January 2019

Multirobot Confidence And Behavior Modeling: An Evaluation Of Telerobotic Performance And Efficiency

Nathan Lucas
Wayne State University

MULTIROBOT CONFIDENCE AND BEHAVIOR MODELING: AN EVALUATION OF TELEROBOTIC PERFORMANCE AND EFFICIENCY

by

NATHAN PAUL LUCAS

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2019

MAJOR: COMPUTER ENGINEERING

Approved By:

_____

Advisor Date

_____

_____

_____

DEDICATION

To my parents, wife, and children

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Abhilash Pandya, who has provided much help, encouragement, and guidance over the years, and been a role model of mentorship. This work would not have been possible without his enthusiastic support and dedication. I am also grateful to my dissertation committee, Dr. Mohamad Hassoun, Dr. Le Yi Wang, and Dr. Darin Ellis, for their valuable advice, feedback, and professionalism.

Thank you to Dr. CJ Chung for introducing me to robotics, and to Dr. Lisa Anneberg for supporting the start of my journey in robotics research. Thanks to Darol Straub for kickstarting many young engineers, computer scientists, and physicists at a pivotal time in their life.

Many thanks to my colleagues and friends at Wayne State, especially Dr. Sam Lee, whose work was the foundation for this research, and Dr. Tony Composto and Dr. Prem Sivakumar for their ideas, feedback, and conversation during many coffee and lunch breaks. Special thanks to Dr. Luke Reisner for thoughtful advice and time spent helping me prepare for presentations. Thank you to Mostafa Rahbar and Parinaz Farajiparvar for enduring practice presentations and providing valuable feedback, and to all the other members of the CARES lab, past and present.

Thanks to all my friends for your patience and support during this endeavor. My gratitude to the research study participants who volunteered their time and contributed data that were instrumental in completing this work.

Last but not least, I would like to thank my family. Thank you to my parents, who taught me the value of life-long learning and instilled the values that give my life meaning. I am forever grateful for your patience and wise counsel. Most importantly, innumerable thanks to my wife Katherine for her unwavering support and encouragement, and our children Helena, Katarzyna, Natalia, and John. You bring joy to life and provide endless inspiration.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION AND MOTIVATION

1.1 Motivation

Advancements in mobile robot technology have enabled a surge in unmanned vehicle development and deployment in a variety of operational environments. Unmanned aerial vehicles (UAV) and unmanned ground vehicles (UGV) have enjoyed considerable technology and market growth, from aerial surveillance to robotic floor vacuums to improvised explosive device (IED) interrogation. Robots have long been viewed as ideal for dull, dirty, and dangerous work, but there are many other potential benefits. In addition to carrying human occupants, manned vehicles include human-machine interface (HMI) controls, heating and air conditioning systems, seats, restraints, transparent windows, structural reinforcements for safety and increased weight capacity, and additional fuel. Without the burden of these requirements, unmanned vehicles can be smaller, lighter, and more efficient. Unmanned systems can also operate free from the constraints of human physiological limits, like acceleration tolerance, physical endurance, and fatigue.

Although great progress has been achieved, mobile robots still face technological impediments to autonomous navigation in cluttered and dynamic environments, and safe operation in proximity to humans. Despite these persistent challenges, the increasing prevalence of unmanned systems and ongoing research are opening new possibilities for multiple robots operating in concert to overcome some of the limitations of independent unmanned vehicles.

A multirobot system is a team of robots operating with some level of coordination, ranging from naïve collective behavior to complex collaboration [1-5]. This coordination enables multirobot systems to perform a wide variety of distributed, hazardous, and complex tasks that are difficult or impossible for independent robots. For example, a team of robots might work

together to move an object too massive for one robot alone, or search a large space in far less time by simultaneously exploring multiple areas. Research toward multirobot systems includes learning and control algorithms [6-8], architectures [9], collaborative localization [10], area exploration [11], and heterogeneous robot cooperation [12]. For a review of taxonomy and research trends see [1], [3] and [4].

