HARDWARE AND SOFTWARE SYSTEMS FOR CONTROL OF ASSISTIVE
ROBOTIC DEVICES USING POINT-OF-GAZE ESTIMATION

by

CHRISTOPHER DALE McMURROUGH

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

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2013
In loving memory of Glenna Jo Bloomquist.
ACKNOWLEDGEMENTS

I would like to first and foremost thank my advisor, Professor Fillia Makedon, for her support and encouragement. Her mentoring has been extremely valuable to me, and is greatly appreciated. I would also like to thank my committee members Professor Vassilis Athitsos, Professor Gian-Luca Mariottini, and Professor Frank Lewis for their guidance and insight.

Additionally, I would like to thank my past and present colleagues at the Heracleia Human Centered Computing Laboratory and also at The University of Texas at Arlington Research Institute. Working with each of you has been a great pleasure.

November 20, 2013
ABSTRACT

HARDWARE AND SOFTWARE SYSTEMS FOR CONTROL OF ASSISTIVE ROBOTIC DEVICES USING POINT-OF-GAZE ESTIMATION

CHRISTOPHER DALE MCMURROUGH, Ph.D.

The University of Texas at Arlington, 2013

Supervising Professor: Fillia Makedon

Eye gaze based interaction has many useful applications in human-machine interfaces, assistive technologies, and multimodal systems. Traditional input methods, such as the keyboard and mouse, are not practical in many situations and can be ineffective for some users with physical impairments. Knowledge of the user point of gaze (PoG) can be a powerful data modality in intelligent systems by facilitating intuitive control, perception of user intent, and enhanced interactive experiences.

This research aims to advance the use of non-traditional, multimodal interfaces in assistive robotic devices for the benefit of users with severe physical disabilities. The data modalities which are of particular interest in this work are perception of the environment using 3D scanning and computer vision, estimation of the user point of gaze using video occulography, and perception of user intent during interaction with objects and locations of interest. A novel, mobile headset design is presented that provides both monocular and binocular pupil tracking together with a 3D reconstruction of the user’s field of view. Computational methods for obtaining a true 3D gaze
vector and final PoG are also discussed, along with a demonstration of practical use.
# TABLE OF CONTENTS

ACKNOWLEDGEMENTS .............................................. iv

ABSTRACT ......................................................... v

LIST OF ILLUSTRATIONS ........................................ x

LIST OF TABLES .................................................. xii

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Related Work</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Hypothesis</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Overview</td>
<td>4</td>
</tr>
<tr>
<td>2. TOWARDS MOBILE 3D POINT-OF-GAZE ESTIMATION</td>
<td>5</td>
</tr>
<tr>
<td>2.1 A Dataset for Head-Mounted Eye tracker Evaluation and Benchmarking</td>
<td>6</td>
</tr>
<tr>
<td>2.1.1 Existing Datasets</td>
<td>7</td>
</tr>
<tr>
<td>2.1.2 Data Collection Methodology</td>
<td>8</td>
</tr>
<tr>
<td>2.1.3 Coordinate Systems</td>
<td>13</td>
</tr>
<tr>
<td>2.1.4 Dataset Structure and Format</td>
<td>14</td>
</tr>
<tr>
<td>2.1.5 Results</td>
<td>16</td>
</tr>
<tr>
<td>2.2 3D Head Pose Tracking Relative to a Fixed Frame</td>
<td>19</td>
</tr>
<tr>
<td>2.2.1 Tracking Hardware</td>
<td>20</td>
</tr>
<tr>
<td>2.2.2 Coordinate Systems</td>
<td>23</td>
</tr>
<tr>
<td>2.2.3 Experimental Setup</td>
<td>24</td>
</tr>
<tr>
<td>2.2.4 Results</td>
<td>25</td>
</tr>
<tr>
<td>2.3 Pseudo 3D Mobile PoG Tracking</td>
<td>27</td>
</tr>
</tbody>
</table>
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Data collection participant during a recording session</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>Example smooth pursuit target pattern shown to participant</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>Representation of the 3D space perceived by motion capture</td>
<td>11</td>
</tr>
<tr>
<td>2.4</td>
<td>Motion capture system setup at Heracleia Laboratory</td>
<td>12</td>
</tr>
<tr>
<td>2.5</td>
<td>Eye tracker equipped with reflective tracking markers.</td>
<td>14</td>
</tr>
<tr>
<td>2.6</td>
<td>Coordinate systems used in the dataset</td>
<td>15</td>
</tr>
<tr>
<td>2.7</td>
<td>An example video frame contained in the dataset</td>
<td>16</td>
</tr>
<tr>
<td>2.8</td>
<td>Headset with mounted tracking hardware</td>
<td>20</td>
</tr>
<tr>
<td>2.9</td>
<td>Eye camera view</td>
<td>22</td>
</tr>
<tr>
<td>2.10</td>
<td>Coordinate systems used by the head pose tracker</td>
<td>25</td>
</tr>
<tr>
<td>2.11</td>
<td>Comparison of stereo tracking and motion capture</td>
<td>26</td>
</tr>
<tr>
<td>2.12</td>
<td>Headset with eye and scene cameras</td>
<td>28</td>
</tr>
<tr>
<td>2.13</td>
<td>Scene color and depth camera information</td>
<td>30</td>
</tr>
<tr>
<td>2.14</td>
<td>Mapping of gaze vector to scene</td>
<td>32</td>
</tr>
<tr>
<td>2.15</td>
<td>Reference frames used by the headset</td>
<td>34</td>
</tr>
<tr>
<td>2.16</td>
<td>Digital CAD design of 3D printed parts</td>
<td>39</td>
</tr>
<tr>
<td>2.17</td>
<td>Assembled 3D gaze tracking headset</td>
<td>40</td>
</tr>
<tr>
<td>2.18</td>
<td>Images collected during the hand-eye and gaze vector calibration</td>
<td>42</td>
</tr>
<tr>
<td>2.19</td>
<td>Reconstruction of coordinate frames and calibration gaze vectors</td>
<td>43</td>
</tr>
<tr>
<td>3.1</td>
<td>Software architecture of the iDog system</td>
<td>49</td>
</tr>
<tr>
<td>3.2</td>
<td>Image processing pipeline</td>
<td>50</td>
</tr>
</tbody>
</table>
3.3 Comparison of original and equalized histograms .................. 52
3.4 Representation of a straight line in polar coordinates ............... 53
3.5 Hough space of an example image frame .............................. 54
3.6 Single frame example of vanishing point detection ................... 56
3.7 Multiple tabs of the iDog System Controller Application ............ 60
3.8 Vision and ranging data perceived by the iDog system ............... 61
3.9 Holonomic manipulation platform with arm ........................... 62
3.10 Layout of wheel module with key measurements ..................... 63
3.11 Completed wheel module after fabrication .......................... 64
3.12 Platform suspension and motor arrangement ........................ 65
3.13 Assembled mechanical platform ..................................... 66
3.14 ROS software control architecture .................................. 67
3.15 Assembled platform .................................................... 68
3.16 System human interface hardware ................................... 70
3.17 Intelligent wheelchair platform with user ............................ 73
3.18 System software architecture and communication flow .............. 74
3.19 Example calculation of BCI decision boundary ...................... 74
4.1 Visual salience heatmap in 2D [1] ..................................... 76
4.2 Comparison of 3D scene with overlayed visual salience heat map .... 80
4.3 Point cloud processing steps .......................................... 83
4.4 SURF keypoint matches ............................................. 84
4.5 Collection of objects in experimental scene .......................... 87
4.6 PR2 robot manipulating tabletop objects .............................. 90
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Example metadata for the video frame shown in Figure 2.7.</td>
<td>17</td>
</tr>
<tr>
<td>2.2 Experimental results of the 3 tracking methods.</td>
<td>19</td>
</tr>
<tr>
<td>2.3 RMSE of position and orientation components.</td>
<td>27</td>
</tr>
<tr>
<td>4.1 Object identification results</td>
<td>88</td>
</tr>
<tr>
<td>4.2 Table of average runtimes</td>
<td>89</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

Eye gaze based interaction has many useful applications in human-machine interfaces, assistive technologies, and multimodal systems. Traditional input methods, such as the keyboard and mouse, are not practical in many situations and can be ineffective for some users with physical impairments. Knowledge of the user point of gaze (PoG) can be a powerful data modality in intelligent systems by facilitating intuitive control, perception of user intent, and enhanced interactive experiences.

Gaze tracking devices have proven to be extremely beneficial to impaired users. In the case study presented in [2], 16 Amyotrophic Lateral Sclerosis (ALS) patients with severe motor impairments (loss of mobility, unable to speak, etc.) were introduced to eye tracking devices during a 1-2 week period. The patients were assessed by a psychologist during an initial meeting in order to evaluate their general quality of life. Eye tracking devices and proper training, as well as access to a speech and language therapist and a computer engineer, were provided for the duration of the study. Patients completed questionnaires related to their experiences with the equipment several times during the study. Several patients reported a clear positive impact on their quality of life during the study, resulting from the enhanced communication facilitated by the eye tracking devices over other non-gaze based assistive devices.

1.1 Related Work

While the utility of gaze interaction in a variety of applications has been demonstrated, the availability of the technology has been a limiting factor in more
widespread use. Due to the relatively high monetary cost and proprietary nature associated with commercial eye tracking equipment and software, several low-cost solutions have been developed using inexpensive off-the-shelf components. Many of these designs have been made publicly available through the open source community. The OpenEyes project [3] presents a low-cost head-mounted eye tracker which uses a pair of inexpensive IEEE-1394 cameras to capture images of both the eye and scene. This hardware device, coupled with the open source Starburst algorithm [4], facilitates estimation of the user PoG in the 2D scene image. A similar open source project, the EyeWriter [5], provides detailed build instructions for creating a head-mounted eye tracker from a modified Playstation Eye USB camera. The project was designed to enable digital drawing by eye gaze control for artists with ALS while using the device with the accompanying open source software. Interestingly, in [6], the effectiveness of a low-cost eye tracker is shown to be comparable to that of commercial devices for target acquisition and eye-typing activities.

The head-mounted eye gaze systems mentioned above facilitate effective interactive experiences with some limiting constraints. In general, these solutions are designed for interaction with fixed computer displays or 2D scene images. These types of systems provide a 2D PoG, which does not directly translate into the 3D world without additional data modalities. An accurate estimate of the 3D user PoG can be especially useful in mobile and ubiquitous applications, human-robot interaction, and in designing intelligent assistive environments. Knowledge of the 3D PoG within an environment can be used to detect user attention and intention to interact [7], leading to multimodal attentive systems able to adapt to the user state. Additionally, assistive robotic devices, such as mobile manipulators or intelligent wheelchairs, could be issued commands based on the user’s PoG when verbal or tactile input is
not feasible. This would lead to an overall increase in user independence and quality of life.

Some mobile 3D PoG tracking systems have been proposed in literature. In [8], a head-mounted multi-camera system is presented that estimates the 3D PoG by computing the intersection of the optical axis of both eyes. This approach gives the 3D PoG relative to the user’s frame of reference, but does not provide a mapping of this point to the environment in which the user is present. A similar stereo camera approach is presented in [9], which also includes a forward facing scene camera for mapping of the 3D PoG to scene coordinates. While multi-camera approaches such as these provide a 3D PoG, their use is limited by increased uncertainty at increasing PoG depths. Another limiting factor is the scene camera, which is generally a standard 2D camera that does not provide any 3D information of the environment itself.

1.2 Hypothesis

The key hypothesis of this dissertation work is that currently emerging or under-utilized data modalities, such as eye tracking and 3D scanning, can be combined with existing robotics platforms in such a way that the resulting system will enable a severely disabled user to perform environmental interaction and object manipulation beyond the capabilities provided by currently existing approaches. Additionally, the realization of the hypothesis meets another important criterion: that relatively inexpensive and readily available hardware devices can provide the majority of the necessary human interfacing capabilities for such a solution, particularly by utilizing eye gaze tracking and environmental RGB-D scanning techniques.
1.3 Overview

The underlying goal of this work is to prove the proposed hypothesis by developing gaze aware assistive systems with the ability to provide environmental navigation, object manipulation, and communication capabilities to physically disabled users. The target user group in this study is subjects who are affected by physical disabilities such as ALS and paraplegia, specifically cases where upper and lower limb movement are completely or severely limited but eye movements are unaffected.

The realization of the robotic aids will take the form of intelligent electric wheelchairs and mobile manipulation platforms. The key contribution, the human interface, will take the form of a mobile headset device which will facilitate 3D PoG estimation by combining video occulography-based eye tracking with real-time 3D environment scanning. Eye tracking will provide the critical control input while all other modalities, such as blink detection or voice commands, will play a supporting role in perception of the user intent.

