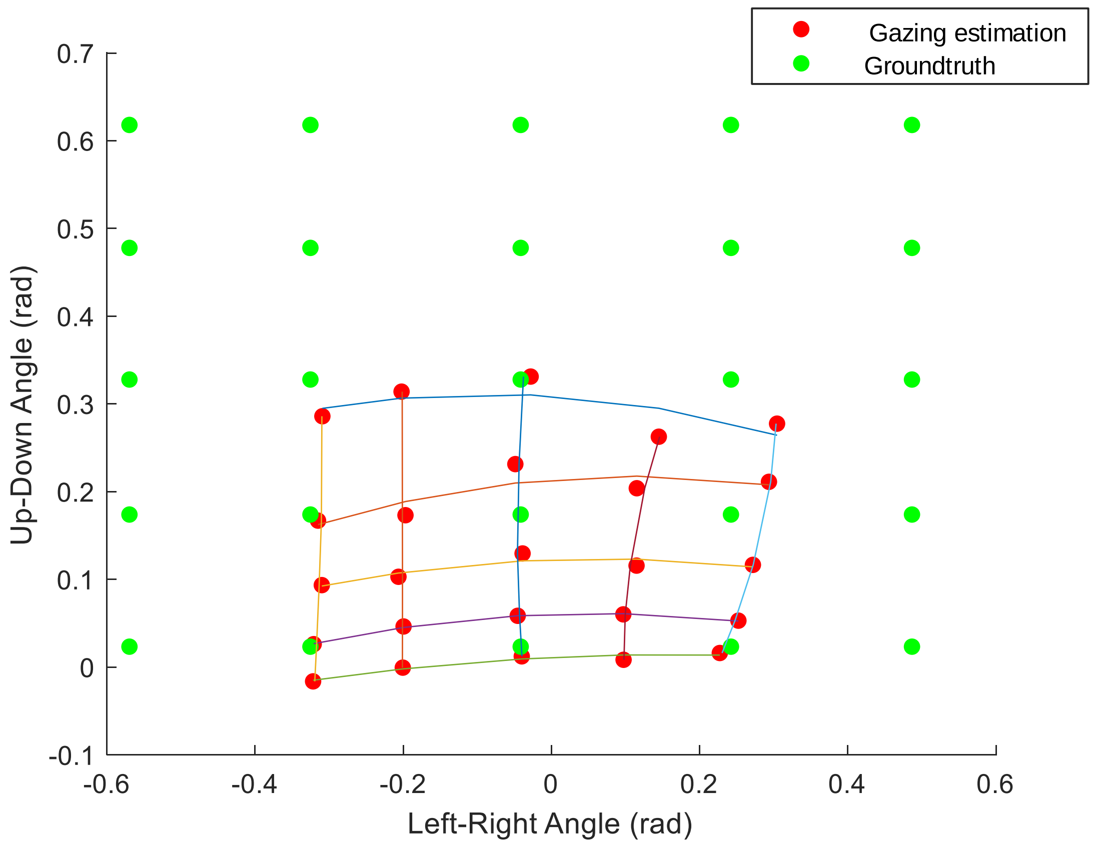


Figure 3.1. Comparison between Gaze Estimation and Ground Truth.

As can be seen in Figure 3.1, the performance of eye gaze estimation in practice is significantly off from the ground truth. This is explained by the large turning angle in head pose which degrades the landmarks detected in the face. Additionally, there is a distortion problem for the sample that has the same up-down angle which leads to curving in the detected image. The curving problem increases when the up-down angle increases. This is because the intersection between the image plane of the eye tracker camera and the user's visual plane is a curve. As a result, it is necessary to correct the eye gaze to the ground truth. The end goal is to map the gaze estimation to the forward camera which needs to take into account the image plane of the camera.

3.2 Calibration to the On-Screen Image Plane

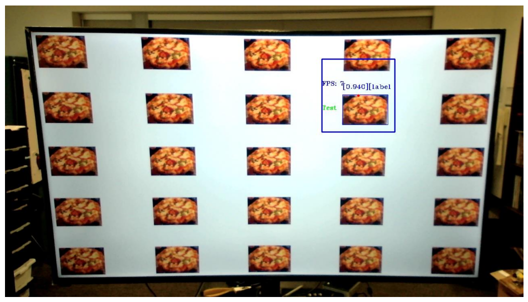
3.2.1 Calibtation procedure

To calibrate the gaze predictions to the object camera in a practical way that could be utilized in the field, the setup in Figure 3.2 is used with the object appearing in the camera instead of relying on ground truth. The forward camera is adjusted to cover the view 3 meters away from the user. The screen will display the gaze target used for calibration so that the object detection algorithm can recognize it and obtain its coordinate in pixel coordinates (x_{pixel} and y_{pixel}). An ROI will slide over the whole image from the forward camera to get all points of interest. First of all, a rule is defined to pick up the right target for calibration. The target detected in the ROI is considered valid for calibration if it satisfies the following criteria:

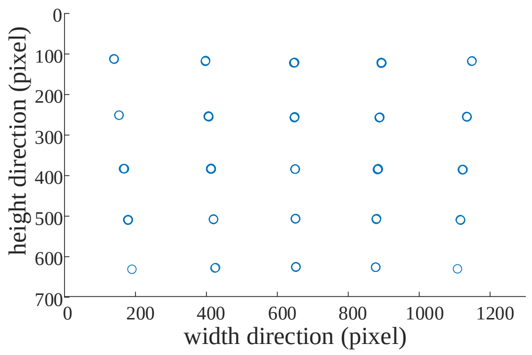
- There is a target object (in this example label 59) detected in the ROI with the bounding box R
- The bounding box R and the ROI must have sufficient margin (4 *pixels*) to assure that the target object is fully inside the ROI.

To increase the performance of the detection algorithm on the target object for calibration, a picture of a pizza in the COCO dataset is chosen here as it yielded high recognition rates across all viewing angles in the image. Furthermore, the size of the ROI is tightened to reduce the number of detections from one specific sample. The calibration process is implemented as follows:

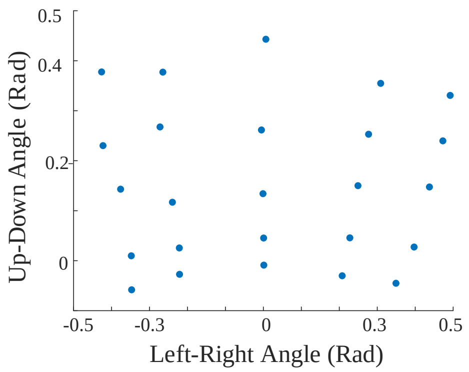
- **Forwarding Camera Calibration.** A ROI will sequentially slide from left to right and from top to bottom in the image with a sliding window of 60 pixels. For each step, the detection algorithm will detect whether a target object is fully inside the ROI and record the center of the object in image coordinates (x_{pixel} and y_{pixel}). There are multiple data points collected for each specific sample



(a) Image from forward camera



(b) Calibration points from the screen



(c) Eye gazing estimation samples

Figure 3.2. Data from Calibration.