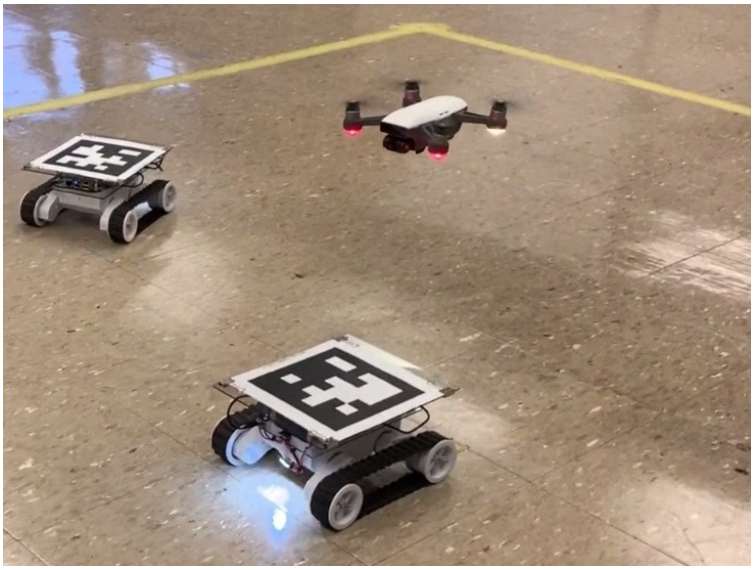

Figure 1-1: A heterogeneous robot team can leverage the unique strengths of each platform.

There is considerable interest in multirobot systems for a number of defense, security, and space applications [5]. Specific research has focused on space exploration due to the hostile conditions and spatial distribution involved, including work toward robotic construction and assembly of structures in orbital, lunar, and planetary environments [13-18]. Most of these use cases are also applicable to terrestrial settings. See [19] for a survey of literature related to space applications. Other common tasks for which multirobot research has focused include search and rescue [20-23] (or more generally foraging [24]), disaster response [25], [26], surveillance [27], and robotic assembly for manufacturing [28]. All of the above applications also have dual-use potential within the defense and security domains.

Multiple robots can perform spatially distributed tasks, provide redundancy in hazardous environments, or operate as a team of specialized machines to complete complex tasks together. Unmanned aerial, ground, surface, and submarine platforms now work beside and take the place of humans in extremely dangerous situations on the battlefield, like interrogation of suspected road-side bombs and aerial logistics resupply missions in austere environments. Robots also perform material handling in commercial warehouses. The advent of military and material handling robots has stimulated visions of manned-unmanned teams, with humans and intelligent machines completing complex tasks and missions by working together. Operational concepts include teams of aerial and ground robots performing autonomous resupply and reconnaissance missions in direct support of military personnel in the field [29], and lunar multirobot excavation [13].

In order to fully realize the benefits of coordinated ground and aerial robots in complex environments for various domains [5], [13], [30-35], techniques are needed to help manage the complexity of interacting with multiple mobile robots [31], [36], [37].

1.2 Research Aim

The aim of this research is to develop techniques incorporating operator attention as input for teleoperation interfaces in order to enable effective and efficient control of multiple mobile robots. This aim is motivated by the need to overcome the limitations of human perception and cognition affecting the ability of operators to integrate information from multiple sources, switch between multiple spatial frames of reference, and divide attention among many sensory inputs and command outputs. Robot autonomy is necessary to help the operator manage increasing demands as the number of robots scales up; however, more automation does not necessarily equate to better performance.

1.3 Research Objectives

This research was divided into three specific objectives. Figure 1-2 provides an overview of these objectives. The first objective was to equip a human-operated multirobot test platform with the ability to estimate operator attention in real time. Eye gaze tracking provides a means of measuring physiological properties related to human cognition. Incorporating this technology in the platform enabled the development and evaluation of novel techniques applicable to user interface designs for the remote operation of multiple unmanned vehicles.