In Chapter 2, we develop the necessary PoG headset hardware and software. These tools provide the foundation for control of assistive robotic platforms of interest, which are discussed in Chapter 3. In Chapter 4, we develop specific applications using the previously mentioned technologies. Finally, in Chapter 5, we provide a discussion of open problems, future work, concluding remarks.
CHAPTER 2

TOWARDS MOBILE 3D POINT-OF-GAZE ESTIMATION

In this chapter, we present efforts to transition 2D PoG techniques into 3D in order to facilitate advanced gaze interaction. First, a publicly available dataset providing synchronized oculography video frames from a commercially available head-mounted eye tracker with tracking data from a high-fidelity motion capture environment is discussed. A low-cost, 3D head-tracking hardware solution which can be easily added to existing head-mounted eye trackers for interaction with fixed displays during head motion is presented. Finally, a novel head-mounted eye tracker with integrated 3D environment scanning is detailed. This solution provides true 3D PoG interaction with the environment around the user, independent of any fixed displays or orientation beacons.

While the utility of gaze interaction has been demonstrated, existing eye gaze systems suffer from some limiting constraints. In general, they are designed for interaction with fixed computer displays or 2D scene images, and the 2D PoG of these systems does not directly translate into the 3D world. An accurate estimate of the 3D user PoG within an environment is clearly useful, as it can be used to detect user attention and intention to interact [7]. For example, knowledge of the user 3D PoG could be used to identify objects of interests for manipulation by an assistive robot. An intelligent wheelchair could also utilize 3D PoG as a primary data modality for assisted navigation.

Furthermore, existing systems tend to lack mobility, and the mobile 3D PoG tracking systems that have been proposed in literature suffer from their own limita-
tions. The head-mounted multi-camera system presented in [8], for example, gives the 3D PoG relative to the user’s frame of reference, but does not map this point to the user’s environment. Finally, the high monetary cost and proprietary nature of commercial eye tracking equipment limits widespread use. This has led to interest in the development of low-cost solutions using off-the-shelf components.

2.1 A Dataset for Head-Mounted Eye tracker Evaluation and Benchmarking

While many different algorithms exist for processing video occulography data, there are very few standards or methods for providing quantitative performance analysis and benchmarking. In addition, acquiring video streams from head-mounted eye trackers requires expensive commercial devices or custom made hardware, access to test participants, and proper data synchronization which may not be easily attainable by all researchers. In the case of eye tracking under head motion, an estimate of the head pose relative to the viewing surface is necessary and also requires additional hardware.

In order to alleviate these challenges in the eye tracking research community, we developed a publicly available eye tracking dataset aimed to be used as a benchmark for Point of Gaze (PoG) detection algorithms [10, 11], headset devices, and calibration routines. The dataset consists of a set of videos recording the eye motion of human test subjects as they were looking at, or following, a set of predefined points of interest on a computer visual display unit. The eye video data was recorded using a commercially available head-mounted monocular eye tracker. The ground truth of the actual point of gaze and head pose in 3D are provided together with the data. The ground truth regarding the point of gaze is known in advance since the subjects are always looking at predefined targets, whereas, the head position in 3D is captured using a high-fidelity motion capture system.
2.1.1 Existing Datasets

Since the problem of eye tracking in general has been already studied by previous researchers, there have already been efforts for the creation of datasets to facilitate experiments with different aspects of the problem. However, to the best of our knowledge, all the previous publicly available datasets are either not well suited to the problem of PoG detection, or limited by other parameters such as insufficient head tracking accuracy or lack of ground truth which does not allow a reliable evaluation of methods developed for PoG detection via eye tracking. In this section we give an overview of existing eye tracking datasets and we explain their limitations regarding the problem of PoG detection.

In [12], the authors have collected a head pose and eye gaze dataset of 10 subjects using a web camera placed in front of the subjects. The subjects were instructed to perform a set of head and eye motions and then specific head pose and eye gaze estimation methods were tested on the collected data. The ground truth about the head’s pose was extracted using 3 LEDs mounted at the subject’s head. Using this method, one can determine the location and rotation of the head relative to the camera, but only in the 2D space, i.e. the subjects cannot move towards or far from the camera. As for the ground truth regarding the gaze position, only 3 classes of gaze directions were used: looking straight forward, looking to the extreme left and to the extreme right. That is insufficient for applications where the exact point of gaze needs to be detected in continuous space.

A similar dataset is provided in [13]. In this dataset the ground truth regarding the head pose was obtained by asking the subjects to point a laser beam mounted on their head to a specific location. An extra limitation of this dataset is that it only provides a set of images instead of video, which makes unsuitable for tracking problems.
The eye tracking dataset found in [14] has been generated for the purposes of examining visual attention models on a set of video sequences. The eye point of gaze on the display was determined using an out-of-the-box Locarna Pt-Mini head mounted eye tracker, therefore the tracking accuracy depends on the tracker used by the device itself and no ground truth is provided.

Finally, in [15], the authors provide a database of visual eye movements from 29 observers as they look at 101 calibrated natural images. The eye movements of the subjects as they look at the images are recorded using a Forward Technologies Generation V dual-Purkinje eye tracker, while holding a fixed head position. Again, the final PoG provided relies on the capabilities of the eye tracking system used.

What differentiates our dataset from the above ones, is that the ground truth of both the point of gaze on the display and head rotation/translation in the 3D space are known in advance, which makes it ideal for testing the accuracy of methods estimating the point of gaze based on the eye motion assuming that the head pose is known.

2.1.2 Data Collection Methodology

A total of 20 subjects (2 women and 18 men) participated in our data collection sessions. The participants were graduate and undergraduate students of the University of Texas at Arlington. Some of them had normal vision and some of them wore contact lenses or spectacles. The participants that normally wore spectacles did not use them during the data collection process. However, their vision level was still good enough to locate the displayed target on the monitor. The special characteristics of each subject are given as metadata together with the dataset.

Each subject participated in two video recording sessions in which they looked at (or followed) a target on a Samsung LN32C350 (32 inch, 1360x768 pixels) display.
In the first session the subjects were allowed to move only their eyes while keeping their head still, whereas in the second session, they were allowed to freely move their heads and eyes at their convenience. Figure 2.1 shows a photo taken during the data collection process. In the photo, the reader can see the experimental setup used for the data collection.

In the first session the subjects were asked to keep their head still and their eye motion while looking at different patterns of targets appearing on the computer display was recorded. In the first pattern, a target appeared at 9 different positions of the display for a few seconds and the subjects were instructed to look at the target as soon as it appears and until it disappears. Note that since the human eye may require a few milliseconds from the moment the target appears on display until the moment the eye point of gaze moves to fall onto it, the target location and the point of gaze
Figure 2.2. Example smooth pursuit target pattern shown to participant.

may not align for a few milliseconds after the appearance of each target. However, they should be aligned soon after.

Similarly, in the second pattern, 16 targets appeared on the display and the subject had to repeat the same procedure. The number of targets and their location on the display was chosen so as to resemble common calibration patterns used by eye trackers. Figure 2.2 shows an example of targets appearing on different locations of the display. The third pattern, instead of static targets, involved a target moving inside the display and the subjects had to follow the target with their eyes at all times. This patterns is particularly useful for eye tracking methods which do not statically determine the point of gaze but follow the center of the pupil or other eye features over time.

Since in real life people do not move only their eyes but also their heads to look at different targets (or follow a target as it is moving), in the second session the subjects repeated the same process as in the fist session, but this time they did not have any constraints regarding their head motion. The exact position of the head
in the 3D space was tracked by the Vicon system using a set of markers attached to the head mounted eye video recorder. For convenience, four markers were also attached at the corners of the computer display. This allows us to determine the exact location of the head and the display monitor in the same coordinate system. Figure 2.3 visualizes the locations of the head and the computer display in the 3D space as captured by the Vicon System.

Requiring from the users to keep their head still while moving only their eyes is a common practice used by many eye tracking systems which incorporate some kind of calibration before using the eye input. The accuracy of such systems deteriorates over time even with subtle head movement and they need to be re-calibrated to become usable again. One of the reasons that we included the session of trying to hold the head still in our collected data was to consider such cases and provide the option for experiments that evaluate the accuracy loss due the head position drifting. The
exact head pose and orientation is still tracked by the Vicon system and provided as ground truth.

2.1.2.1 Vicon Motion Capture System

The user head position and pose was measured during the data collection process using a Vicon motion capture environment. The motion capture system consists of 16 tracking cameras surrounding an area measuring roughly 10 x 10 meters. The system is able to track the position and orientation of multiple rigid structures equipped with reflective markers at a rate of 100 Hz with sub-millimeter accuracy. Using this information, we are able to reconstruct the homogeneous transformation matrix for each tracked structure at each time step. The tracked structures of interest in our case were the display unit and the subject’s head. Figure 2.4 shows the Vicon setup that was used during the data collection process.

Figure 2.4. Motion capture system setup at Heracleia Laboratory.
2.1.2.2 Eye Video Recording Device

The device used to obtain the eye video data is an Applied Science Laboratories Mobile Eye-XG. The Mobile Eye-XG is worn on each subject’s head during data collection, and is positioned such that the user’s right eye is centered in the video frame. The recording of the right eye over the left is advantageous in that a higher percentage of the population exhibits right eye versus left eye dominance. The video data is recorded with a resolution of 768 x 480 pixels at a frame rate of 29.97 Hz. Each video is provided as an individual AVI file encoded with the Motion JPEG Video (MJPG) codec. The Mobile Eye-XG projects a triangular infrared glint pattern on the user’s eye during recording, which can be used as an additional tracking feature at the discretion of the dataset user. The eye tracker with attached tracking markers is shown in Figure 2.5.

2.1.3 Coordinate Systems

In this section we describe the coordinate systems (CSs) of the proposed setup. We have the following CSs:

1. \( \{ W; x_w, y_w, z_w \} \), the world coordinate system located on the ground behind the test participant;
2. \( \{ M; x_m, y_m, z_m \} \), attached to the upper left corner of the monitor;
3. \( \{ C; x_c, y_c, z_c \} \), attached to the eye camera;
4. \( \{ E; x_{ce}, y_{ce}, z_{ce} \} \), attached to the center of the Mobile Eye-XG device.

Each CS is defined using the right-hand notation. For the CS \( \{ W \} \), \( z_w \) points up and \( x_w \) points to the right. The Vicon Motion Capture System returns position and orientation data relative to this CS. CS \( \{ M \} \) is attached to the center of the monitor, with \( z_m \) pointing up and \( x_m \) pointing to the right. CS \( \{ E \} \) is attached to
the center of the Mobile Eye-XG device, with $z_w$ pointing up and $x_w$ pointing to the right. This CS also corresponds to head position, as the Mobile Eye-XG is worn on, and does not move relative to the head. Figure 2.6 shows the relative assignment of CSs used in the dataset.

2.1.4 Dataset Structure and Format

In this section we describe the structure of the dataset and the details of the provided format. For each one of the twenty participants, the dataset includes six videos in avi format and corresponding metadata. File sets 00001, 00002, 00003, come from the first session in which the participant was to remain as still as possible, and file sets 00004, 00005, 00006, come from the second session in which the participant was freely allowed to move their head.
Figure 2.6. Coordinate systems used in the dataset.

File sets 00001 and 00004 contain the videos of the first pattern of each session (a target appearing in 9 different locations of the display unit), file sets 00002 and 00005 contain the second pattern (16 target locations), and file sets 00003 and 00006 contain the videos coming from third pattern (target moving within the display).

The metadata includes two homogeneous transformation matrices, $W_M H$ from $\{W\}$ to $\{M\}$ and $W_E H$ from $\{W\}$ to $\{E\}$, and it also includes the $(i, j)$ pixel location of the target location for each video frame. This data is packaged as a MATLAB (.mat) data file and as a CSV file. The .mat file contains a structure which includes the two homogeneous transformation matrices and the pixel locations. The data is written into the CSV file with each column corresponding to a frame in the video, rows 1:16 represent the flattened homogeneous transformation matrix $W_E H$, rows 17:32 represent the flattened homogeneous transformation matrix $W_M H$, and rows 33:34 represent the
flattened pixel locations. The transformation matrices are unflattened by their $(i, j)$ value being the $(((i-1) \times 4)+j)$ of their respective columnar data. The pixel locations are unflattened by their $(i, j)$ value being the first and second values respectively. Pixel locations can be easily converted to metric units using $\frac{W}{M}$, $H$ and the known display resolution.

Figure 2.7 shows a single frame from the video set, while Table 2.1 shows the metadata corresponding to the video frame and participant.

2.1.5 Results

To demonstrate the utility of the proposed dataset, we used it to evaluate the performance of the popular starburst algorithm [4] compared to some other conventional approaches.

The starburst algorithm works by roughly estimating the pupil center (simple thresholding), fitting an ellipse to the pupil 'blob', and then refining the ellipse by
considering pupil edge points, which lie on rays projected outward from the center of the first ellipse. It then finds a corneal reflection (a small dot made from an IR LED), and computes the vector between the pupil ellipse center and corneal reflection center.

In the experiments, we evaluated the effectiveness of the starburst algorithm versus simple pupil blob center tracking and rough ellipse fitting. The blob center method tracks only the center of mass of the thresholded pupil image, while the fitted ellipse method tracks the center of the best-fit ellipse around the pupil blob with no
other refinements. These methods are each performed using the same 5 test video sets which were selected from the overall dataset of 20 participants.