(see Figure 3.2.b). After the sliding window detection process is complete, 25 points are kept for calibration algorithm estimation.

- **Eye Gaze Calibration.** The 25 points representing the centers of the calibration targets are shown on the screen. Users will be asked to sequentially look at them from left to right and top to bottom. When they are looking at the point, they were asked to confirm by typing 1 in the computer so that the sample will be recorded. In the end, 25 samples of eye gaze angles are collected (see Figure 3.2.c)

3.2.2 Mapping Algorithm Model

After the calibration process, 25 eye-gaze directions will be mapped to the 25 data samples for the corresponding screen coordinates. In order to estimate the mapping function, the *Gauss-Newton Method* is deployed [46]. Gauss-Newton is an optimization algorithm commonly used for solving least squares problems, which involve minimizing the sum of squared residuals between observed data and a model function. The algorithm iteratively updates the parameters of the model function to find the optimal values that minimize the residuals. At first, a model of the mapping function needs to be established. While an analytic model could be established to capture pure optical distortions related to the camera angles, the mapping function sought here should also address gaze direction estimate distortions which are much harder to predict as they are the result of the feature extraction and estimation approach, and thus the form of the mapping function has here to be determined empirically based on observations in the data. Once the structure of the mapping function has been set, there are several ways to interpret the class from the profile by lightweight methods: optimizing the mean square error on impedance profile [47] and Gaussian and regression filtering on on-machine measurement profile [48].

Let $x_{estimate}$, $y_{estimate}$ denote the estimation of a sample in image coordinates with respect to α and β, which are left-right and up-down angles, respectively. In order to address distortions, and accounting for the difference between the two directions, 2 models need to be established for $x_{estimate}$, $y_{estimate}$ as below:

$$x_{estimate} = f_x(\alpha, \beta) \tag{3.1}$$

$$y_{estimate} = f_y(\alpha, \beta) \tag{3.2}$$

Horizontal model. As can be seen in Figure 3.1 and Figure 3.2, the eye gaze data exhibits a curve when the data has the same hight or the same y_{pixel}

coordinate which has been mentioned in the previous section. Therefore, a second-order polynomial component of α is added into f_x to unwrap the data into a straight line as in Figure 3.2.b. Nevertheless, the more the user turns the head to the side, the smaller the step is, which suggests a *sine* component. The data in angle space also tilted compared to that of pixel space which is indicative of a first-order polynomial component of $\alpha * \beta$. As a result, the model for horizontal mapping chosen here has the form:

$$x_{estimate} = b_{x1} * \alpha^2 + b_{x2} * sin(\alpha) + b_{x3} * \alpha * \beta + b_{x4} * \alpha + b_{x5} \tag{3.3}$$

Vertical model. The curving problem can also be observed here in angle space while there is a straight line in pixel space. That would suggest a second-order polynomial of β in f_y. Secondly, the β angle has increasing step size while the step size in y_{pixel} remains the same. This is due to the translation from the angle to the pixel, and the fact that the eye tracker is looking upward at the user's face, which causes the distortion with the higher β angle. A *cosine* component and second-order polynomial components of α and $\alpha * \beta$ are also added to the function. Finally, the model for vertical mapping is chosen as:

$$y_{estimate} = b_{y1} * \beta^2 + b_{y2} * \beta + b_{y3} * \alpha^2 + b_{y4} * cos(\beta) + b_{y5} * \alpha^2 * \beta^2 + b_{y6} \tag{3.4}$$

3.2.3 Mapping Algorithm Estimation

In Eq. 3.3 and Eq. 3.4, the model of eye-camera mapping has been established. In this section, the details of the *Gauss-Newton Estimation* are provided which is used to determine the parameter vectors $b_x = [b_{x1}, b_{x2}, b_{x3}, b_{x4}, b_{x5}]$ and $b_y = [b_{y1}, b_{y2}, b_{y3}, b_{y4}, b_{y5}, b_{y6}]$. For this, a Non-Linear Least Square problem is defined as below:

Find n unknown parameters b=$(b_1, b_2, b_3, ..., b_n)$ for a none-linear model $y = f(x, b)$ that fits m-observations $(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)$ such that the sum square error is minimum

A residual function R is established as follows:

$$R_i = y_i - f(x_i, b) \tag{3.5}$$

Where R_i is the residual value of sample i, y_i is the value in pixel space that we need to estimate, and b is the parameter vector. The problem now is to minimize the *Sum Square Error* which is stated as $S = \sum_{i=1}^{n} R_i^2$. The gradient of S is given by:

$$\nabla S = 2\sum_{i=1}^{m} R_i \frac{\partial R_i}{\partial b_i} \tag{3.6}$$

To find the optimal value of b_i iteratively, an update procedure of b_i is defined as:

$$b_{i,t+1} = b_{i,t} + \Delta \tag{3.7}$$

At this stage, the b_i need to be adjusted to satisfy Newton's method for minimizing a function S [46]:

$$b_{i,t+1} = b_{i,t} - H^{-1}\nabla S \tag{3.8}$$

where H is the Hessian matrix which is defined as the square matrix of second partial derivatives of a multivariable function and ignoring the second-order derivative terms.

$$H = \begin{bmatrix} \frac{\partial R_1^2}{\partial b_1^2} & \frac{\partial R_1^2}{\partial b_1 \partial b_2} & \cdots & \frac{\partial R_1^2}{\partial b_1 \partial b_n} \\ \frac{\partial R_2^2}{\partial b_2 \partial b_1} & \frac{\partial R_2^2}{\partial b_2^2} & \cdots & \frac{\partial R_2^2}{\partial b_2 \partial b_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial R_n^2}{\partial b_n \partial b_1} & \frac{\partial R_n^2}{\partial b_n \partial b_2} & \cdots & \frac{\partial R_n^2}{\partial b_n^2} \end{bmatrix} \tag{3.9}$$

To solve the derivative, the Jacobian matrix is introduced as the matrix of derivatives of the residual over the parameter vector which results in $H = J * 2J^T J$ and $\nabla S = 2J^T R$

$$J = \begin{bmatrix} \frac{\partial R_1}{\partial b_1} & \frac{\partial R_1}{\partial b_2} & \cdots & \frac{\partial R_1}{\partial b_n} \\ \frac{\partial R_2}{\partial b_1} & \frac{\partial R_2}{\partial b_2} & \cdots & \frac{\partial R_2}{\partial b_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial R_m}{\partial b_1} & \frac{\partial R_m}{\partial b_2} & \cdots & \frac{\partial R_m}{\partial b_n} \end{bmatrix} \tag{3.10}$$