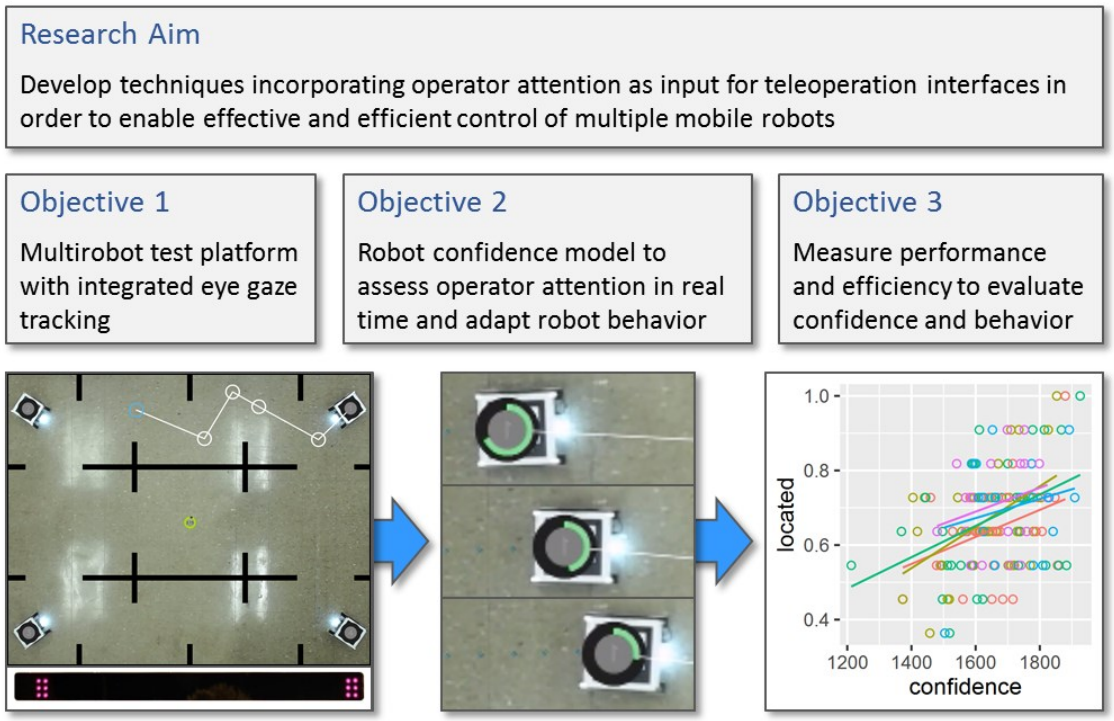

Figure 1-2: Research aim and specific objectives.

The second objective was to enable system feedback based on operator attention that can be used to mitigate challenges related to how operators use automation in the context of multiple unmanned vehicles. Specifically, this research developed and implemented a model of robot confidence that transformed attention-related inputs to adaptive robot behaviors.

The third objective of this research was to evaluate task performance and efficiency in

relation to robot confidence and adaptive behaviors. A series of user studies was conducted to assess alternative approaches to confidence-based robot behavior.

1.4 Dissertation Outline

Chapter 1: Introduction and Motivation outlines specific opportunities to advance multirobot teleoperation by incorporating real-time assessment of operator attention and adaptive robot behaviors in response to attention. Chapter 2: Background provides relevant information about multirobot autonomy, augmented reality, measuring teleoperation performance, eye gaze tracking, and robot confidence. Chapter 3: Analysis of Autonomy for Multirobot Search Tasks describes a user study conducted to measure multirobot teleoperation task performance at three levels of robot autonomy, and opportunities to mitigate overreliance on autonomy by assessing and responding to operator attention in real time. Chapter 4: Multirobot Platform with Eye Tracking covers the design and functionality of a multirobot test platform developed to implement techniques that employ operator attention as system feedback and facilitate user studies evaluating these techniques. Chapter 5: Robot Confidence and Task Performance introduces a robot confidence model which was used to adapt robot behavior in response to operator attention, and details three user studies conducted to evaluate telerobotic task performance and efficiency in relation to a number of adaptive behaviors. Chapter 6: Discussion and Future Work summarizes the specific contributions of this research and opportunities for future work incorporating and expanding upon the results.