Video based eye trackers are generally sensitive to factors such as eye color, eye shape, user age, previous corrective surgery, camera position, etc. This can often be mitigated by adjusting tracking parameters for each user, but occasionally a tracker may fail completely under certain conditions. The test video sets were chosen such that they gave meaningful, easy to compare results for each of the methods that we evaluate, and also in order to perform the necessary processing in a reasonable amount of time.

In each case, the tracker was calibrated using the 9 point calibration pattern video corresponding to the test set. A linear mapping between the pupil/CR vectors and known screen coordinates was created, one for each of the evaluated methods for each participant. Each method is run on the remaining two head motion-free videos acquired during the first participant recording sessions. Each estimated point-of-gaze (PoG) was measured against the known screen target locations, provided by the dataset metadata files for each video. The videos containing head motion from the second participant recording sessions were not used, since the trackers we evaluated were not designed with this in mind.

The RMSE error of each method is presented in Table 2.2. Simple outlier rejection was performed on the tracking results (an error over 100 pixels was removed from consideration). This was generally caused by blinking and/or delay as the user fixates on a new target location. No other filtering of the data was performed. Using simple outlier rejection, 8.49% of the tracking estimates were removed (i.e. 91.51% of the frames where kept to compute the results in Table 2.2).
### Table 2.2: Experimental results of the 3 tracking methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>blob center</th>
<th>ellipse center</th>
<th>starburst</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outliers</td>
<td>93</td>
<td>51</td>
<td>42</td>
</tr>
<tr>
<td>Average error</td>
<td>38.22</td>
<td>26.49</td>
<td>17.75</td>
</tr>
<tr>
<td>MSE</td>
<td>1869.18</td>
<td>1017.11</td>
<td>513.22</td>
</tr>
<tr>
<td>Root MSE</td>
<td>43.23</td>
<td>31.89</td>
<td>22.65</td>
</tr>
</tbody>
</table>

#### 2.2 3D Head Pose Tracking Relative to a Fixed Frame

In order for a PoG to be computed accurately in either 2D or 3D, the head position relative to the world must be known. In the simplest case, this can be done by limiting the head motion of the user such that the calibration mapping from pupil pixel locations to screen coordinates is performed once per session with a constant head pose. While this simple approach offers good performance initially, the calibration tends to “drift” even when the head is physically constrained with a chin rest or similar rigid mount due to accumulated changes in head pose from even the smallest muscle movements. In addition to the drift problem, it is often impractical or undesirable to constrain the head motion of the user due to factors like comfort and usability.

Tracking of the user head pose makes it possible to apply a transformation to the calibration mapping in order to correct for head motion. The difference between the head pose during the calibration routine and the frame of interest gives this corrective transformation. While this approach supports head motion, accurate head tracking is generally too expensive or physically cumbersome to use in practice for head-mounted eye trackers. To address this, we present a low-cost solution for real-time tracking of a human user’s head position with respect to a video display source for eye gaze estimation in an assistive setting [16]. The solution utilizes a wearable
headset equipped with sensors found in commercially available off-the-shelf video gaming devices in order to minimize hardware complexity and expense. A pair of Nintendo Wiimote imaging sensors are used to create a stereo camera for 6 DOF position tracking of the headset, while a modified Playstation Eye monocular camera is used to track the pupil position. The resulting tracking hardware is able to measure the 3D position of four infrared LEDs mounted at known locations on the video display using triangulation of the stereo camera data. Integration of the head tracking estimate with a computer vision based pupil tracking solution in order to compute the user’s point of gaze is also described.

2.2.1 Tracking Hardware

The head and eye tracking solution consists of four key subsystems: the stereo camera assembly, eye tracking camera, video display with infrared markers, and the host computer. The stereo camera and eye tracking components are both attached to the headset worn by the user, while the video display and host computer are located remotely. The headset structure is provided by a pair of modified sport sunglasses for comfort. Figure 2.8 shows the assembled headset.
The stereo camera subsystem is used to obtain the 3D positions of the display markers relative to the headset. These measurements, obtained using stereo triangulation, are then used to estimate the position and orientation of the headset relative to the monitor.

Implementation of this subsystem was achieved using a pair of Pixart imaging sensors extracted from Nintendo Wiimote controllers. These sensors consist of a CMOS camera with integrated vision processing, capable of tracking up to four infrared light sources at a rate of 100 Hz. These specifications allow for the creation of an effective stereo tracking solution for fixed sources of infrared light, such as LEDs. The sensors are mounted on the left and right sides of the headset frame in such a way that they do not obstruct the view of the user.

Before the stereo camera subsystem can be used to obtain 3D positions, the imaging sensors must undergo a calibration process. Through this process, an estimation of the intrinsic parameters of both imaging devices, as well as the extrinsic parameters of the stereo pair can be acquired. In our approach, the Camera Calibration Toolbox for Matlab [17] is used to obtain the calibration constants. These constants need only be computed once, assuming that the position and pose of the two imaging devices relative to each other are fixed.

A 13 x 13 calibration grid was created using a perforated board with small holes drilled on a 1 inch grid. Data from the Wiimotes is collected until all 169 distinct grid points are detected as an infrared LED is moved along the back of the fixed board. Once enough points are collected, the data is saved in an image file and the stereo pair is moved to a different position and orientation. This process is repeated until enough calibration images are acquired (in our case, 12 different positions). The resulting image set is then used to estimate the intrinsic and extrinsic camera parameters using the calibration toolbox.
The eye tracking subsystem, consisting of a single camera, is used to extract the pupil position during eye movements using computer vision techniques. In our system, a Playstation Eye device is positioned in front of the left eye such that the pupil image is as large as possible in the camera frame while minimizing the visual obstruction to the user. The Playstation Eye device was chosen based on the results presented in previous work on the EyeWriter project [5]. The device is capable of providing images with a resolution of 640x480 at a rate of 60 Hz. The interface to the host computer is provided using the USB 2.0 standard, which removes the need for a dedicated video capture interface for real-time streaming of images.

The Playstation Eye Camera was modified to record close-up infrared video of the pupil using the same method presented in the EyeWriter project. This resulting high-quality images are robust to changing conditions in ambient light. A captured image using the eye camera is shown in Figure 2.9.
The Playstation Eye device comes equipped with an infrared filter and wide field of view lens. In order to provide close-up infrared video of the pupil, the filter and lens must be replaced. The modification process used in our approach is identical to the method documented by the EyeWriter project. After replacing the factory lens and image filter, two 850nm infrared LEDs were added to the camera in order to illuminate the user’s eye outside of the visible light spectrum. This wavelength was chosen to minimize the possibility of interference with the Wiimote sensors, which are most sensitive to 940nm infrared light [18]. This resulting high-quality images are robust to changing conditions in ambient light. A captured image using the eye camera is shown in in Figure 2.9.

2.2.2 Coordinate Systems

In this section we describe the coordinate systems (CSs) of the proposed setup. We have the following CSs:

1. \( \{M; x_m, y_m, z_m\} \) attached to the upper left corner of the monitor, and coincides with the world CS.
2. \( \{CL; x_{cl}, y_{cl}, z_{cl}\} \) attached to the left camera
3. \( \{CR; x_{cr}, y_{cr}, z_{cr}\} \) attached to the right camera
4. \( \{O; x_{ce}, y_{ce}, z_{ce}\} \) attached to the occulography camera that monitors the eye

The left/right cameras make a stereo pair and their calibration was performed using the matlab stereo calibration toolbox. The calibration calculated the intrinsic parameters of the cameras and the homogeneous transformation matrix \( ^{CL}H_{CR} \) from the left to the right camera.

On the four corners of the monitor we fixed four LEDs (their positions related to \( \{M\} \) are known), which can be recognized in the images of our stereo cameras after embedded processing. The next step is stereo triangulation of the identified
points. The matching of the points is simply done by matching the respective corners of the projected rectangle in both images. Once the points are obtained, we extract the homogeneous transformation matrices $M_{CL}^H$ and $M_{CR}^H$ from the monitor $\{M\}$ to $\{CL\}$, and $\{CR\}$ using the least-squares approach described in [19].

The position of the eye in the occulography frame $\{O\}$ can then be expressed in the monitor reference frame $\{M\}$ using the equation

$$M_p = M_{CL}^H \cdot CL_O^H \cdot O_p$$  \hspace{1cm} (2.1)$$

where $O_p$ can be estimated along with the supporting plane with an algorithm like the “one circle” [20]. Knowing the supporting plane and by taking the normal vector that passes through the image center it is possible to find the intersection with the monitor plane, which gives the point of gaze. The transformation matrix $CL_O^H$ can be estimated either by mechanical means or by solving (2.1) for several points and applying a regression approach such as [21].

2.2.3 Experimental Setup

Testing and evaluation of the headset tracking accuracy was performed in a motion capture environment. The motion capture system consists of 16 tracking cameras surrounding an area measuring roughly 10 x 10 meters. The system is able to track the position and orientation of rigid structures equipped with reflective markers at a rate of 100 Hz with sub-millimeter accuracy. This provides a ground-truth comparison for our solution. Transformation matrices for the marked headset and monitor positions are saved along with the estimates provided by stereo triangulation at each time step. This data is synchronized such that the tracking error of the triangulation estimate relative to the motion capture data can be computed offline.
2.2.4 Results

In order to compare the tracking estimate of the headset to with that of the motion capture system, the individual roll, pitch, and yaw angles were extracted from the transformation matrices along with the corresponding position components. The tracked position values by both the headset and motion capture system are shown in Figure 2.11(a), while the rotational components are shown in Figure 2.11(b). The solid blue lines correspond to the headset tracking values, while the dashed red lines correspond to the motion capture system.

The root-mean-square errors (RMSE) were computed individually for each position and orientation component and are shown in Table 2.3. The measurements from the motion capture system and headset were not filtered prior to analysis, which contributes to much of the error found in the results. A simple low-pass filter would
Figure 2.11. Comparison of stereo tracking and motion capture.
Table 2.3: RMSE of position and orientation components.

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>φ</th>
<th>θ</th>
<th>ψ</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE</td>
<td>31.465</td>
<td>57.829</td>
<td>55.516</td>
<td>0.045</td>
<td>0.038</td>
<td>0.0517</td>
</tr>
</tbody>
</table>

significantly reduce the error, as the Wiimote sensors readings to contain some noise. Still, even with the amount of error present in the experiment data, the approach is accurate enough to provide a useful degree of tracking capabilities.

2.3 Pseudo 3D Mobile PoG Tracking

In this section, we present a low-cost, wearable headset 3D Point of Gaze (PoG) estimation in assistive applications [22, 23]. The device consists of an eye tracking camera and forward facing RGB-D scene camera which, together, provide an estimate of the user gaze vector and its intersection with a 3D point in space. The resulting system is able to compute the 3D PoG in real-time using inexpensive and readily available hardware components. We describe this approach as “pseudo 3D” because of the relaxed calibration constraints used to compute 3D gaze points. A truly 3D extension of this system is described in following sections.

2.3.1 Tracking Hardware

The solution presented in this section is designed to provide information about the environment existing around the user, together with the points or areas within the environment that the user interacts with visually. In order to realize these goals, a wearable headset was developed that provides a 3D scan of the area in front of the user, a color image of this area, and an estimate of the user’s visual PoG. These 3 data modalities are provided by an eye tracking camera, which observes the user’s
eye motions, and a forward facing RGB-D camera, providing the scene image and 3D representation. These two components are mounted on rigid eyeglass frames such that their position remains fixed relative to the user’s head during movement. The complete headset hardware solution is shown in in Figure 2.12.

The system eye tracking feature is accomplished using an embedded USB camera module equipped with an infrared pass filter. The user’s eye is illuminated with a single infrared LED to provide consistent image data in various ambient lighting conditions. The LED also produces a corneal reflection on the user’s eye, which can be seen by the camera and exploited to enhance tracking accuracy. The LED was chosen according to the guidelines discussed in [24] to ensure that the device could be used safely for indefinite periods of time.

The eye camera is positioned such that the image frame is centered in front of one of the eyes of the user. The module can be easily moved from the left or
right side of the headset frame so that either eye may be used (to take advantage of user preference or eye dominance), while fine adjustments to the camera position and orientation are possible by manipulating the flexible mounting arm. Streaming video frames are provided with a resolution of 640x480 at a rate of 30 Hz, which facilitates accurate tracking of the pupil and corneal reflection using computer vision techniques such as [25, 4, 26].

Information about the environment in front of the user is provided by a forward facing RGB-D camera, the Asus XtionPRO Live. This device provides a 640x480 color image of the environment along with a 640x480 depth range image at a rate of 30 Hz. The two images are obtained from individual imaging sensors and registered by the device such that each color pixel value is assigned actual 3D coordinates in space. This provides a complete scanning solution for the environment in the form of 3D “point clouds”, which can be further processed in software [27]. Figure 2.13 shows an example of the 2D RGB color scene image and 3D point cloud returned by the RGB-D camera.