From equations 3.7,3.8, 3.9 and 3.10, an update of b_i is formulated as:

$$\Delta = -(J^T J)^{-1} J^T R \tag{3.11}$$

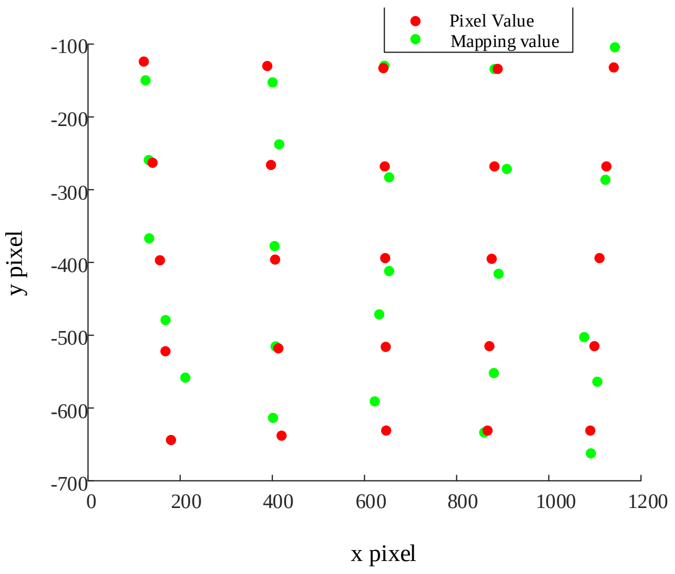


Figure 3.3. Estimation Result.

Put together, a procedure for finding the optimal value of the parameter vector is defined as:

- **Step 1**: Initialize the parameter vector b.

- **Step 2**: Calculate Jacobian Matrix J and Residual R over m observations.
- **Step 3**: Calculate the update step Δ and update the value of parameter vector b.
- **Step 4**: Continue Steps 2 and 3 until the Sum Square Error S is converged or until a specific iteration to avoid an infinite loop.

The estimations for horizontal model f_x and vertical model f_y are implemented separately. The performance of the mapping function is shown in Figure 3.3. Note that this calibration process is done for specific users due to the effect of user height. There is still an error in estimation value which is caused by the uncertainty of eye gaze prediction. However, the overall performance seems good enough for pixel estimation in our application.

3.2.4 Approximate Compensation

In the previous section, eye-camera mapping functions are established to map the eye-gaze input to the camera pixel on the image plane at a specific distance. However, there will be a shift in the vertical dimension (y_{pixel}) when the object appears at a farther distance (see Figure 3.4). This problem is caused by the vertical displacement between the user's eye and the camera which is mentioned in Section 2.3. Specifically, different image planes at different distances will result in different y_{pixel}. Therefore, the estimated ROI is always below the point that the user is looking at in the first few experiments. It is impossible to precisely adjust y_{pixel} without knowing the distance between the user and the context of focus. While this could be addressed by using a depth sensor, we are interested here in keeping the system as simple as possible and thus want to limit the sensors to standard cameras due to their lower cost and superior capabilities in outdoor situations as compared to depth cameras. Thus no precise compensation is possible here. However, an approximate compensation is analyzed based on the camera's range of operation.

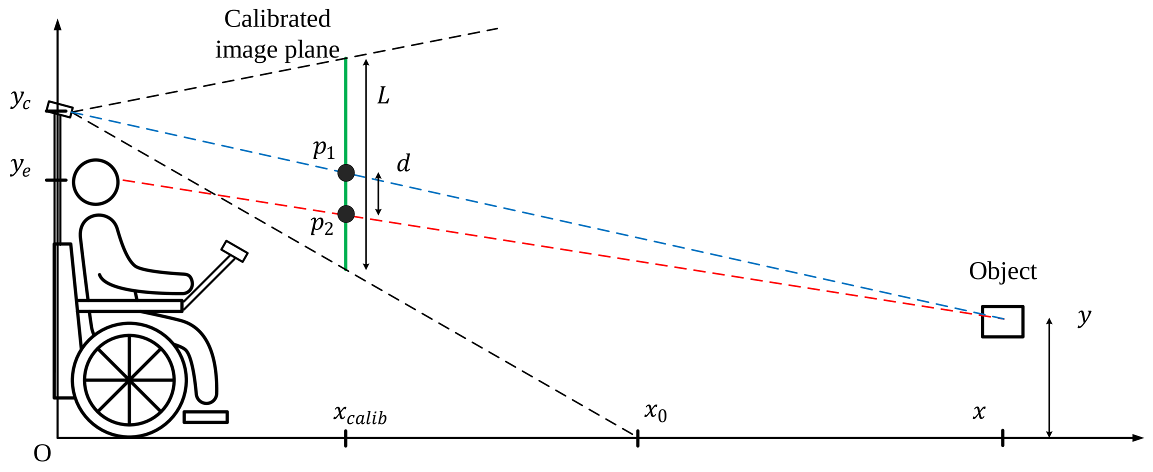


Figure 3.4. Distortion on Image Plane in y_{pixel}.

The object is assumed to appear at distance x from the camera and at height of y from the ground. When the user looks at the object, the mapping algorithm will estimate the ROI at pixel p_2 as the algorithm is calibrated at the image plane at x_{calib} distance. However, the object will actually appear at pixel p_1 in the camera image. Therefore, there will always be an error of $\Delta p = p_2 - p_1$ in y_{pixel} estimation. Depending on the location of the object, the distance between p_1 and p_2 would be:

$$\begin{aligned} d &= (y_c - y) \cdot \frac{x - x_{calib}}{x} - (y_e - y) \cdot \frac{x - x_{calib}}{x} \\ &= (y_c - y_e) \cdot \frac{x - x_{calib}}{x} \end{aligned} \tag{3.12}$$

where y_c and y_e are the vertical position of the camera and the users eyes, respectively. As a result, the displacement in pixel would be:

$$\Delta p = d \cdot \frac{H_{image}}{L} \tag{3.13}$$

where H_{image} is the image height and L is the vertical dimension of the calibrated image plane. The pixel displacement is a function of x which is the distance of the object. The camera can theoretically see the object at a distance from x_0 to ∞ , or x_0 to x_1 more practically depending on the size of the object, which limits the pixel displacement as:

$$(y_c - y_e) \cdot \frac{(x_0 - x_{calib}) \cdot H_{image}}{x_0 \cdot L} \leq \Delta p < (y_c - y_e) \cdot \frac{(x_1 - x_{calib}) \cdot H_{image}}{x_1 \cdot L} \tag{3.14}$$

The configuration for the wheelchair setup is shown in Table 3.1. As a result, Δp