CHAPTER 2: BACKGROUND

2.1 Overview

This chapter provides general background relevant to this research. Section 2.2 highlights fundamental concepts related to multirobot autonomy which are relevant to the entirety of this work, most specifically Chapter 3. Augmented reality as applied to human-robot interaction (HRI), briefly reviewed in Section 2.3, is relevant to Chapter 3 and Chapter 4. The methods of measuring manned-unmanned system performance surveyed in Section 2.4 provide context for Chapter 4 and Chapter 5. Certain physiological measures also relate to Section 2.5, which contains information on eye gaze tracking relevant to Chapter 4 and Chapter 5. Finally, Section 2.6 provides an overview of concepts related to confidence in the field of robotics, which is relevant to the work presented Chapter 5.

2.2 Multirobot Autonomy

Certain task requirements, technological limitations, and environmental conditions will necessitate human interaction with applied multirobot systems. Specific actions expected to require human intervention include approving targets and resolving navigational impasses. In addition, manned-unmanned teams will benefit from the unique advantages of human cognition, reasoning, ingenuity, and soft skills for the foreseeable future. Ethics and morality in particular may ultimately remain an exclusively human function.

Teleoperation of multiple mobile robots involves information from many sources, multiple frames of reference, and competing tasks. Factors affecting single robot control via video-based interfaces include restricted field of view, difficulty ascertaining orientations of the environment and robot, unnatural and occluded viewpoints, limited depth information, and poor video quality [38]. Increasing the number of robots teleoperated multiplies these challenges, with each robot having potentially unique and dynamic orientations, camera perspectives, and

sensory frames of reference. The demands of multitasking can overload the operator and limit the scalability of HRI as the number of robots increases [39-43].

Span of control, also called fan-out [41], is the maximum number of robots that can be simultaneously operated or supervised by a single human [40], [42-44]. Equation 2-1 shows fan-out as defined by [39], where NT is neglect time and IT is interaction time. Neglect time is the expected time duration a robot can be ignored before performance drops below a defined threshold. Interaction time is the expected time duration necessary for the operator to interact with the robot in order to restore it to maximum performance.

$$\mathrm{Fanout} = \frac{\mathrm{NT}}{\mathrm{IT}} + 1 \tag{2-1}$$

General approaches to address operator overload due to multi-tasking include redesigning tasks and interfaces to reduce demands, training operators to develop automaticity and improve attention management, and automating tasks and task management [45]. Specific areas of research toward multirobot teleoperation and autonomy include task switching and the allocation of operator attention [21], such as methods of identify where an operator should focus and using this information to influence the operator's behavior via visual cues in a graphical user interface [46]. Other research includes determining which aspects of a given task are most suitable for automation [24], measuring and influencing operator trust in team autonomy [27], using intelligent agents to help human operators manage a team of multiple robots [20], and augmented reality interfaces to integrate information from multiple sources and project it into a view of the real world using a common frame of reference [31], [47], [48].

2.3 Augmented Reality for Human-Robot Interfaces

Augmented reality (AR) is the registration and visual integration of computer-generated graphics and real-world environments [49], [50]. Techniques can be categorized as optical

blending, such as head-mounted displays, and video blending with graphics overlaid on video frames displayed on a screen. Telerobotic systems very often rely on real-time video from the perspective of, or external to the robot. One challenge of teleoperation is limited visuospatial perspective. AR techniques such as color-coded orientation cues that visually map controller input axes to end effector axes can improve telemanipulator navigation, with significant reductions in trajectory distance, deviations from the ideal path, and navigation error [51].