2.3.2 Point of Gaze Estimation

An estimate of the user PoG is computed using a modified version of the starburst algorithm presented in [4]. This algorithm creates a mapping between pupil positions and 2D scene image coordinates after a simple calibration routine is performed. During the pupil detection phase of the algorithm, an ellipse is fitted to the pupil such that the ellipse center provides an accurate estimate of the pupil center. The center of the infrared corneal reflection is detected during the next phase of the algorithm, which is then compared to the pupil center to acquire a difference vector. The difference vector between the pupil center and corneal reflection is then used to create the calibration mapping. Figure 2.14 shows the fitted pupil ellipse, pupil
Figure 2.13. Scene color and depth camera information.
center, corneal reflection, and difference vector computed from a single eye camera image frame.

The mapping from difference vector to 2D scene image coordinates is made possible by a 9 point calibration procedure. During calibration, the user sequentially gazes upon 9 different points in the scene image. The difference vector for each calibration point is saved, and the 9 point mapping is used to interpolate a 2D PoG from future eye camera frames. The 3D PoG can be obtained easily from the 2D point by looking up the 3D coordinates of the pixel in the point cloud data structure provided by the RGB-D camera. Exploitation of the RGB-D point cloud structure removes the need for stereo eye tracking during 3D PoG estimation as used in the previously mentioned methods [8, 9].

2.3.3 Results

The head-mounted 3D PoG estimation system was shown to accurately detect the gaze vector intersection with points in the 3D scene image using a simple 2D calibration with the forward facing RGB camera. We have shown that by including a 3D scan of the user’s field of view, we can eliminate the need for the multiple eye tracking cameras used in stereo systems that can obstruct the user’s vision. We have shown that with our system, we can estimate the user’s gaze vector and determine the most likely 3D PoG. We have also demonstrated the benefit of using the additional scene depth information provided by the RGB-D camera, rather than a 2D scene camera alone. As the technology matures and devices are further miniaturized, our system will provide an increasingly attractive option for eye tracking and 3D PoG estimation.

One significant limitation of the simple calibration approach is that the accuracy of the gaze vector intersection decreases as the distance between the RGB-D camera
(a) Detection of pupil and CR

(b) Scene image annotated with PoG

Figure 2.14. Mapping of gaze vector to scene.
and 3D PoG deviates from the distance used during calibration. This is because the optical axis of the RGB-D camera is offset from that of the user. In following sections, we develop true 3D calibration routines to address this caveat.

2.4 Full 3D Mobile PoG Tracking

While the approach presented in the previous section is able to provide a “pseudo 3D” PoG by taking advantage of registered RGB-D point clouds, it ignores the intermediate step of computing a true 3D gaze vector. This introduces an error source, as the scene camera does not share the same pose as the user’s tracked eye. Improving upon [23], we provide a full 3D calibration process and mathematical solution for the necessary coordinate transformations and parameters necessary to compute a true 3D gaze vector. We also provide details of our experimental setup and a discussion of future work.

2.4.1 Approach

The gaze tracking headset relies on coordinate frames defined at the scene camera \( \{S\} \), occulography camera \( \{O\} \), and the center of the user’s eye \( \{E\} \). The assignment of coordinate frames is shown in Figure 2.15.

Before the system can accurately estimate 3D gaze vectors, a two-step calibration process is performed. The first step is to solve for the transformation from the scene and occulography frames using hand-eye calibration. The second step is to obtain a mapping from pupil center pixel coordinates to 3G gaze vectors using a least-squares fitting approach. Three basic assumptions about the operation of the device are made:

1. The occulography camera is oriented such that the user’s eye lies on the camera’s optical center.
2. The focal length of the occulography camera is fixed.

3. The coordinate frames \(\{S\}\), \(\{O\}\), and \(\{E\}\) are fixed prior to calibration and do not change once calibration is performed.

An outline of the system initialization steps (hardware calibration and user calibration) is provided below.

1. The headset is adjusted to the user such that the device is securely fitted and the occulography camera is suitably positioned.

2. The transformation from \(\{S\}\) to \(\{O\}\) is computing using hand-eye calibration.

3. The transformation from \(\{O\}\) to \(\{E\}\) is computed by translating along the \(z\) axis of \(\{O\}\) a distance of \(d\).

4. The pupil position is tracked using the occulography camera while calibration gaze points are measured by \(\{S\}\).
5. The calibration points are transformed into the \( \{O\} \) reference frame and used to compute a calibration mapping of pupil center locations to 3D gaze vectors.

6. After calibration, pupil center locations in the occulography camera are used to compute gaze vectors which are transformed into the \( \{S\} \) frame to acquire a 3D PoG.

2.4.2 Coordinate Transformations

In order to compute 3D gaze vectors originating from the eye using known 3D gaze points measured with respect to the scene camera, we must first know the transformations between the scene \( \{S\} \), occulography \( \{O\} \), and eye \( \{E\} \) reference frames. The transformation of 3D points from \( \{O\} \) to \( \{S\} \) can then be expressed in terms of 3x3 rotation and 3x1 translation matrices as follows

\[
\overline{S} p = \overline{S} t + \overline{S} R \overline{O} p \tag{2.2}
\]

Premultiplying both sides by \( \overline{S} R^T \), we obtain

\[
(\overline{S} R^T) \overline{S} p = (\overline{S} R^T) \overline{S} t + (\overline{S} R^T) \overline{S} R \overline{O} p \tag{2.3}
\]

\[
\overline{O} p = -(\overline{S} R^T) \overline{S} t + (\overline{S} R^T) \overline{S} p \tag{2.4}
\]

Similarly, we define the transformation of points from frame \( \{E\} \) to \( \{O\} \) as

\[
\overline{O} p = \overline{O} t + \overline{O} R \overline{E} p \tag{2.5}
\]

Which can then be expressed as

\[
(\overline{O} R^T) \overline{O} p = (\overline{O} R^T) \overline{O} t + (\overline{O} R^T) \overline{O} R \overline{E} p \tag{2.6}
\]

35
\[ E_p = - (O_E R^T) E_t + (O_E R^T) O_p \]  

(2.7)

We define the reference frame \( \{E\} \) to have the same rotation as \( \{S\} \), such that

\[
O_E R = O_S R = S_R T
\]  

(2.8)

Additionally, under the assumption that the headset is positioned such that the center of the eye lies along the optical center of the occulography camera, the translation from \( \{O\} \) to \( \{E\} \) is defined as

\[
O_E t = \begin{bmatrix} 0 \\ 0 \\ d \end{bmatrix}
\]  

(2.9)

where \( d \) is the distance along the optical center of the occulography camera to the center of the eye. Substituting (2.3) into (2.6) yields

\[
E_p = - (O_E R^T) E_t + (O_E R^T) [ - (O_S R^T) S_t + (O_S R^T) S_p ]
\]  

(2.10)

\[
E_p = - (O_S R) E_t + S_R [ - (O_S R^T) S_t + (O_S R^T) S_p ]
\]  

(2.11)

Where \( O_S R \) and \( O_S t \) can be computed using hand-eye calibration [28, 29]. We are now able to express a normalized gaze vector \( \hat{g}_i \) originating from \( \{E\} \) as

\[
\hat{g}_i = \frac{E_p_i}{\|E_p_i\|}
\]  

(2.12)
2.4.3 Gaze Vector Calibration

We now create a calibration mapping between pupil center pixel coordinates \((u, v)\) in the occulography image to gaze vectors. We formulate this mapping as a least-squares estimation problem \([21]\) using the following regression models

\[
\theta = a_0 + a_1 u + a_2 v \tag{2.13}
\]

\[
\phi = b_0 + b_1 u + b_2 v \tag{2.14}
\]

where \(\theta\) and \(\phi\) are the angular components of the estimated gaze vector in spherical coordinates. Given a collection of \(n\) 3D calibration vectors \(\hat{g}_i\) computed using (2.12) and corresponding pixel values \((u_i, v_i)\), we define the calibration data matrices \(X\), \(Y_\theta\), and \(Y_\phi\) as

\[
X = \begin{bmatrix} 1 & u_0 & v_0 \\ 1 & u_1 & v_1 \\ \vdots & \vdots & \vdots \\ 1 & u_n & v_n \end{bmatrix}, \quad
Y_\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}, \quad
Y_\phi = \begin{bmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_n \end{bmatrix} \tag{2.15}
\]

where \(Y_\theta\), and \(Y_\phi\) contain the angular components of \(\hat{g}_i\) in spherical coordinates. The least-squares estimators \((a_0, a_1, a_2)\) and \((b_0, b_1, b_2)\) are then solved using the following regression formulas

\[
\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = (X^T X)^{-1} X^T Y_\theta \tag{2.16}
\]
\[
\begin{bmatrix}
  b_0 \\
  b_1 \\
  b_2
\end{bmatrix} = (X^T X)^{-1} X^T Y_\phi
\] (2.17)

Once the least-squares estimators are computed, a 3D gaze vector can be found from pupil center pixel coordinates using (2.13) and (2.14).

2.4.4 Hardware Implementation

Information about the environment in front of the user is provided by a forward facing RGB-D camera, the Asus XtionPRO Live. This device provides a 640x480 color image of the environment along with a 640x480 depth range image at a rate of 30 Hz. The two images are obtained from individual imaging sensors and registered by the device such that each color pixel value is assigned actual 3D coordinates in space. This provides a complete scanning solution for the environment in the form of 3D "point clouds", which can be further processed in software.

The system eye tracking feature is accomplished using dual embedded USB camera modules equipped with an infrared pass filter. While the system features binocular occulography cameras, only one is activated in the approach presented in this paper (we intend to explore binocular pupil tracking with the headset in future work). The user’s eye is illuminated with a single infrared LED to provide consistent image data in various ambient lighting conditions, which was chosen according to the guidelines discussed in [24] to ensure that the device could be used safely for indefinite periods of time. The pixel resolution of the device is 640x480 with a 30 Hz refresh rate.

The completed headset, shown in Figure 2.17, is mounted to a head strap that minimizes movement of the sensors relative to the user’s eye. The occulography
cameras are mounted on adjustable tubing arms that remain rigid during use, ensuring that the hand-eye calibration remains valid once the cameras are positioned for a particular user. A set of 3D printed mounts, shown in Figure 2.16 allow the sensor assemblies to be detached from the head strap such that the hand-eye calibration can performed easily.

2.4.5 Experimental Setup

In order to evaluate the performance of our approach, we must first perform hand-eye calibration to solve for $\mathbf{SR}$ and $\mathbf{St}$. Because the headset must be adjusted to each user to achieve the best results, we want to use a calibration approach that can be performed quickly with accurate results. The endoscope calibration solution presented in [30] allows us to solve for $\mathbf{SR}$ and $\mathbf{St}$ by collecting images and pose information of the headset while the eye camera is directed toward a small (3 cm square) calibration
grid. The grid is placed on a stationary flat surface while the headset is moved. As the scene camera is directed upward toward the ceiling, point clouds are collected and registered using the Iterative Closest Point (ICP) based solution presented in [31]. At each time step, the pose of the scene camera relative to the world is computed and paired with the image of the calibration grid acquired by the occulography camera. These inputs are sufficient to solve for $\hat{O\mathbf{R}}$ and $\hat{O\mathbf{t}}$ using hand-eye calibration. We also set $d = 0.064$ meters in $O\mathbf{t}$ for our particular occulography camera, which was estimated by measuring the distance from the occulography camera to the eye when
the image was optimally focused. A calibration grid image and reconstructed scene camera pose acquired during the hand-eye calibration process is shown in Figure 2.18(a) and 2.18(b).

The next step is to perform the gaze vector calibration discussed in section 2.4.3. We accomplish this by directing the user to gaze upon a tennis ball placed in the environment. Point clouds from the scene camera are paired with eye images from the occulography camera until a sufficient amount of images are acquired. The 3D centroid of the tennis ball is easily computed using RANSAC based segmentation [23], while the pupil center coordinates in the eye images are computed using the pupil tracker presented in [25]. In our experiments, we acquire 15 calibration points at distances ranging from 2 to 4 meters. Once the least-squares calibration mapping is computed, the device is able to estimate gaze vectors in real-time. A tracked pupil image and point cloud pair taken during the gaze vector calibration process is shown in Figure 2.18(c) and 2.18(d), and a reconstruction of the system coordinate frames and calibration gaze vectors is shown in Figure 2.19.

Once all of the necessary calibration steps are performed, we are ready to evaluate the performance of the system. Using test eye and scene image pairs acquired in the same manner as described above, we compare the estimated gaze vectors to the known gaze points in the 3D scene. Evaluation was performed with a single user using data acquired during 3 different calibration routines. In each experiment, 15 hand-eye poses and 15 3D calibration data points were acquired, along with 20 test data points.