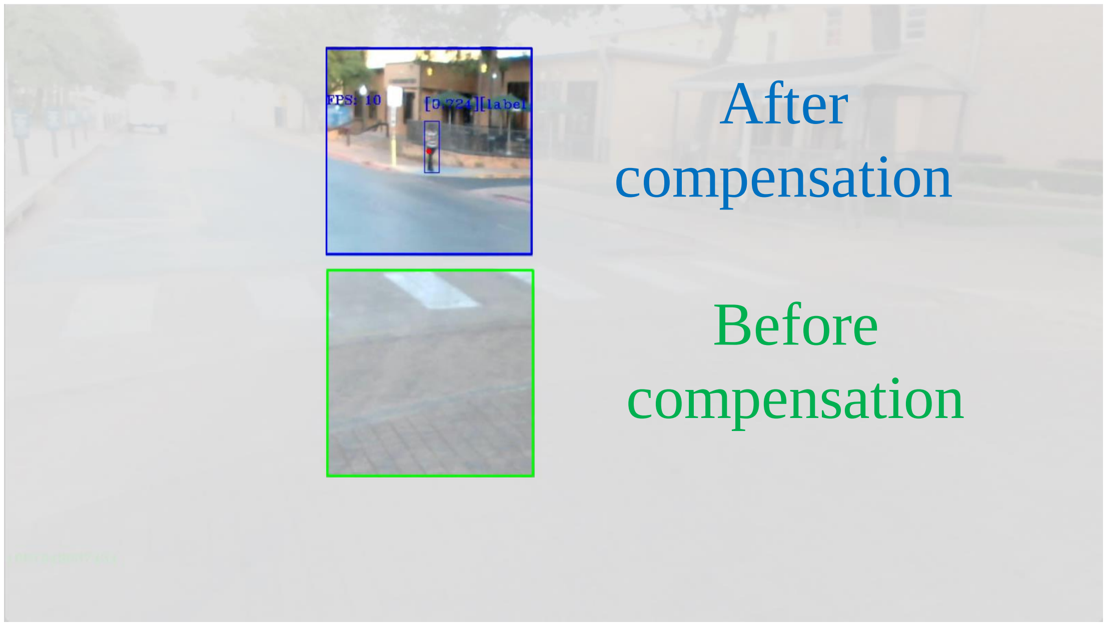


Figure 3.5. Effect of compensation.

will vary in the range of $(217, 335)$ pixels. The variation is 118 which is relatively small compared to the ROI size of 240x240 pixel. Thus a compensation of $\Delta p = 217 + 118/3 \approx 257$ pixel is added to the y_{pixel} estimation model to shift up the ROI. The compensation value will be one-third of the range due to the fact that the closer the distance, the bigger the object is captured in the image frame which needs to be precisely compensated in the model to cover the whole object. Another reason is that the compensation value will shift up the upper edge of the ROI more than 360 pixels which already covers the area with the distance of ∞. The effect of compensation is indicated in Figure 3.5 where the ROI can capture the object and detect it.

CHAPTER 4

NAVIGATION INTENTION DETECTION

4.1 Motivation

Eye gaze for navigation is one of the most intuitive and natural approaches and has been investigated in previous chapters. Navigation activities are categorized into 2 classes: navigation intention and navigation-related attention.

- **Navigation intention** is defined as a short-term intention of humans to move toward a particular place with a specific condition or environment [49]. This means that the person looks at the locations such as the end of the hall, the entrance of the building, or potentially a person because they want to move to that place. The navigation intention is short-term because the intention can be changed due to the change of the environment such as when the way is blocked by a wall or one person stepping into the path, causing the need to change the navigation path temporarily.
- **Navigation-related attention** does not refer to the activities of moving to a particular place, but something related to the navigation intention. For example, humans look around to recognize the environment near the navigation path without intending to move toward that place. However, the location they are looking at still has a relation to the path and can thus allow inferences about the place an individual wants to navigate to. For example, if a person is looking up and down a street from the sidewalk, this might ndicate that the navigation objective is to cross the street. Navigation-related attention can be the activities of looking at the ground to see if it is possible to reach the end

goal, if there is anything coming from the side of the street, or if a person steps into the original path. If can also be a situation where the attention is on a better path to reach the navigation place instead of continue moving on the current route.

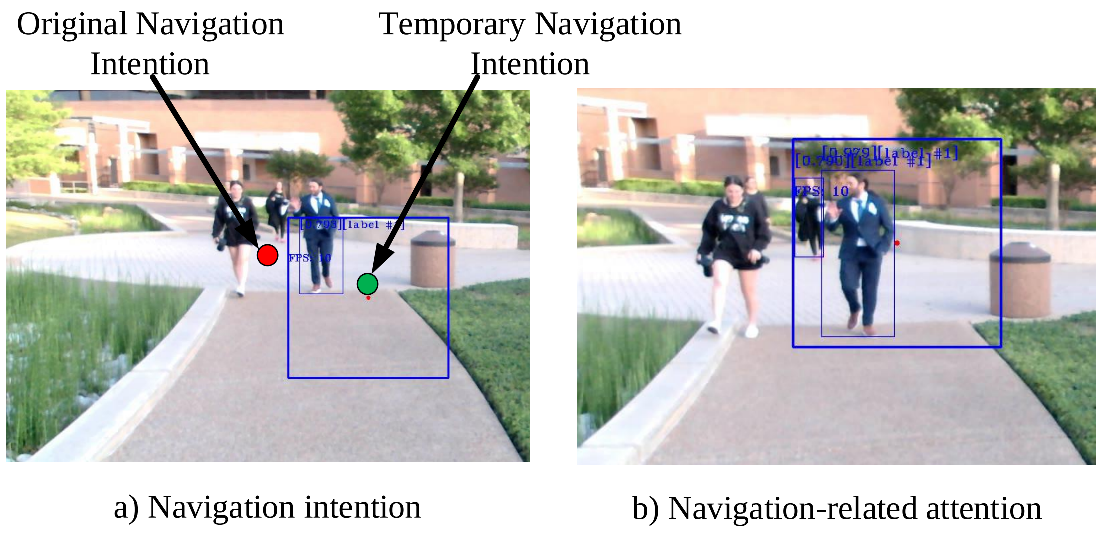

Original Navigation Intention
Temporary Navigation Intention
a) Navigation intention
b) Navigation-related attention

Figure 4.1. Examples of Navigation Activities.

Figure 4.1 shows two examples of these intention categories. The two behaviors vary in terms of their gazing activities and movement objectives. Regarding gazing behavior, humans intermittently look at their intended destination, irrespective of what's happening in their surroundings. The gaze direction should also match the head orientation during movement. Conversely, during movement, humans only glance once to recognize the environment for navigation-related attention. The eye-gazing direction for navigation-related attention is typically towards the sides or ground, which exhibit distinct patterns. Regarding movement, humans will move toward their intended navigation direction rather than the navigation-related direction.