AR can also reduce visual search and mental integration demands. During traditional neuronavigation, a surgeon must mentally transform two-dimensional medical imaging data into three-dimensional structures, and project this information on her or his view of the patient. Systems for augmented neuronavigation can perform transformations by computer and display composite video with models of structures of interest projected on the surgical site, resulting in significantly lower task time and fewer errors [52].

Human control of multiple mobile robots requires considerable divided attention, the integration of information from many sources, and switching between multiple frames of reference. Projecting sensed data onto the real-world scene, at the point of observation or at the point being observed, may help alleviate the cognitive burden of mentally integrating information from various sources. Demonstrated techniques include overlaying sensed data onto individual robots via wearable head-up display [47] and superimposing arrows on 20 robots to create a gradient toward a target location [48].

2.4 Manned-Unmanned Systems Performance

Many metrics have been proposed for evaluating the performance of human-robot teams. See [53] for a review emphasizing task-oriented mobile robots, and [54] focusing on multiple remotely operated robots. This section reviews a few areas of performance relevant to the research presented in subsequent chapters.

2.4.1 Task Performance

Quantitative performance metrics include task completion time, percentage of tasks successfully completed, and count of errors. Total mission or task time can be decomposed into various types of activities (e.g., navigation, target identification, failure condition). The duration of specific activities or the allocation of time can be evaluated and compared [37]. Time allocations are expressed in units of time, percentage of task time, or a ratio (e.g., ratio of robot time to operator time) [53].

2.4.2 Situation Awareness

Situation awareness (SA) is critical to decision making and human performance in complex and dynamic environments. SA is related to attention, but the two are distinct constructs. SA in dynamic decision making can be described as the perception of elements in the environment, comprehension of the current situation, and projection of future status [55]. Situation assessment is the process of achieving, acquiring, or maintaining SA.

The Situation Awareness Global Assessment Technique (SAGAT) [56], [57] was developed to assess operator SA across a spectrum of requirements. The principal prerequisite is a detailed analysis of SA requirements for which a battery of corresponding queries is created. SAGAT is administered by interrupting simulation trials at randomly timed and unpredictable freeze points, and presenting queries randomly to the subject.

Physiological measurement techniques such as electroencephalography (EEG) can be less intrusive than tools like SAGAT, but most cannot determine memory retention, level of comprehension, and other cognitive processes important to SA [57]. Eye tracking can provide indications that information has visual focus, but does not capture elements of SA related to peripheral vision and cognition.

2.4.3 Operator Workload

Workload can be thought of as the cost to achieve task requirements or a certain level of

performance. It is influenced by a variety of factors linked to task conditions, human perception, and cognition. These complexities make workload difficult to define and measure. Proposed measures include a number of subjective assessment tools that rely on user responses to workload related queries, such as the Air Traffic Workload Input Technique (ATWIT) [58]. See [59] for a comparison of three such instrument: NASA Task Load Index (TLX), Subjective Workload Assessment Technique (SWAT), and Workload Profile (WP) instruments.

NASA TLX [60-62] is a commonly used multidimensional subjective workload assessment prevalent in human-machine systems research. As seen in Equation 2-2, TLX computes a global score of perceived workload w_{TLX} as the weighted sum of user ratings for six factors, generically s_f, and corresponding weights w_f. The factors—mental demand (MD), physical demand (PD), temporal demand (TD), performance (PF), effort (EF), and frustration (FR)—reflect the many aspects of workload.

$$w_{\mathrm{TLX}} = \frac{\sum_f (w_f s_f)}{\sum_f (w_f)}, \text{where } f \in \{\mathrm{MD}, \mathrm{PD}, \mathrm{TD}, \mathrm{PF}, \mathrm{EF}, \mathrm{FL}\} \tag{2-2}$$

There are several objective physiological responses to workload [63], including heart period, heart rate, blood pressure, respiratory cycle time, blink interval, blink duration, and pupil dilation. Applied metrics include blood pressure, heart rate, electroencephalography (EEG) response, eye blink rate via electrooculography (EOG), and task-evoked pupillary response (see [64] for a review). Physiological methods are capable of providing dynamic, objective measures of workload, but are generally more intrusive and cumbersome than subjective assessment techniques.