2.4.6 Results

The average angular error over the 3 test routines was found to be 7.41 degrees. The main sources of error are as follows. First, the hand-eye calibration introduces
some error when calculating the back projection of the calibration grid images. Second, our estimation of the parameter $d$ is not perfect due to the difficulty in measuring this distance by hand (we hope to eliminate this measurement in the future using the camera intrinsic calibration parameters and a model of the human eye). Third, the RGB-D scanner was not recalibrated prior to our initial experiments, which contributes to some error when detecting the tennis ball. Finally, the linear least-squares calibration approach does not fully model the non-linear mechanics of the eye (this could be mitigated by a different approach, such as [32]). While the pupil tracking
Figure 2.19. Reconstruction of coordinate frames and calibration gaze vectors.

approach also adds some error during real-time operation, we visually inspected each test image frame and found this error to be quite small.

Though the error from our initial experiments was larger than existing 2D approaches, the mobility and true 3D nature of our system still provides many advantages. The accuracy of our system should substantially improve when the scene camera is properly calibrated and a more appropriate non-linear eye model is utilized.
CHAPTER 3
ASSISTIVE ROBOTIC PLATFORMS

Assistive devices have proven to be valuable tools in improving the quality of life for individuals suffering from a wide array of disabilities. The field of robotics in particular has shown great potential in addressing the challenges of some disabilities for which treatment options have remained relatively unchanged for some time, though mobile robots have yet to see widespread use. This is due partly to the difficulty in effectively perceiving human intent, especially when traditional input devices are not available, while guaranteeing safety for nearby individuals. Additionally, mobile robots operating in human-centric environments face mobility constraints which are not present in robot-centric environments, such as factories and warehouses. Because of this, there remains a significant amount of work to be done in robotic platform design and evaluation.

In this chapter, we present various robotic platforms developed for assistive scenarios. First, an outdoor UGV for visually impaired users is discussed. Second, a highly mobile holonomic mechanical platform is presented. Finally, an intelligent wheelchair designed to support non-traditional human computer interface methods from previous chapters is detailed.

3.1 A Guide Dog Inspired Assistive UGV Platform

Many assistive scenarios involving mobile robots will require the device to assume the role of a navigational aid. Users with cognitive, visual, or motor impairments stand to benefit from robotic platforms with localization, navigation, and obstacle
avoidance capabilities. One of the more obvious uses for this technology would be for the benefit of blind users. Developing a truly useful assistive robotic platform for the blind would also provide benefits in other scenarios, since the underlying navigational capabilities are universally necessary.

Perhaps the most identifiable and useful aids for the blind are specially-trained guide dogs. Guide dogs are trained to lead their owner through the environment while recognizing and avoiding hazards such as stairs, doors, and moving objects. Guide dogs have proven to greatly increase the mobility and safety of blind people by allowing them to traverse great distances on foot in both indoor and outdoor environments. Guide dogs are able to effectively avoid obstacles while remaining on a given path. While the benefits of guide dogs are undeniable, many factors make them unsuitable or impractical in several situations. Guide dogs require extensive training before they can be matched with a user. Once a suitable dog has been identified and trained, the person must also be trained to operate as a single unit with the dog. This can make guide dogs less suitable for those who have difficulties adjusting to the special relationship needed for the team to be effective. Allergies can also be a limiting factor, though certain breeds can be used to mitigate this problem in some cases. Perhaps the greatest limiting factor in guide dog use is their cost. The average cost of a guide dog in the United States, including the necessary training for both the dog and the user, is $42,000 [33]. The location and availability of effective training schools, together with the time required for training, make guide dogs impractical in many cases.

The use of robotics is a promising alternative to guide dogs. A properly designed robot-navigation aid could be both performance and cost effective. Recent advances in sensing technology, particularly in computer vision, allow a robot to identify paths and objects, thus enabling effective location and hazard recognition in both indoor and
outdoor environments. In addition to possibly matching the abilities of a biological guide dog, this technology could enable the robotic aid to respond to more complex commands. A location-aware robot would remove the burden of high-level navigation from the user, resulting in more capable solution.

We present here the design and development of an assistive guide robot that provides vision-based navigation as well as laser-based obstacle and collision avoidance to a visually impaired person. The novelty of our implementation is the use of RANSAC with adaptive thresholds to estimate the vanishing point of the visual scene as well as the fusion with the laser data in order to navigate the robot both indoors and outdoors. The feasibility and effectiveness of the approach is demonstrated by successfully guiding the user down the center of a path, such as a hallway or sidewalk, while navigating around obstacles and walls.

3.1.1 Related Work

The problem of guiding a robot along a certain outdoor path or corridor belongs to the broader range of lane detection problems. These types of problems have been attacked using various approaches during the last years and even competitions like the DARPA Urban Challenge [34] or the Mini Grand Challenge [35]. There have been different approaches for lane detection, including particle filters [36] and homography [37]. One of the most popular approaches consist of using RANSAC for robust clustering of multiple lines in the image space in order to determine the vanishing point in an image [38, 39, 40, 41].

Nevertheless most researchers have focused on car traffic and not assistive robots. To our knowledge there has been only one previous attempt of an assistive guide robot for the blind, which focused on indoor navigation [42, 43]. This system utilizes a Pioneer robotic platform, a SICK LIDAR for obstacle avoidance, a
webcam, and RFID in order to locate a specific product in a store and then navigated to its position. The navigation was accomplished using a colored tape on the floor of the aisles and RFID tags on the intersections. This system requires a number of modifications in an existing store in order to be deployed, such as RFID tags on all product packages and placing colored tape on all aisles. In contrast, our approach includes outdoor navigation and uses Computer-Vision techniques to navigate a specific path.

3.1.2 System Architecture

The main components for our iDog system are the iRobot platform, a notebook computer, a Logitech USB webcam and a Hokuyo LIDAR unit. The camera is the primary sensor for accomplishing the navigation task and is used to estimate the vanishing point from the captured video. This video sequence is processed using OpenCV 2.1 which extracts prominent lines in the image. Then RANSAC is used to determine the most probable vanishing point. After the vanishing point has been determined, the deviation from the principal point of the camera is calculated and the robot steers accordingly in order to move parallel to the direction of the road. While moving, the robot uses the LIDAR scanning system for obstacle detection and avoidance.

In addition to the navigation task, we provide a modular development platform to facilitate the evaluation of a variety of potential sensing and control approaches. This plug and play philosophy is loosely based around the Joint Architecture for Unmanned Systems (JAUS) [44]. The JAUS architecture is a collection of standards, originally developed by the United States Department of Defense, for unmanned systems. JAUS is designed to govern the way that unmanned systems are designed at the networked component level, as well as the networked agent level. While a
fully compliant JAUS implementation is out of the scope of this project, the software architecture utilized in the robotic guide dog follows many of the guidelines established by JAUS (distribution of software modules, UDP communication between modules, etc). This increases the level of interoperability at the component level, allowing a new software module to be quickly and easily integrated in the system without changes to other components.

3.1.2.1 Software Architecture

As previously mentioned, the architecture employed by the system is distributed at the component level. Individual software modules were created for specific tasks, such as polling a particular sensor or sending actuator commands to the iRobot platform. These background processes, or Daemons, communicate over a network protocol using UDP packets in a specific format. The modules are able to run on the same machine or on their own dedicated hardware simply by specifying the IP addresses and port numbers accordingly. This networked approach makes it possible to distribute computational load across several computers for increased performance and expandability. The software architecture employed by the iDog is shown in the diagram in Figure 3.1.

Each software module is assigned a unique IP address and port number pair, allowing them to be individually addressed across multiple hardware units. The vision and ranging daemons monitor the camera and the LIDAR, respectively, compute error signals, and send perceived values to the system controller. The system controller runs the vehicle PID controller, provides error reference signals to the sensing daemons, and visualizes the incoming data streams. Every aspect of the vehicle control law (such as the period, PID gains, and filtering parameters) is configurable from the system controller. Filtering of the incoming data streams is performed independently,
with both a moving average and moving median filter implemented. These filters, running in parallel, can be modified during execution to aid in testing and performance evaluation.

3.1.2.2 Vision Daemon

The vision daemon is responsible for estimating the pixel position of the vanishing point at each time instant. We will begin by describing the image processing techniques used which is based on the use of a Canny edge-detector in conjunction with Hough transform that extracts the prominent lines in each image. Since the effectiveness of this feature extraction step has been proven, we decided to also use
it in our assistive robot application. The main vision pipeline is illustrated in Figure 3.2. After the current frame is obtained, Canny Edge Detection algorithm is used to extract the image edges. The Hough transform is then adopted to extract only the most prominent image lines. Finally, RANSAC is adopted to detect the position of the vanishing point while simultaneously extracting those image lines parallel to the path.

Figure 3.2. Image processing pipeline.
3.1.2.3 Image Edge Detection

The first step of image processing is the detection of edges in each frame of the captured video. In order to improve the performance of our edge detector, each image has to be preprocessed. Initially the image is converted to grayscale, since Canny operates on this kind of images. Secondly, histogram equalization is performed on the image in order to increase contrast and therefore emphasize edges [45]. Histogram equalization is an image enhancement technique that operates on the spatial domain. It modifies the distribution of the pixels to become more evenly spread out over the available pixel range. Since a histogram of a grayscale image displays the distribution of the pixel intensity values, histogram equalization attempts to reshape the probability distribution function (PDF) into a uniform function. Therefore, although a dark image will have mostly low intensity pixels and a bright image only high intensity pixels thus lowering the contrast, an image with a uniform PDF will have pixel values at all valid intensities [46]. This process is shown in Figure 3.3.

Before proceeding to the actual edge detection, a median filter operates on the image. This is a smoothing filter that causes blurring but at the same time preserves larger edges. This aims at adjusting the size of edges that we want to preserve by choosing a corresponding filter size. In our application the median filtering uses a window of 5x5.

After preprocessing, the Canny edge detector is used to extract the edges of the image. The operation of the Canny edge detector in general can be summarized as follows. Normally, a Gaussian blurring filter is first convolved with the image in order to discard noise and unimportant edges, although this is not performed in the OpenCV implementation of Canny. Then, a Sobel operator is applied in order to calculate the gradient norms in the x and y axes. This is followed by hysteresis
thresholding which uses 2 thresholds in order to determine whether a pixel is an actual edge pixel, discarding small edges and noise. The high threshold for the canny edge detector is set adaptively using the formula presented in [47], while the low is set to 40% of the high threshold. Finally non-maxima suppression is applied on the image in order to reduce the edges thickness to 1 pixel to clearly define the contours in the image.

3.1.2.4 Line Detection Using the Hough Transform

Since our ultimate goal is to determine the vanishing point in the image, we first need to detect the perspective lines that intersect at the vanishing point. Therefore after acquiring the edges of the given image, we use the Hough transform of the road [46] to detect the most prominent lines. Hough transform is a voting algorithm that can be used to determine whether there are enough pixels to form a particular shape

Figure 3.3. Comparison of original and equalized histograms.
in the image, in our case a particular line. In order to accomplish that, each line has to be expressed in polar coordinates \((\rho, \theta)\), so that a generic pixel point \((x, y)\) belonging to a line will satisfy the following equation

\[
xcos(\theta) + ysin(\theta) = \rho
\] (3.1)

where \(\rho\) represents the distance from the origin to the line along a vector perpendicular to the line and \(\theta\) is the angle between the x-axis and the perpendicular vector as illustrated in Figure 3.4;

Figure 3.4. Representation of a straight line in polar coordinates.

By using this transform, a line can be represented by a single point in the polar-coordinate parameter space. Similarly, since infinite lines pass through any given pixel in the original image, the representation of a pixel in the parameter space is a unique sinusoidal curve (representing all the lines that can pass through that pixel), as shown in Figure 3.5. The point of intersection between multiple sinusoidal curves in the parameter space represents the line passing through all the pixels. Therefore the more the intersection points, the more pixels a line passes through. The parameter
space is then divided into bins in the $\rho$ and $\theta$ space. The total number of intersections in each bin is then saved into the accumulator, and the highest voted lines are returned [46].

![Hough space of an example image frame.](image)

Figure 3.5. Hough space of an example image frame.

In order to consider a number of collinear points as a line, the accumulator threshold has been set adaptively as 75\% of the mean size of the image. In addition, only a maximum number of 30 lines with the higher accumulator values that result from the Hough transform are considered in order to save computation time.

After acquiring the parameters of the lines detected in the image, the following filtering procedure is conducted in order to discard lines that cannot be considered for the vanishing point estimation. Since the camera is mounted at a certain height, we assume that the edges of the path cannot appear as horizontal lines. In addition we assume that the robot remains in the path, and therefore there can be no vertical
lines coming from the edges of the path. Thus we filter lines that deviate by 5 and 10 degrees from the vertical and horizontal axes, respectively.