Humans also have more distinct gazing behaviors which do not relate to navigation. As we are interested in navigation here, we group all of these into one class. For example, during movement, humans can look at one person to try to recognize him or her or look at the advertisement flyer on the wall. This type of activity is classified as **Non-Navigation Attention**. The probability of non-navigation attention will increase if there is an object in the focus ROI of the user. In addition to that, the user will only focus on the non-navigation object for a certain amount of time. Examples of non-navigation attention scenarios are shown in Figure 4.2. For the

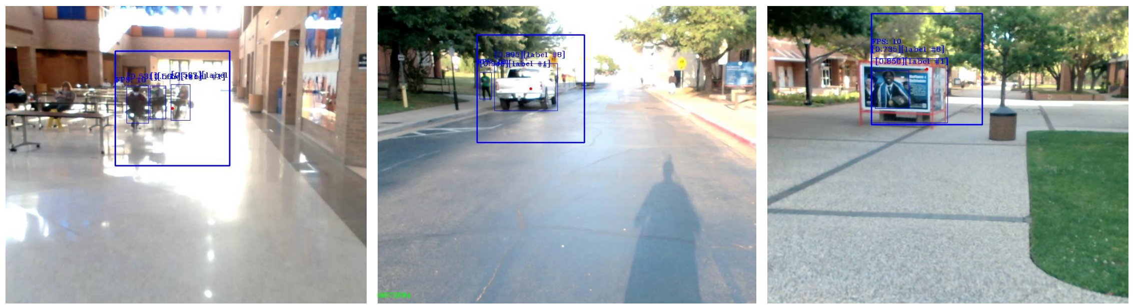


Figure 4.2. Examples of Non-Navigation Activities.

HCI purpose, navigation intention is the most important and needs to be classified correctly so that the system can control the wheelchair accordingly while identification of the other attention types is not as important. Therefore, it is important to embed the navigation intention detection model into the system. In this section, a neural network is developed to classify the three aforementioned classes.

4.2 Experimental Setting

First, a series of experiments are conducted to collect the data of user gaze as well as corresponding object detection videos. A joystick is used to control the wheelchair while moving around the campus (see Figure 4.3). Two videos are captured from both cameras with the UNIX timestamp for synchronization purposes. In addition, a log file is recorded, including UNIX time stamp, gaze angles, head angles, ROI coordinates, object coordinates, and object size.

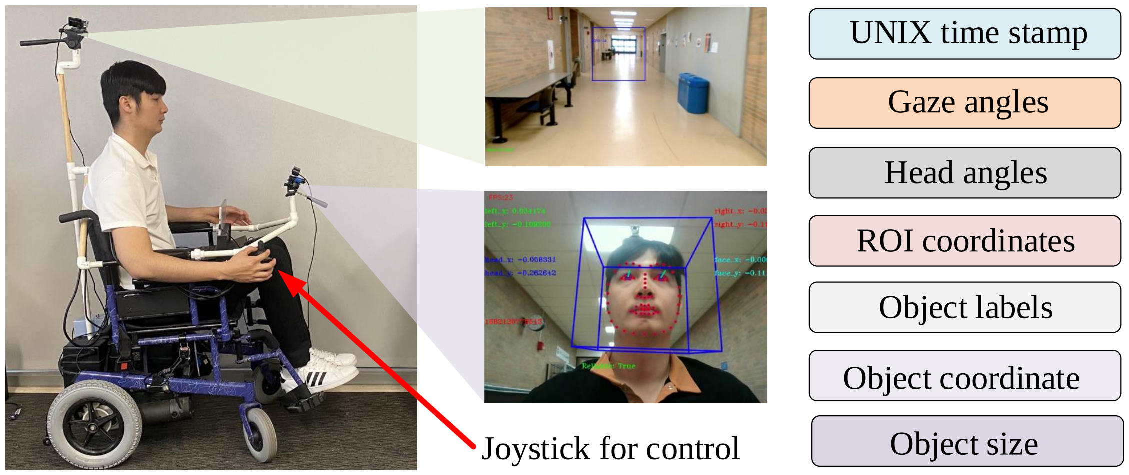


Figure 4.3. Experiment Setup.

For better generalization, each experiment is recorded for three to five minutes at different places on campus including indoors and outdoors, on the sidewalk or on the road, from indoors to outdoors, and vice versa. The data with two videos are saved on the Surface Pro 4 for later processing. Figure 4.4 shows examples of the large diversity of scenarios included in the data collection experiments.

4.3 Data Preparation

After being collected, the data is then labeled based on the synchronized video. The video is decompressed into multiple image frames. A data engineer looks at each image frame and labels the sample whether it is navigation intention, navigation-related attention, or non-navigation attention. Those three classes are tokenized by:

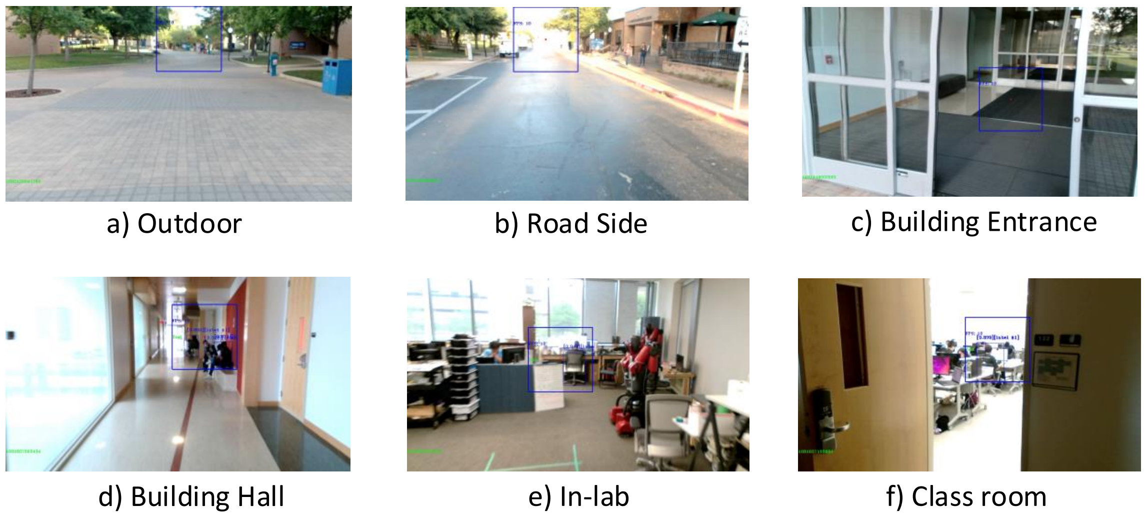


Figure 4.4. Experiment Settings.

- **0** as Navigation Intention.
- **1** as Navigation-Related Attention.
- **2** as Non-Navigation Attention.

One problem in the data is that the number of objects detected in the ROI varies from zero to six, which can lead to inconsistency in the data format. In the analysis of the data, less than two objects are detected in the ROI most of the time. Therefore, only the two biggest objects detected are kept for further processing. If there is a Null value in the data sample due to fewer than two objects being present, -1 will be added to avoid the computational complication of variable-length feature vectors. To avoid overfitting, one set of data which is recorded by one experiment is kept back for testing; the remaining data is used for training which makes the training and testing data entirely independent.