2.5 Eye Tracking for Human-Robot Interaction

2.5.1 Introduction

Eye tracking technology has its origin in nineteenth-century studies of basic eye movements.

Applied research beginning in the mid twentieth century investigated the role of eye movements and visual attention in reading, scene perception, visual search, and other information processing (see [65] for past and contemporary applications). Non-invasive techniques like video-based eye tracking (head-mounted or remote) [66], [67], and improvements in automatic fixation and saccade detection [68], have inspired applications for human-computer interaction (HCI) (see [69] for a review).

A fixation is a relatively stable visual gaze at an area of interest. A saccade is a rapid ballistic movement to a new area of interest. The detection of fixations and saccades from raw eye movement data was a principal focus of early work toward eye tracking for HCI [70-75]. See [68] for a review of fixation identification techniques. Eye tracking devices are capable of accurately measuring the point in space at which a human is looking, but the resulting raw gaze points are inherently noisy. Algorithms to identify fixations and saccades must account for jitter and other non-saccadic eye movements.

Several algorithms have been developed for identifying fixations and saccades, as compared by [68]. Of these, the velocity threshold (VT) method is the simplest to implement and requires the lowest computational overhead, but is least robust. This method exploits the relatively low velocity of fixational movements (<100°/second) and high velocity of saccadic movements (>300°/second) to filter gaze points. The algorithm first calculates point-to-point velocity. A point below a certain velocity threshold is labeled a fixation point. Otherwise, the point is labeled a saccade point. The VT algorithm then collapses consecutive fixation points into a fixation group and removes saccade points. The centroid of the fixation group is computed to represent the fixation as $\langle x, y, t, d \rangle$, with coordinates x and y, at time t of the first point, and duration d from the first point to last point in the group.

The dispersion threshold (DT) method provides more accurate and robust fixation

identification as compared to VT, but with an increase in overhead. DT identifies a fixation as a group of consecutive points within a particular dispersion, or maximum separation. A minimum duration threshold of 100-200 milliseconds is typical, with a dispersion threshold set to include 0.5° to 1° of visual angle based on distance from the eye to the screen. The algorithm uses a moving window that spans consecutive points, initialized to cover points within the duration threshold. The dispersion D of the points in the window is determined by summing the range of x and y values as in Equation 2-3:

$$D = [\max(x) - \min(x)] + [\max(y) - \min(y)] \quad (2\text{-}3)$$

For an initialized window, if D is greater than the dispersion threshold the first point is removed and the window is advanced by one point. Otherwise, the DT algorithm adds points to the fixation group until D exceeds the threshold. Upon exceeding the dispersion threshold, a fixation is noted and the points within the window are removed. Like VT, a fixation is defined at the centroid of the points.

Considerable research has been conducted toward the application of eye tracking for HCI [65], [69], including analytical applications to derive indicators of cognitive processes and interactive applications to use eye tracking as an input device. Improvements in eye tracking technology and the recent availability of low-cost systems present new opportunities to improve measurements of interface performance and develop novel interface techniques.

2.5.2 Analytical Eye Tracking

The human eye is a unique window into the processes of perception and cognition. Researchers have used eye tracking to study reading [76], [77] and other information processing tasks [78]. There is ongoing interest in analyzing eye movement data to measure attention, situation awareness, workload, and fatigue.