3.1.2.5 Vanishing Point Estimation

The result from the line localization step consists of a set of lines in the image at a specific slope and distance to the image center. We will assume hereafter that most of the lines in the image will pass near the vanishing point. Under this assumption, we can use RANSAC (RANdom SAmple and Consensus) [41] in order to randomly sample the above set of lines and find an estimate of the point of intersection (i.e., the vanishing point) which has the highest consensus. RANSAC is a non-deterministic iterative method for estimating the parameters of a model that best fits the given data, while ignoring outliers contained in the data due to noise or erroneous measurements. RANSAC can be used to estimate the vanishing points and corresponding lines using the process below:

1. The intersection of two randomly selected lines detected by the Hough transform is calculated by solving for \([x, y]^T\) in the following equation

\[
\begin{bmatrix}
\cos (\theta_1) & \sin (\theta_1) \\
\cos (\theta_2) & \sin (\theta_2)
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix} =
\begin{bmatrix}
\rho_1 \\
\rho_2
\end{bmatrix}
\]  
(3.2)

where \(\rho\) and \(\theta\) correspond to the parameters of the intersecting lines.

2. After the coordinates \((x, y)\) of the intersection point are computed, we calculate the distance \(d\) of each line to this point using

\[
d_i = x\cos (\theta_i) + y\sin (\theta_i) - \rho_i
\]  
(3.3)

where lines with a distance below a certain threshold \(T_d\) are considered inliers. The inlier-set represents the consensus for the vanishing point calculation.
3. If the consensus set is larger than a certain threshold $T_c$, then we consider these inliers to be a good estimate and the algorithm stops, returning the estimated intersection point. Otherwise, we repeat the previous steps until either a good inlier set is found or a maximum number of iterations is reached.

In addition to obtaining the coordinates of the vanishing point for every frame, we store the coordinates of the last 5 frames and return the median of these coordinates. This acts as a buffering mechanism since it suppresses radical changes in the coordinates of the vanishing point due to incorrect estimation in specific frames. The $x$ coordinate of this filtered vanishing point is then used to calculate the deviation from the principal point of the camera. In our implementation we assume that the principal point is in the middle of the image and that the camera points parallel to the axis of motion of the robot. The deviation, which takes values from 0 to 255 from far left to far right, is then sent to the controller module using UDP datagrams.

![Figure 3.6. Single frame example of vanishing point detection.](image)

(a) Original scene image  
(b) Processed image with Hough lines, inliers, and estimated vanishing point
An example image acquired by the system is shown in in Figure 3.6(a). In Figure 3.6(b) we can see the image edges with the lines detected after applying the Hough transform. Despite the large number of lines, RANSAC has picked the actual path edges as the inliers (amber lines). The red circle represents the vanishing point estimated for the current frame, the light blue circle is the buffered vanishing point after taking into account the last 5 estimations and the green circle is the simple least squares solution. The horizontal red line represents the deviation from the center of the camera and is equal to the x-coordinate of the blue circle. This value is sent to the controller for the steering command calculation.

3.1.2.6 Laser Daemon

The laser daemon is designed to serve as the software interface to the LIDAR unit. Measurements from the Hokuyo URG-04-LX-01 ranging unit are acquired using the SKIP 2.0 USB protocol for Hokuyo ranging devices. The URG-04-LX-01 returns readings from an envelope of 240 degrees with 0.36 degree resolution. Each ranging measurement is achieved with a resolution of 1mm, and is assembled into a UDP packet for transmission to the system controller.

Once received by the system controller (at a rate of 10 Hz), the UDP packet is used to populate an array of measurements. These measurements are then filtered using either a moving average or moving median filter, both with adjustable window sizes. After filtering, a safety envelope is scanned for potential obstacles. Laser measurements falling within this safety area are counted, and if this number exceeds a user defined threshold, the laser signal overrides the vision signal as the source of error for the PID controller.
3.1.2.7 System Controller

As a central component of the iDog, our system controller is responsible for fusing real-time data streams from sensing components. This module performs the selection and filtering of sensor streams in order to generate the control law error signal. At each time step, filtered error signals are used as input to the controller. The System Control GUI, in addition to conditioning data streams and providing user interface settings, performs discrete iterations of the PID controller. This common controller is used to calculate the turning radius command that is sent to the iRobot platform. The error $e(t)$ is defined as the deviation of the vehicle from the desired path at time step $t$. The vehicle turning radius commanded at time $t$ is then given by

$$
\text{Vehicle turning radius} = K_p e(t) + K_i \int e(t) dt + K_d \frac{de(t)}{dt}
$$

The control gains $K_p$, $K_i$, and $K_d$ are configurable in the System Control GUI, along with the integration window parameter $\hat{t}$. This allows the user to tune and to evaluate every aspect of the radius controller.

In order to avoid collision with static and moving obstacles, the system controller switches the error signal source from the vision daemon to the laser daemon when obstacles are detected within the safety envelope. The centroid of all the scanning points lying within the safety envelope is then used to set $e(t)$ to its upper or lower bound value, such that the platform spins in place away from the obstacle centroid. This behavior is repeated until the safety envelope of the robot is determined to be clear.

3.1.2.8 Robot Daemon

The iRobot daemon receives command packets from the system controller. These command packets are validated and transformed into iRobot Create OI com-
mands, which are sent to the platform through a configurable serial port [48]. In addition to serving as control relay point, the daemon allows the user to override any UDP command with the keyboard arrows. This feature provides an additional debugging utility for the user.

3.1.3 Results

In this section we present the results of some experiments conducted to validate the effectiveness of the proposed design. We tested the iDog system in various scenarios in order to evaluate its robustness to changes in illumination and obstacle avoidance. During our tests we came up with practical modifications that increased both the accuracy and the performance of our system. The first was to restrict the coordinates of the initial intersection point of the pair of lines to the middle 1/3 of the image vertically and inside the boundaries of the image horizontally. This is based on the assumptions that the path cannot have an extreme inclination and that the camera always keeps the path in its view. This made the vanishing point estimation more robust, since even when a large number of false lines were detected, since most of them intersected at arbitrary points, they were not considered in the execution of RANSAC. It also resulted in a small execution time benefit, since RANSAC was executed selectively. The second modification was to process only the lower half of the image in outdoor navigation, thus reducing drastically the processing time and the occurrence of false lines coming from trees and other objects. The intersection point of the pair of lines was restricted to the upper half of the image.

The highest navigation accuracy was achieved indoors since there was an increased number of perspective lines coming from the walls and the ceiling. This resulted in accurate vanishing point estimation. Outdoor performance was lower due to the fact that collinear edge pixels coming from objects or noise in the image were
detected as lines, introducing inaccuracies in the execution of RANSAC. The collision avoidance system proved to operate robustly in all the environments.

Finally, the user interacts with the robot through a GUI, shown in 3.7, from which he can switch the system on and off using two large and distinct buttons. In addition the interface allows for the operating parameters of the robot to be tuned for better performance and for matching the robots operation to the users needs. The GUI also provides a visualization, shown in 3.8, of the captured video with the estimated vanishing point as well as the direction of the robot and any obstacles detected.

![Figure 3.7. Multiple tabs of the iDog System Controller Application.](image-url)
Our iDog guide robot proved to be an effective assistive system for the visually impaired. It can successfully navigate a given path and can avoid obstacles. Additionally, it is cost effective and can be easily built since it uses low cost, off-the-shelf parts.

3.2 Omniwheel-based Holonomic Assistive Robotic Platforms

Holonomic platforms, such as the the manipulation platform shown in Figure 3.9, are of particular interest to the field of assistive robotics due to their ability to move in any direction with any orientation. This can be extremely useful for mobile manipulation and transportation within human environments.

Most holonomic platforms require very flat surfaces to operate reliably, and thus are not realistic for outdoor environments. In this section, we present a robust omnidirectional robotic platform for outdoor operation on non-smooth surfaces. The design of an off-road, low-cost omnidirectional robotic platform is presented along with a suspension system.
that allows the platform to traverse rough terrain [49]. We also provide a control architecture based on the open-source Robotic Operating System (ROS).

3.2.1 Omniwheel Design

The omniwheel utilized by our platform is designed to provide traction on rough surfaces when moving in the direction of wheel rotation, while providing near frictionless casting in directions along the axis of the drive shaft (i.e., the wheel should freely slide sideways). To facilitate this, we designed an assembly that utilizes commercially available 100 millimeter polyurethane wheels that are commonly found on recreational scooters. These wheels were chosen due to their relatively low-cost, inte-
grated bearings for smooth rotations, and durability in outdoor environments. Each of the 16 wheels are mounted on either side of a piece of aluminum extrusion using a single carriage bolt. The resulting 8 wheel pairs are bolted to a central aluminum disc along with a wheel hub. The wheel hub can be changed to accommodate a wide range of motors and drive shafts.

The wheel design, shown in Figure 3.10, has an overall diameter of 15.0 inches. This diameter supports the arrangement of 16 wheels such that the gaps between wheels are minimized in order to provide the smoothest operation possible. Other omniwheel designs often employ barrel shaped orbital wheels in multiple layers to eliminate this gap, though this approach does not work well on rough surfaces and is generally difficult to fabricate. Our design can be fabricated with simple machine
shop equipment, such as a hand-held hacksaw and power drill. The fully assembled wheel module is shown in Figure 3.11. The wheel assembly is driven by brushed DC motors which can be salvaged from discarded electric wheelchairs.

3.2.2 Platform Suspension and Layout

In order for omniwheel platforms to move successfully, each drive wheel must maintain contact with the ground when actuated. This design consideration is commonly overlooked in most approaches, since it is generally assumed that the surface of the operating environment is both flat and smooth. Our approach uses an independent vertical suspension system for each drive module. Each motor and wheel pair is allowed to slide upward along a pair of vertical rails against a spring. The spring applies a constant force in the downward direction, such that the drive module is normally pressed against the vehicle chassis when the wheel is on flat ground. When the platform is on rough terrain, the wheel modules will move upward toward the
springs, which absorbs the shock and causes the wheel to maintain contact with the ground. The rail and spring drive suspension assembly is shown in Figure 3.12(a).

Our platform is designed to utilize 4 drive modules mounted in a square configuration. Each drive module is positioned on one side of a chassis base with a 90 degree offset from the adjacent motors. When mounted on a 24.0 inch square chassis base, the resulting platform is 37.0 inches in length and width. The arrangement of the drive assemblies and suspension rails on the chassis base is shown in Figure 3.12(b). The fully assembled mechanical platform is shown in Figure 3.13.

3.2.3 Control Software

Our system utilizes the open-source Robot Operating System [50] software to facilitate code modularity and reduce development time. Each core software process is implemented as a ROS node, which compiles and executes independently and pub-
lishes or subscribes to ROS topics. The resulting architecture, shown in Figure 3.14, shows the minimum set of software processed needed to control the platform using the methods discussed in related work. New nodes can be added to accommodate additional hardware without affecting existing nodes, which will be necessary as the platform is adapted to various applications. Our architecture uses standard ROS topics and messages, which are shown in Figure 3.14. The control software is currently under development, and will be uploaded to the public ROS repositories once complete.

3.2.4 Results

Testing of the platform was performed on both a smooth indoor surface and on mixed outdoor terrain. The outdoor terrain consisted of grass, tightly packed
Figure 3.14. ROS software control architecture.

dirt, and rough patches resulting from light soil erosion. The platform was controlled manually using a USB joystick attached to a netbook running ROS. In all cases, the vehicle maintained full control authority over all degrees of motion freedom (2D position and heading). While the omniwheels produce some vibration, the vertical suspension system absorbs the majority of the mechanical disturbances before they are transmitted to the payload area containing the electronics. Figure 3.15 shows the platform during testing in an area consisting of mixed grass and dirt.

3.3 A Development Platform for Non-Tactile Wheelchair Controls

This section presents an intelligent wheelchair designed to be used as a development and evaluation platform for alternative, non-tactile power wheelchair controls
The system is designed to be highly modular such that new human-computer interface devices and methods can be quickly integrated and evaluated as necessary. The current configuration provides full proportional steering and speed control outputs using a combination of voice commands, video occulography (eye tracking), and a single point electrode based electroencephalography (EEG) brain-computer interface.

3.3.1 System Hardware

The system platform is built upon a modified Pride Jazzy 614 HD electric wheelchair. This model was chosen due to its popularity in the rehabilitation community and balance of performance and affordability. The existing motor controller was replaced with a Roboteq HDC2450 dual H-bridge motor driver in order to provide an interface to the control computer. The platform is also equipped with an Asus Xtion Pro RGB-D sensor for localization and mapping with the operational environ-
ment, and an embedded Intel Atom based motherboard which is powered directly from the existing wheelchair batteries. A laptop computer can also be used when a display is needed during development. Both options provide enough computing resources to support processing of 2D and 3D data provided by the RGB-D sensor and other input devices.

The Pupil Tracking subsystem, consisting of a single camera, is used to provide video occulography data for pupil tracking. A Playstation Eye camera was modified in order to provide infrared images, and mounted to a lightweight sunglass frames using the approach presented in the EyeWriter project [5]. The camera is able to provide 640x480 video frames at a rate of 60 Hz using a standard USB 2.0 interface.

Audio data for the Speech Recognition subsystem is provided by an integrated microphone on the Playstation Eye camera, which is conveniently located close to the mouth of the user when mounted on the sunglass frames.