4.4 Navigation Attention Model

The data has 16 features, including angles, pixels, and object labels. A neural network is proposed to classify the data into the three classes mentioned in the previous section. A dense neural network is proposed that includes 5 hidden layers with down-scale node distribution as shown in Figure 4.5. The five hidden layers use Rectified Linear Unit (reLU) activation functions to reduce the computational complexity, while the output layer has softmax activation so that it gives us the probability of each class. The overall architecture of the network is shown in Figure 4.5.

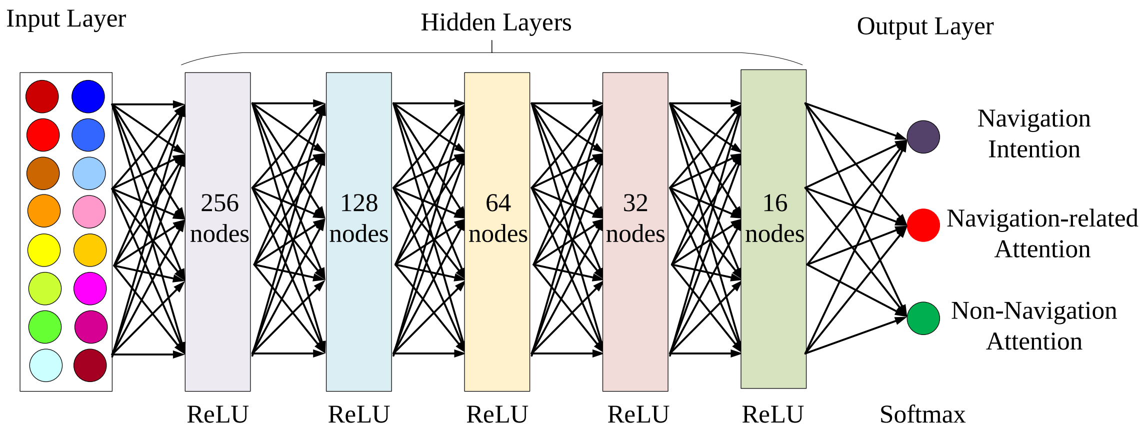


Figure 4.5. Architecture of Navigation Intention Model.

This network architecture was determined experimentally by testing a range of different layer and node combinations, incrementally refining the architecture to achieve the highest accuracy without overfitting. Table 4.1 shows a set of the network architectures evaluated with the corresponding accuracy results.

The model is trained for 500 epochs with a SparseCategoricalCrossentropy loss function to optimize the cross entropy between the classes. A spare categorical is suitable for classes tokenized by integers. The metric to evaluate the performance of

Number of hidden layers	1st layer	2nd layer	3rd layer	4th layer	5th layer	6th layer	Performance
3	70	70	70	N/A	N/A	N/A	93.79
4	70	70	70	70	N/A	N/A	97.04
4	128	64	32	16	N/A	N/A	95.6
4	256	128	64	32	N/A	N/A	96.62
4	32	64	128	256	N/A	N/A	95.36
5	70	70	70	70	70	N/A	96.44
5	256	128	64	32	16	N/A	97.83
5	16	32	64	128	256	N/A	93.97
6	70	70	70	70	70	70	88.36

Table 4.1. Network Configurations and Performances

the model would ultimately be sparse categorical accuracy so that it will be consistent with the loss function.

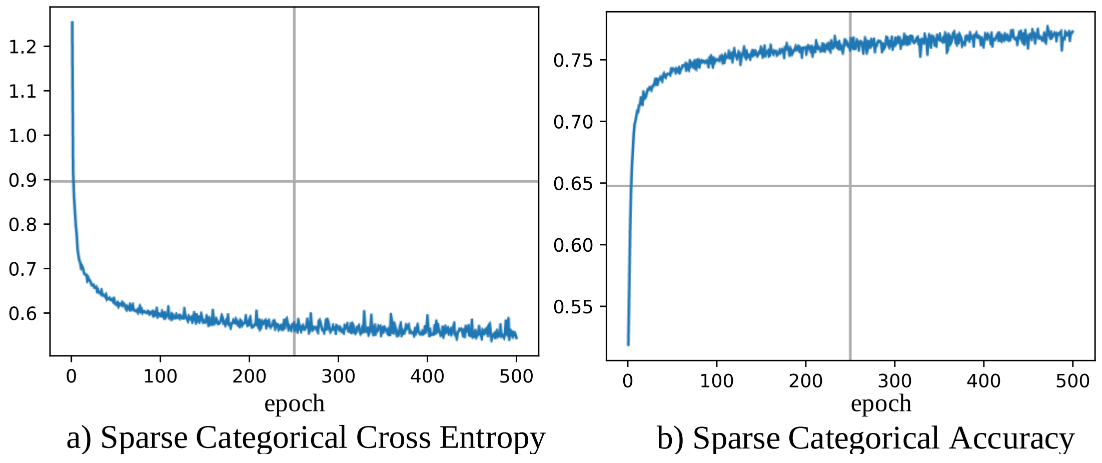


Figure 4.6. The Convergence of the Model During the Training Process.

As can be seen in Figure 4.6, the Sparse Categorical Cross Entropy is reduced to around 0.55 n this model while the Sparse Categorical Accuracy converges to around 0.79. This demonstrates that the data used for training has distinct representations for each class. Multiple hyperparameter turnings are also implemented with different values for the number of epochs, the number of hidden layers, different activation functions, and the number of nodes in one layer. The optimal values have been reported in this thesis with the configuration in Table 4.1. With that configuration, each step of requires around 1 ms of training time.

4.5 Performance

To determine the generalization ability of the learned model, the pre-trained network is then evaluated on the test data representing one specific experiment which is not included in the training data. The test data is being processed in the same way as the training data. Each inference step requires around 20ms, which is fast enough to assure real-time operation of the system. The confusion matrix is shown in Figure 4.7.

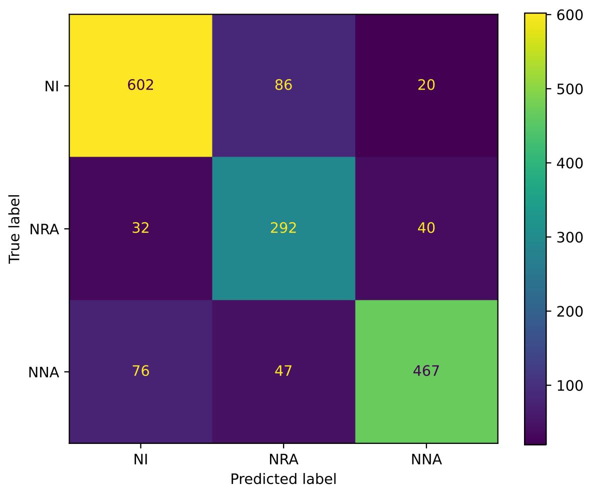


Figure 4.7. The Confusion Matrix.