2.5.2.1 Attention

HCI research has long sought to exploit eye gaze as a proxy for attention [79-81]. [82] modeled dynamic operator overload based on the operator's attention to a critical situation associated with impending failure. The response time before initial fixation represented delayed attention. The number of fixations on an object represented allocation of attention.

Fixation has been applied as a measure of attention allocation for an online predictive model of operator overload during supervisory control of multiple UAVs within a simulation environment [83]. A logistic regression model, developed to predict vehicle damage when an operator failed to correct a collision course, was applied to generate real-time alerts. The model was a function of the delay prior to allocating visual attention to the vehicle, how much attention was diverted away from the vehicle once attended, and how much time remained before the collision will occur.

2.5.2.2 Situation Awareness

Eye gaze fixations have also been used to measure SA. [84] measured fixations to examine SA reacquisition after brief task-related breaks during supervisory control of multiple simulated UAVs. Fixations were categorized as either a re-fixation on an object that had been previously fixated, or a novel fixation on an object not previously fixated. Task conditions requiring SA reacquisition were associated with faster fixations on more objects and more re-fixations. SA preservation was associated with slower fixations on fewer objects and more novel fixations.

[85] developed a hazard prediction model with three fixation-based predictors of SA. A logistic regression was calculated from data collected during simulations with five semi-autonomous, homogeneous UAVs. The model predicted UAV damage from hazard events the operator failed to prevent, presumably the result of insufficient situation awareness.

2.5.2.3 Workload

Pupil dilation is a reliable and sensitive indicator of mental processing load [64], [86]. A task-evoked pupillary response to subtask workload during HCI has been observed, although average response might not accurately reflect peak periods of workload throughout a task [87]. Video-based techniques for measuring task-evoked response have been demonstrated using a remote eye tracker [88], which is less invasive than head-mounted systems. Video pupillometry is non-intrusive, but may be impractical for applications unless all light sources are well controlled. The pupillary light reflex is evoked by changes in ambient lighting conditions and the brightness of objects within view (e.g., graphics displayed on a screen).

Blinks may also be an indicator of cognitive workload. [63] observed increased blink interval and decreased blink duration associated with higher visual load, and decreased blink interval associated with higher memory loads. Blink duration did not appear affected by memory load.

Techniques using eye-gaze to assess workload are difficult to apply in real-world environments where ambient lighting and other factors affecting human vision are not highly controlled. [89] evaluated workload during simulated air traffic control. Workload was first assessed using ATWIT [58] and correlated with aircraft density in the simulation. Eye movement was recorded with between 2 to 9 aircraft, varied to manipulate workload. Shorter blink durations, decreased saccade distance, and increase pupil diameter were observed as the number of aircraft increased. These results provide evidence that eye gaze can be applied to assess workload and other measures of operator perception and cognition in real-world applications.

2.5.2.4 Fatigue

Eye gaze has been researched for measuring operator fatigue. [90] varied time-on-task (TOT) and task complexity (TC) to manipulate mental fatigue during controlled visual search. The

peak velocity-magnitude slopes of saccades and microsaccades decreased, and mean fixational drift velocity increased, with increased TOT. The results indicate saccade and microsaccade dynamics, and drift velocity are all affected by mental fatigue during prolonged visual search. However, fixational and saccadic eye movements were not significantly affected by TC.

2.5.3 Interactive Eye Tracking

Interactive eye tracking can be used for hands-free user input for the disabled [91], predicting a vehicle driver's intent [92-94], gaze interaction for automobiles [95], and automatic camera viewpoint for robotic and laparoscopic surgery [96]. Eye gaze pointing and camera control are two areas of interest that have potential for a broad range of HCI.