The BCI subsystem is provided by a Neurosky Mindwave headset. The low-cost EEG device provides brainwave data from a single electrode placed over the forehead. The assembled eye tracking headset with integrated microphone, along with the BCI device is shown in Figures 3.16 and 3.17.

3.3.2 System Software

Our system utilizes the open-source Robot Operating System [50] software to facilitate code modularity and reduce development time. Each core software process is implemented as a ROS node, which compiles and executes independently and publishes or subscribes to ROS topics. The resulting architecture, shown in Figure 3.18, allows interoperability and rapid prototyping of nodes. New nodes can be added to accommodate additional hardware without affecting existing nodes. In our architecture, we designate an independent node for the Speech Recognition, Pupil Tracking,
and BCI input modalities. We process these modalities in the Data Fusion node, which generates motion commands that are interpreted by the wheelchair platform driver.

To facilitate control of the wheelchair using voice commands we implemented a simple dialogue system, that receives input from the eye tracking headset’s built-in microphone, translates it into text and reacts accordingly. To translate the user’s utterance into text we used the Pocketsphinx recognizer [52] that provides very robust automatic speech recognition. The vocabulary of our system comprises 16 words that represent the available voice commands: *ahead, back, backward, fast, faster, forward, full, go, halt, move, reverse, slow, slower, speed, stop, and wait*. To create the
language model and dictionary, necessary for the recognizer to work, we used the sphinx language model tool. In order to identify commands in the user’s utterance, we developed a simple parser that matches words in the user’s utterance with words in our vocabulary. The identified commands are then passed on to the dialogue manager, which is responsible for processing the commands and sending the appropriate output which will then be used to control the motors of the wheelchair. That output contains information about direction, speed and whether a stop command has been issued. The resulting dialogue system is also able to disambiguate commands such as “go forward and backward” or “move slower and go full speed”. In favor of safety, the command that yields the slowest speed prevails and a stop/halt command overrides everything. When a command is detected, a message containing the command label is published in the form of a ROS string message.

In order to generate proportional steering commands in a continuous manner, we developed a ROS wrapper for the open-source pupil tracking algorithm presented in [25] which reports higher tracking accuracy than the methods presented in [4] and [53]. This node estimates and publishes the location of an ellipse which is fit to the pupil in the eye camera image frame. This estimate is performed on each of the acquired video frames, which are received at a rate of 30 Hz. A pupil ellipse “home position” is maintained by the sensor fusion node, which is averaged over several frames when the node is initialized. This can be reset at any time in a manner similar to a calibration routine utilized by most eye tracking approaches. The pupil center estimate is published in the form of a custom ROS “occulography” message with each processed frame in real-time.

The Neurosky Mindwave neuroheadset is a simple, inexpensive single electrode EEG device. The circuitry of the device provides two scalar indicators, attention and meditation at a rate of 1Hz. It also provides raw delta, theta, alpha, beta and gamma
brain wave data at the rate of 500Hz. Our goal was to extract a binary event based on the intention of the wheelchair user to begin a motion command. The data from the sensor contains a large amount of noise, and a simple thresholding of the attention and meditation indicators are not sufficient to detect user intent.

It was determined that a calibration step was needed for each user of the device. A simple logistic regression based classifier was implemented using the meditation and attention indicators provided by the headset. During the training process, the user is instructed to alternate between relaxation and concentrating intently on “moving the wheelchair with their mind” for 5 second intervals for a total of 30 seconds. The resulting decision boundary takes the following form of

\[ x = C_1 \ast attention + C_2 \ast meditation + C_3 \]

Where values of \( C_1 \), \( C_2 \), and \( C_3 \) are given by the training process which is performed prior to each session of wheelchair use. An example with computed values of \( C_1 = 0.042055 \), \( C_2 = 0.038797 \), and \( C_3 = -5.206809 \) is shown in Figure 3.19.

Combination of the input data modalities is performed by the data fusion node, which handles the mapping between continuous (eye tracking) and event-based modalities (speech recognition and BCI). In our current configuration, the event-based modalities are used to control the starting and stopping of motion, while the continuous modalities are used to control proportional steering. The mixing of inputs results in a velocity and steering ROS message, which is processed by the safety override node. The safety override node provides a final check for emergency conditions, such as obstacle detection, communication timeout, or activation of the e-stop switch.
Figure 3.17. Intelligent wheelchair platform with user.
Figure 3.18. System software architecture and communication flow.

Figure 3.19. Example calculation of BCI decision boundary.
CHAPTER 4
APPLICATIONS

In this chapter, real-world applications made possible by the preceding hardware and software contributions are discussed. First, we present methods for aggregating and visualizing 3D PoG information using visual saliency heat maps. Second, an approach for object of interest detection and recognition is discussed which utilizes several modalities provided by the 3D PoG headset. Finally, the use of 3D PoG for wheelchair control and robot-assisted object manipulation is presented.

4.1 3D Mapping of Visual Saliency

Visual salience, in a physiological sense, refers to the tendency of certain areas in a perceived image to stand out from neighboring areas. This results in the “attention grabbing” effect, where visual fixation is immediately attracted to certain image areas. Similarly, in computer vision, salience refers to the tendency for certain image areas to contain more distinguishing features. This can be useful for multimedia databases, since image areas that exhibit more salience can be used for efficient retrieval. While physiologic saliency and computer vision saliency are independent of each other, there is some commonality. Image salience detection methods based on natural behavior exist [54], and many of the methods that are not explicitly based on nature often present similar results.

Related work in eye tracking has shown that the salience of an image can be inferred by monitoring the point of gaze [55, 56, 57]. This is often represented as a salience “heat map”, an example of which is shown in Figure 4.1.
While these methods all assume static 2D scenes, the same could be done in 3D environments given the proper hardware. It follows that this “heat map” concept could be applied to 3D scene reconstructions provided by RGB-D data. This could be useful for a variety of applications, such as visual attention studies, automatic eye tracker user calibration, and gaze-based robot assisted object manipulation.

In the case of eye tracker calibration, the general process for most methods involves recording the pupil positions of a user while they direct their gaze toward several known locations. A mapping is then generated which transforms pupil locations to either 2D PoG locations or a full 3D gaze vectors as we have discussed in previous sections. With statistical knowledge of the relative salience of various points in the scene, it follows that a continuous calibration adjustment could be applied while the device is in use. For example, knowing that certain locations in a scene will be focused on more than others, it follows that calibration points can be matched in a probabilistic manner over time. This may be effective especially when an ini-
tial sub-optimal calibration mapping is provided from a previous manual calibration routine.

In order for assistive robotic devices to successfully manipulate objects for a user, the robot must be able to comprehend the desirability of an object along with its physical bounds and coordinates in 3D space. We address the location and bounding aspects of this problem using 3D object segmentation and clustering, but we must also provide a means by which gaze vectors are translated into manipulation commands. This is not only a function of 3D space, but also of time, as a momentary glance at an object does not necessarily equate to a desire for the object to be displaced or visually identified.

We address these challenges by applying the concept of heat maps to a real-time 3D environmental scan provided by our headset. Given a gaze vector with some bounded error (provided by the calibration routine), we compute a virtual “heat value” that describes the relative visual salience of a particular point as a user directs their gaze through a scene. We model the application of visual “heat” using concepts of physical heating and cooling, which provides an intuitive means for visualizing and reasoning about visual salience over a given time window. Our approach is able to account for gaze vector error, visual fixation duration, and shifting attention through tunable variables that can be adjusted online.

In order to apply heat to a given location of interest, we first must compute the distance of the point to the gaze vector. This can be solved using the 3D point-line distance formula [58]

\[
D = \frac{|(X_0 - X_1) + (X_0 - X_2)|}{|X_2 - X_1|} \tag{4.1}
\]
where $X_1$ and $X_2$ are two unique points lying on a line in 3D space and $X_0$ denotes a given point of interest. We then wish to assign a saliency heat value in the range of $[0, 1]$ with a gaussian function \[ f(x) = ae^{-\frac{(x-b)^2}{2c^2}} + d \] (4.2)

where $a$, $b$, $c$, $d$ set the curve’s peak, horizontal offset from 0, bell width, and vertical offset from 0, respectively. Our goal is to assign a heat value from $[0, 1]$ centered around a distance of 0 to the gaze vector, thus we set $a = 1$, $b = 0$, $d = 0$. The bell width, $c$, we leave as a free parameter which is set according to the error of the gaze vector. In our headset approach, we are able to estimate this parameter given the known hardware and user calibration errors. It then follows that we assign heat to a given point $i$ using the following formula

\[ H_i = e^{-\frac{(D)^2}{2c^2}} \] (4.3)

where $D$ is computed using (4.1). Combining (4.1) and (4.3) to achieve a final heat formula for a point $i$ given the gaze vector defined by points $X_1$ and $X_2$ with error constant $c$ yields

\[ H_i = e^{-\left(\frac{|(X_0-X_1)+(X_0-X_2)|}{|X_2-X_1|}\right)^2/2c^2} \] (4.4)

In real-time applications, we wish to decrease the perceived saliency of objects or points over time if the user’s attention shifts toward another location. We also want to decrease the contribution that short duration gaze vectors have on our perception of saliency, such as those measured during saccade eye motion. Intuitively, we model this “cooling” of 3D saliency map locations using basic heat transfer equations.
Recalling Newton’s Law of Cooling [60], the temperature of a cooling object at time \( t \) is given by the function

\[
T(t) = T_e + (T_0 - T_e)e^{-kt}
\]  

(4.5)

where \( T_e \) is the temperature of the surrounding environment and \( T_0 \) is the initial temperature of the object. The constant \( k \) controls the rate of cooling based on the material properties of the given object. In our application, we define the temperature of a point with no relative saliency to be equal to zero, thus we set \( T_e \) to zero. We then rewrite Eq (4.5) as

\[
T_i' = T_i e^{-kt}
\]  

(4.6)

where \( T_i \) is the initial saliency temperature of point \( i \) and \( T_i' \) is the updated temperature of the point after cooling for \( t \) seconds. Equations (4.6) and (4.4) are then used to generate dynamic 3D visual salience heat maps as shown in Figure 4.2.

4.2 Object of Interest Detection and Recognition

This section describes a computational approach used for object of interest identification and classification using the 3D PoG headset [23]. The four steps of the process are to estimate the PoG using the eye and scene cameras, assign a geometric classification based on the 3D object of interest structure, perform visual classification using SURF feature matching and color histograms, and fuse the multimodal data for a final result. The subsections below describe the classification steps in detail, along with experimental results.
Figure 4.2. Comparison of 3D scene with overlayed visual salience heat map.
4.2.1 Geometric Classification

Point cloud manipulation is performed with the utilization of the Point Cloud Library (PCL) [27]. PCL provides the methods necessary to extract information from point clouds, the contribution presented in this section is the overall process for which the given methods are applied.

Instead of applying the model segmentation on the initial point cloud, a series of operations must be performed on the point cloud to remove points that are not of interest. A large portion of the point cloud is comprised of these points, which include the points that correspond to the floor, wall, or ceiling, and the points that lie outside the area of interest. We can assume these points are not of interests due to the fact that points of interest must provide interactivity and lie within a reasonable distance to the user’s PoG.

Planar models are quicker to detect than more complex models, such as cylinders or spheres, so it is beneficial to remove large planes from the point cloud prior to detecting the models belonging to the more interactive geometries. Planes corresponding to tables, walls, the ceiling or floor, will span a large portion of the point cloud. Due to this it will not be necessary to perform the planar segmentation on the full point cloud, and down sampling of the point cloud can be performed. This will provide a performance increase since the fidelity of the point cloud is reduced, while allowing large models to maintain their structure within the point cloud. The removal of these large planes from the point cloud is useful in reducing the point cloud size, as these will not provide valuable interaction for the user.

Objects that are of interest are comprised of several points that are relatively close together and are not disjoint. PCL provides a method to detect the euclidean clusters within a point cloud. These clusters are found by linking points together that are within a defined distance threshold, which further emphasizes the importance
of removing large planes, since they will connect clusters that otherwise would be disjoint. After the clusters are identified, the PoG is combined with the point cloud to determine the cluster closest to the PoG. This cluster is extracted from the point cloud.

The extracted cluster provides a region of interest within the original point-cloud, and the final model segmentation is performed on the subset of points from the initial point cloud that lie inside the area of the extracted cluster region. When segmenting smaller objects, higher fidelity is needed with the point cloud, which is why the region must be taken from the original high-fidelity point cloud. When model segmentation is performed on this final point cloud, cylinder and sphere models are detected. Model parameter estimation is done using the RANSAC algorithm [61]. This model parameter estimation is also done in similar fashion when estimating the planar coefficients discussed previously. Final model classification is assigned based on the results of the segmentation over each of the specified models. The currently available geometric classifications belong to the set \{cylinder, sphere, other\}.

Following the geometric classification, analysis is performed on the RGB data to further classify the object. The input for these methods consists of the geometric classification and a cropped 2D RGB image representing the final extracted point cloud. The cropped image comes from creating a bounding box relative to the 2D RGB image of the region of interest containing the extracted cluster.