In terms of classification accuracy, the model can predict correctly 81.89% of the time during the one test. In addition, the most critical case is the false positive sample when the system predicts the Navigation Intention. This accounts for 7.08% of all predictions. While this might be an acceptable performance for the first attempt at building the context-aware gaze-based interface with navigation intention detection, it might be too high for a practical system and ways to further reduce this were investigated.

4.6 Gaze Behaviour

In the previous section, only a single sample point is used which lacks information on various aspects of gazing behavior. In particular, it makes it impossible for the model to consider the length of a gaze and the dynamics of the change in gaze direction, which can contain important information regarding user intention. The model might need to see the change in gaze angle such as quickly gazing to the side or continuously focusing in a specific direction. A new way for the learning system to look at the data needs to be formed in order to improve the performance of the model and to allow it to consider gaze dynamics. A data concatenate schema is proposed as illustrated in Figure4.8 to represent the relationship in the time series of the gazing activities. The data for the prediction of label i will be the concatenated sequence of data sample i and the four prior samples. The first four samples of each experiment will be removed because there are not sufficient prior samples. However, this should not be a problem since this corresponds only to the first half second, a time in whihc likely no wheelchair movement would have happened yet. With this scheme, the program can assure the real-time property that is necessary for real-world usage and include information on gazing behavior.

With the additional information on gazing behavior, the model can predict a much better result with 97.2% of accuracy (see Figure 4.9)

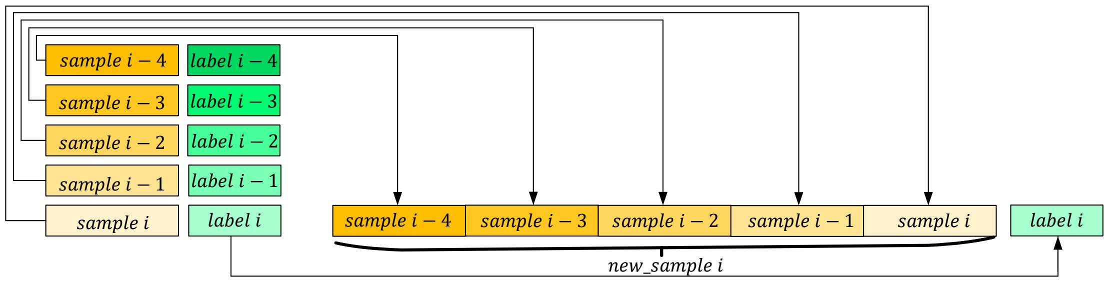


Figure 4.8. Data Concatenate Scheme.

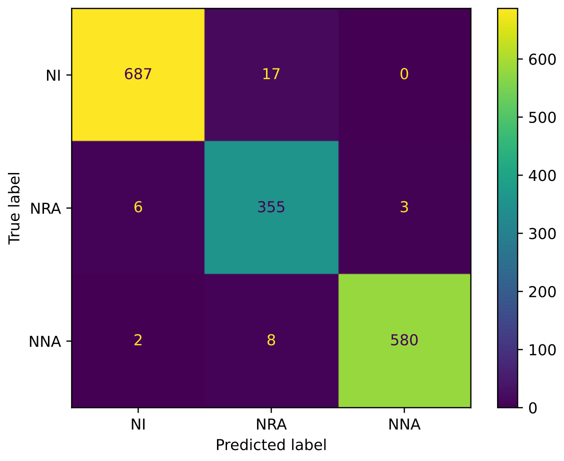


Figure 4.9. The Confusion Matrix Using the New Data Representation.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this thesis, a complete system for a context-aware gaze-based interface has been built using an eye gaze estimator and an object detector connected through ROS. The system can reliably detect the user's face, estimate head pose and eye gaze angles and then capture the region of focus to analyze the navigation behaviors. A Gauss-Newton Estimation approach has been employed to build the mapping functions from eye gaze to the region of focus in the image coordinate system. Additionally, a multi-process ROS-based C++ program has been built to assure the ability of real-time operation for two image processing tasks in constrained computing resources. A PVC-pipe/wooden prototype has been integrated into the electric wheelchair for the in-the-wild setting evaluation. The system has been used to conduct a series of experiments for data collection, calibration, system training, and evalaution. A Navigation Intention model has been developed using a dense neural network that can accurately predict the navigation behaviors of users based on a short time series of gaze and object detections with a confidence of more than 97%.

5.2 Future Work

While the system developed shows promise, there is still a lot of room for improvement and extensions to further increase the performance of the system, including:

- **Re-train the facial landmark model**. As the wheelchair is a personalized device, it is feasible to re-train the facial landmark model so that it will be fitted to the individual user's face and localizes the landmarks more accurately. As a result, the performance of the algorithms for eye-gaze estimation could be improved.
- **Re-train the object detection model**. The current system uses a model trained on the COCO dataset which is largely used to evaluate the performance of object detection models. However, there are various objects which are critical for navigation intention such as roads, pedestrian crossings, a tree, and so on that are not represented to a sufficient degree in this data set. The more relevant objects the system can detect, the richer the information captured for the navigation intention detection model which could improve the performance.
- **Improve the model to navigation intention localization**. The end goal of the Context-Aware Gaze-based Interface is to localize which direction the user wants to go so that the wheelchair can move accordingly. This can be implemented by adding the destination location (in pixels) to the data labeled and reconstructing the architecture of the neural network so that it has the ability to localize the intention direction in addition to the classification problem.

REFERENCES

[1] (2023) Oto bock xeno stnad up wheelchair. [Online]. Available: https://armarhealthcare.com/collections/ottobock/products/xeno

[2] M. R. Walter, S. Hemachandra, B. Homberg, S. Tellex, and S. Teller, “A framework for learning semantic maps from grounded natural language descriptions,” *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1167–1190, 2014.

[3] A. Ghorbel, N. B. Amor, and M. Jallouli, “A survey on different human-machine interactions used for controlling an electric wheelchair,” *Procedia Computer Science*, vol. 159, pp. 398–407, 2019.

[4] Q. X. Nguyen and S. Jo, “Electric wheelchair control using head pose free eye-gaze tracker,” *Electronics Letters*, vol. 48, no. 13, pp. 750–752, 2012.

[5] S. Desai, S. Mantha, and V. Phalle, “Advances in smart wheelchair technology,” in *2017 International Conference on Nascent Technologies in Engineering (ICNTE).* IEEE, 2017, pp. 1–7.

[6] R. C. Simpson, D. Poirot, and F. Baxter, “The hephaestus smart wheelchair system,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 10, no. 2, pp. 118–122, 2002.

[7] R. C. Simpson, “Smart wheelchairs: A literature review,” *Journal of rehabilitation research and development*, vol. 42, no. 4, p. 423, 2005.

[8] T. Rofer, C. Mandel, and T. Laue, “Controlling an automated wheelchair via joystick/head-joystick supported by smart driving assistance,” in *2009 IEEE international conference on rehabilitation robotics.* IEEE, 2009, pp. 743–748.

[9] J. Kim, H. Park, J. Bruce, E. Sutton, D. Rowles, D. Pucci, J. Holbrook, J. Minocha, B. Nardone, D. West, *et al.*, "The tongue enables computer and wheelchair control for people with spinal cord injury," *Science translational medicine*, vol. 5, no. 213, pp. 213ra166–213ra166, 2013.

[10] A. Kaur, "Wheelchair control for disabled patients using emg/eog based human machine interface: a review," *Journal of medical engineering & technology*, vol. 45, no. 1, pp. 61–74, 2021.

[11] (2023) Munevo. [Online]. Available: https://us.munevo.com/

[12] M. Al-Rousan and K. Assaleh, "A wavelet-and neural network-based voice system for a smart wheelchair control," *Journal of the Franklin Institute*, vol. 348, no. 1, pp. 90–100, 2011.

[13] R. C. Simpson and S. P. Levine, "Adaptive shared control of a smart wheelchair operated by voice control," in *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems. Innovative Robotics for Real-World Applications. IROS'97*, vol. 2. IEEE, 1997, pp. 622–626.

[14] A. A. Abed, "Design of voice controlled smart wheelchair," *International Journal of Computer Applications*, vol. 131, no. 1, pp. 32–38, 2015.

[15] L. F. Sanchez, H. Abaunza, and P. Castillo, "Safe navigation control for a quadcopter using user's arm commands," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2017, pp. 981–988.

[16] R. L. Queiroz, I. B. de Azeredo Coutinho, P. M. V. Lima, F. F. Sampaio, and G. B. Xexéo, "Playing with robots using your brain," in *2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE, 2018, pp. 197–1977.

[17] V. Holovenskiy, T. Shmelova, Y. M. Shmelov, O. Tymochko, S. Boiko, A. Khebda, L. Chyzhova, M. Kirukhina, *et al.*, "Unmanned aerial vehicles. perspectives. management. power supply: Multi-authored monograph," 2019.

[18] M. A. Eid, N. Giakoumidis, and A. El Saddik, "A novel eye-gaze-controlled wheelchair system for navigating unknown environments: case study with a person with als," *IEEE Access*, vol. 4, pp. 558–573, 2016.

[19] M. Dahmani, M. E. Chowdhury, A. Khandakar, T. Rahman, K. Al-Jayyousi, A. Hefny, and S. Kiranyaz, "An intelligent and low-cost eye-tracking system for motorized wheelchair control," *Sensors*, vol. 20, no. 14, p. 3936, 2020.

[20] S. Mahmud, X. Lin, J.-H. Kim, H. Iqbal, M. Rahat-Uz-Zaman, S. Reza, and M. A. Rahman, "A multi-modal human machine interface for controlling a smart wheelchair," in *2019 IEEE 7th Conference on Systems, Process and Control (ICSPC)*. IEEE, 2019, pp. 10–13.

[21] Z. Alibhai, T. Burreson, M. Stiller, I. Ahmad, M. Huber, and A. Clark, "A human-computer interface for smart wheelchair control using forearm emg signals," in *2020 3rd International Conference on Data Intelligence and Security (ICDIS)*. IEEE, 2020, pp. 34–39.

[22] (2023) The eye tribe. [Online]. Available: https://github.com/EyeTribe/documentation

[23] (2023) Tobii pro glass 3. [Online]. Available: https://www.tobii.com/products/eye-trackers/wearables/tobii-pro-glasses-3

[24] (2023) Linux kernel for microsoft surface series. [Online]. Available: https://github.com/linux-surface/linux-surface

[25] T. Dang, K. Nguyen, and M. Huber, "Perfc: An efficient 2d and 3d perception software-hardware framework for mobile cobot," in *The International FLAIRS Conference Proceedings*, vol. 36, 2023.

[26] T. T. Dang, J. H. Kim, D. D. Nguyen, and J. W. Jeon, "A Gateway for Multi-device Communication between Mechatrolink-III and RS-485," in *12th International Conference on Control, Automation and Systems.* IEEE, 2012, pp. 294–299.

[27] A. Zadeh, Y. Chong Lim, T. Baltrusaitis, and L.-P. Morency, "Convolutional experts constrained local model for 3d facial landmark detection," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 2519–2528.

[28] T. Baltrusaitis, P. Robinson, and L.-P. Morency, "Constrained local neural fields for robust facial landmark detection in the wild," in *Proceedings of the IEEE international conference on computer vision workshops*, 2013, pp. 354–361.

[29] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[30] J. M. Saragih, S. Lucey, and J. F. Cohn, "Deformable model fitting by regularized landmark mean-shift," *International journal of computer vision*, vol. 91, pp. 200–215, 2011.

[31] J. A. Hesch and S. I. Roumeliotis, "A direct least-squares (dls) method for pnp," in *2011 International Conference on Computer Vision.* IEEE, 2011, pp. 383–390.

[32] F. M. Mirzaei and S. I. Roumeliotis, "Globally optimal pose estimation from line correspondences," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 5581–5588.

[33] E. Wood, T. Baltrusaitis, X. Zhang, Y. Sugano, P. Robinson, and A. Bulling, "Rendering of eyes for eye-shape registration and gaze estimation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3756–3764.

[34] T. Baltrusaitis, A. Zadeh, Y. C. Lim, and L.-P. Morency, "Openface 2.0: Facial behavior analysis toolkit," in *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*. IEEE, 2018, pp. 59–66.

[35] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[36] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[37] ——, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[38] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[39] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[40] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[41] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[42] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[43] (2023) Openvino. [Online]. Available: https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html

[44] (2023) Ros openvino. [Online]. Available: https://github.com/intel/ros_openvino_toolkit

[45] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Appearance-based gaze estimation in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4511–4520.

[46] Å. Björck, *Numerical methods for least squares problems.* SIAM, 1996.

[47] T. Dang, T. Tran, K. Nguyen, T. Pham, N. Pham, T. Vu, and P. Nguyen, "ioTree: a battery-free wearable system with biocompatible sensors for continuous tree health monitoring," in *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, 2022, pp. 769–771.

[48] F. Chen, S. Yin, H. Huang, H. Ohmori, Y. Wang, Y. Fan, and Y. Zhu, "Profile error compensation in ultra-precision grinding of aspheric surfaces with on-machine measurement," *International Journal of Machine Tools and Manufacture*, vol. 50, no. 5, pp. 480–486, 2010.

[49] S. Thompson, T. Horiuchi, and S. Kagami, "A probabilistic model of human motion and navigation intent for mobile robot path planning," in *2009 4th International Conference on Autonomous Robots and Agents.* IEEE, 2009, pp. 663–668.