Figure 4.3 illustrates the differences in the point clouds throughout the process as described above.

4.2.2 SURF Feature Matching

In order to reliably identify a query object by image comparison, there needs to be similarity between image features. Since it is unlikely that the object being
identified will be in the same orientation and position relative to the reference image, it is important to calculate features that are reproducible at different scales and viewing angles. Speeded Up Robust Features (SURF) is an efficient method to find such features, called keypoints, and calculate their descriptors, which contain information about the grayscale pixel intensity distribution around the keypoints[62].

The system maintains a knowledge base of SURF features and descriptors for all reference object images. For these images, the keypoints and descriptors are precomputed and stored to avoid recalculation each time an object is to be identified. The feature/descriptor calculations for the query object images, on the other hand, are necessarily performed on-the-fly as object identifications are requested.

In the SURF-based object identification we perform, the query object image keypoints are compared to those of each reference object image to determine simi-
larity. We use a modified version of the robust feature matching approach described in [63] to do so. A k-nearest-neighbors search is performed to match each keypoint descriptor in the query image with the two most similar descriptors in the reference image, and vice versa. These matches enter a series of tests to narrow down the list of those that are accepted. First, if the two nearest-neighbor matches are too similar to reliably determine which is the better match, neither is used. Otherwise, the best match is tentatively accepted. Figure 4.4 shows several keypoint matches at this stage. Second, if a keypoint matching from the query image to the reference image is not also a match from the reference image to the query image, it is rejected. The surviving keypoint matches are validated using the epipolar constraint so that any matched points not lying on corresponding epipolar lines are rejected, and the number of remaining matches is stored for each image in the knowledge base.

4.2.3 Histogram Matching

Since multiple objects can produce similar features in SURF calculations, it helps to incorporate color information into object identification. We use color histograms to do so, since they provide a convenient way to represent the distribution of
colors in an image and can easily and efficiently be compared. To minimize the effect
on histogram matching of potential differences in brightness and contrast between
reference and query images, a normalized red-green (RG) color space is used for the
calculations.

The histograms we use contain 8 bins in each dimension. So, for the normalized
RG color space, we use 2-dimensional 8 × 8 histograms for a total of 64 bins. As
with the SURF keypoints/descriptors, the histograms for the reference object images
are computed and stored in the knowledge base for easy comparison later, while the
histograms for the test images are calculated at identification time. To identify a query
object by histogram matching, the similarity between the query image histogram and
each reference image histogram is calculated using normalized cross-correlation to
obtain a value in the range [-1, 1].

4.2.4 Data Fusion and Identification

To most reliably identify the object of interest, we must effectively incor-
porate the data from SURF feature matching, geometric classification, and histogram
comparison into a single score for each object in the reference set.

After SURF keypoint match calculations, the number of keypoints matched
from the query object image to each reference object image is stored as a raw score,
n for that particular reference object. A final, normalized SURF score \( \alpha \in [0,1] \) is
calculated for each reference object \( i \):

\[
\alpha_i = \frac{n_i}{m}, \quad \text{for } m = \max_i (n_i)
\]
Similarly, normalized cross-correlation values obtained from the histogram comparisons are stored for each reference object image as a raw histogram score, \( h \in [-1, 1] \). A final normalized histogram score \( \beta \in [-1, 1] \) is calculated for each object \( i \):

\[
\beta_i = \frac{h_i}{k}, \quad \text{for } k = \max_i (h_i)
\]

The third score we calculate is a simple geometric classification match score \( \gamma_i \) for each reference object image \( i \). To determine \( \gamma_i \), the query image’s detected classification \( c \) is compared to the reference classification \( d_i \):

\[
\gamma_i = \begin{cases} 
1 : c = d_i \\
0 : c \neq d_i 
\end{cases}
\]

A final score \( S_i \) is calculated for each object \( i \) as a linear combination of the three scores. To do so, the SURF, histogram, and geometric scores are assigned weights, \( w_\alpha, w_\beta, \) and \( w_\gamma \), respectively:

\[
S_i = w_\alpha \alpha_i + w_\beta \beta_i + w_\gamma \gamma_i
\]

The object \( O \) can now be identified as:

\[
O = \text{argmax}(S_i)
\]

4.2.5 Experimental Setup

To assess the ability of the system to identify the object gazed upon by the user, we created an experiment to reproduce a typical usage application in which the user is seated at a table and desires assistance with an item on the table. The user might, for example, desire some water from a pitcher on the table, but be unable to reach for the object or request assistance through verbal means or gesturing.

To this end, we used the system software to create a knowledge base of known objects and placed an assortment of test items on the table to evaluate the system’s
ability to estimate the user’s point of gaze, use that information to isolate the object of interest, and perform successful identification. An example usage scenario can be seen in figure 4.2.5.

During our experiment, a participant sat in multiple positions in front of a table with an assortment of objects placed on top. They were free to move their head, eyes, and body. We instructed the participant to focus their gaze on an object and notify us with a verbal cue when this was accomplished. On this cue, a trigger event for the system to identify the object was issued. The PoG calibration was performed prior to system use, and the calibration result was checked for validity. In the experiment, the participant focused his gaze on each of the objects from three different locations at distances of up to 2 meters.

The knowledge base used for image comparison and identification consisted of fifteen objects that varied in size from a baseball to a musical keyboard. Each object had two previously collected training images from different angles and distances,
Table 4.1: Object identification results

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SURF Matching</td>
<td>0.711</td>
</tr>
<tr>
<td>Histogram Matching</td>
<td>0.622</td>
</tr>
<tr>
<td>SURF + Histograms</td>
<td>0.756</td>
</tr>
<tr>
<td>SURF + Histograms + Geometry</td>
<td>0.844</td>
</tr>
</tbody>
</table>

which had been obtained using the same headset and automatically cropped via the method described above.

4.2.6 Results

After running the experiments, the raw scores of the image comparisons were processed to determine the optimal values for the three classification weights. Once the score weights were adjusted, the results were collected and analyzed. Table 4.2.6 shows the object identification accuracy for the various classifiers in the system, both individually and in combination.

As can be seen from the results, the ability to identify the object of a user’s gaze significantly improves as additional classifiers are added. Since SURF feature matching is a popularly used method of object matching, we use its accuracy as a baseline for our analysis. We see a significant 18.7% increase in correct object identifications by incorporating color histogram and geometric classification data with SURF matching. These results clearly illustrate the benefit of fusing multiple data modalities. The average execution times, in seconds, for each step in the identification method are presented in Table 4.2.6.
4.3 3D PoG Wheelchair Control

The gaze controlled wheelchair presented in [23] uses pupil positions to issue steering commands to the wheelchair. While this is useful, it is less intuitive than issuing a direct location command. Using the 3D PoG hardware presented in [10], it is possible to infer this desired location by computing the intersection of the gaze vector with the floor plane or other surface. By adding obstacle detection similar to [64], common robot motion planners such as A* or Wavefront Propagation will be able to guide the robot safely to the desired position. This will allow the user to freely move their gaze immediately after issuing the motion command, rather than continuously looking in a particular direction to steer the platform.

4.4 Assisted Object Manipulation

Extending upon the object of interest identification and detection capabilities of the 3D PoG headset [23], position and pose detection of objects in the relative to the user can be relayed to an assistive manipulation platform.

In the previous work on object of interest identification and detection using the 3D PoG headset [23], a subset of points belonging to the surface of an object is selected from a larger point cloud representation of the environment. Once the object of interest is isolated, it is easy to estimate its position and pose. Knowing this

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Execution time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometric Classification</td>
<td>0.329</td>
</tr>
<tr>
<td>SURF Matching</td>
<td>0.201</td>
</tr>
<tr>
<td>Histogram Matching</td>
<td>0.001</td>
</tr>
</tbody>
</table>
Figure 4.6. PR2 robot manipulating tabletop objects.

information makes it possible to issue manipulation commands to assistive platforms, such as the PR2 shown in Figure 4.6.

The PR2 is an ideal manipulation platform with dual compliant arms. The platform is designed to be safe to work in environments where humans are present. Using the perception capabilities of the PR2, along with network connectivity, information from the 3D PoG headset can be shared in order to execute manipulation commands. When the field of view of the human user and the area of perception of the PR2 share some overlap, registration techniques can be used to compute the position and orientation of the two observation points relative to each other [65]. Once this is achieved, identifying the object of interest to the human in the PR2 frame of reference and performing a manipulation task is possible. Extending upon the work
presented in [66], we plan to demonstrate object of interest manipulation based on gaze point estimation.
CHAPTER 5

DISCUSSION

This work has presented past, present, and future contributions to the field of gaze based control for assistive robotic platforms. Multiple hardware and software solutions have been presented with the intention of moving toward utilization of low-cost, 3D PoG detection as a primary human-computer interface modality. In this chapter, open problems, future work, and conclusions are discussed.

5.1 Open Problems

The current implementation of the 3D PoG headset is somewhat cumbersome, and relies on an eye camera that is within the user’s field of view. In order for the device to be as minimally invasive as possible, it should have a physical size and shape similar to common eye wear while retaining it’s sensing and processing capabilities. This can be broken down into 3 sub problems: miniaturization of the RGB-D scanner, relocation of the obstructing eye camera, and miniaturization of the CPU. At the time of this publication, commercially available RGB-D scanners are somewhat bulky for this application, but versions small enough for our purpose are expected to be released very soon. The eye camera can be moved to a non-vision obstructing location using a clever arrangement of mirrors, though this would require considerable mechanical design efforts. The CPU requirements of the headset were too demanding for commercially available mobile processors, but this has recently changed and new solutions are currently being evaluated.
Another open problem is 3D gaze tracking is automatic calibration. Currently, all calibration methods for both 2D and 3D eye trackers drift with time due to several factors (motion, fatigue, etc). It is likely possible that automatic calibration adjustments could be performed over time to compensate for drift using feedback control. Previously, eye trackers had no method of feedback since it was impossible to acquire salience knowledge of the environment without physically defined calibration points. Knowledge of environmental salience, coupled with statistical data regarding eye motion toward salient features could be used to apply these corrections, which might make a full calibration routine robust to errors imposed by drift.

Finally, knowledge of user eye motion can be used as a predictor for various situations of interest. Nystagmus, for example, is a type of involuntary eye movement that occurs when a user is under the influence of drugs or alcohol, having a reaction to medication, or suffering from brain injury. Other types of eye motion are similarly indicative of other conditions, where detection could be useful in an assistive setting. Detection of these sort of events has been hampered by a lack of data, since observations by a processional are required. Pervasive mobile eye tracking will support future work in this area, and may lead to useful automatic methods.

5.2 Future Work

The 3D PoG headset presented in this work, together with the related mathematical and algorithmic frameworks, has shown great potential for use in assistive scenarios. In future work, we plan on miniaturizing the system components and deploying our software on mobile, low-power processors. This will allow the system to be worn by the user in a more discrete fashion, which will be an important factor for future commercialization and product development efforts.
While we have solved the necessary coordinate transformations and calibration challenges for estimating 3D gaze vectors, there remains much work to be done for actual use by robotic platforms. Fortunately, the approach provides useful functionality as a standalone system, and we believe that there is an immediate need for the device as a communication aid. This will allow us to continue to improve the usability and ergonomic factors of the device during near-term development, while building upon lessons learned for future robotic applications.

Additionally, we intend to investigate more appropriate calibration models to increase the accuracy of the system. Some existing work in the augmented reality community suggests that it may be possible to measure the position of the eye in the oculography camera frame automatically without assigning a manually measured translation. This would eliminate the final manually obtained parameter in our mathematical framework, and thus increase overall accuracy and user customization.

5.3 Concluding Remarks

This work is intended to lead to commercialization of the 3D PoG headset. The initial prototype is promising and the remaining work items are currently being developed. The applications mentioned in this work are meant to stimulate interest in 3D PoG as a primary control interface for people with severe physical disabilities. Such a device could facilitate communication, transportation, and independence for people who are not currently able to perform these tasks themselves due to technological limitations. In order to promote commercialization, a large emphasis has been placed on cost-effectiveness throughout the design and development process of the device. The device has already attracted a fair amount of commercial interest, and work is expected to continue into the future.
REFERENCES


http://portal.acm.org/citation.cfm?doid=1117309.1117350


102


BIOGRAPHICAL STATEMENT

Christopher D. McMurrrough was born in Arlington, Texas in 1984. He received his B.S., M.S., and Ph.D. degrees in Computer Engineering from The University of Texas at Arlington in 2008, 2010, and 2013, respectively. He has worked on unmanned systems and micro air vehicles with The Automation and Robotics Research Institute (ARRI) and the Air Force Research Laboratory (AFRL) from 2008 to 2010. In 2010, he joined the Heracleia Human Centered Computing Laboratory and worked on assistive technologies while also collaborating with The University of Texas at Arlington Research Institute (UTARI). His current research interest is eye tracking and assistive robotic devices.