2.5.3.1 Pointing

Gaze-directed pointing is the archetype of interactive eye tracking, and a core motivation for fundamental work like fixation and saccade detection [68]. Spatial input is a highly intuitive use of eye tracking for interactive systems, and considerable research has been conducted to pursue gaze-based pointing [70], [71], [79], [80], [91] (see [69] and [65] for reviews). Overt attention directed at a user interface element is a strong indicator of the user's intent to interact with that element. A simple implementation of a gaze pointer might allow point-and-click operations to be performed by just looking at object. This approach facilitates very fast user input, but it does not distinguish passive viewing from active input. Thus, the challenge is determining a suitable method of selecting objects and actions.

Research has assessed the application of eye tracking to windows, icons, menus, and pointer (WIMP) style interactions, and a variety of selection techniques to address specific use cases [71]. Specific interface object selection methods include dwell time, fixation of a separate large on-screen button, and a hardware button [79]. Eye gaze with 150 millisecond dwell time selection has been observed to be faster than a computer mouse [81]. Other demonstrated

techniques include a zoom method of character selection for hands-free gaze-based typing [97]. The suitability of techniques depends on the type of interaction, including the level of effort necessary to recover from unintentional commands:

- Immediate [71] – for focus, highlight, and non-intrusive information display
- Dwell time [71], [81] – for low to moderate speed actions, and commands that are relatively easy to undo
- Zoom [97] – for letters, numbers, words, or icons in a familiar layout
- Hardware key or button [79] – for high speed or repetitive actions

These techniques can be combined with a dialog box or other command confirmation method if an unintentional command risks user frustration or potentially unrecoverable action (e.g., deleting data).

2.5.3.2 Camera Control

Gaze-based remote camera directional and zoom control is another intuitive application of eye tracking. A user may wish to change camera perspective or zoom level in response to observed actions or objects in the scene. Overt attention, as detected by eye tracking, might indicate such intent.

Techniques have been developed to teleoperate a robot using eye gaze, including an interface to both drive a robot and change the view of an on-board camera [98], [99]. The user interface featured graphical overlays for control elements. Gaze input commands were activated by either dwell-time or a foot clutch, enabling hands-free teleoperation.

Techniques have been developed for gaze-based automatic camera pan and tilt control. These include a simple proportional control algorithm to generate pan and tilt velocity commands for continuously repositioning the camera viewpoint to bring the user's point of gaze to the center of the screen [100]. Eye gaze has also been applied to controlling multiple pan-tilt-

zoom (PTZ) cameras. [101] used eye gaze to control nine PTZ cameras. Video from the cameras was shown in separate preview windows, all on one screen. A fixated window gained focus and was enlarged after a dwell time 3 seconds. Camera pan and tilt were controlled via fixations near the edge of the selected window. The window could be deselected by looking outside it for 2 seconds, after which the window decreased back to the preview size.

Figure 2-1 depicts a systems developed to automatically center laparoscopic camera viewpoint at the user's point of gaze [102]. Like [100], the system responded only to eye movement and did not adapt to the task being performed. Intent prediction [72] and more sophisticated interaction schemes [97] using similar eye movement data hint at the possibility of more robust gaze-based automated camera systems.

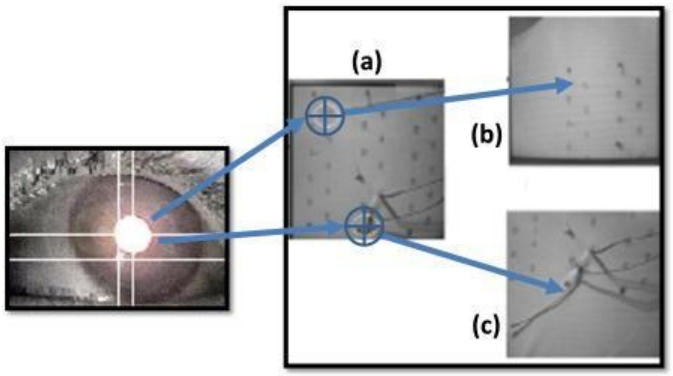

Figure 2-1: Diagram depicting a robotic camera arm used to center a viewpoint based on eye gaze tracking [102].
If the user fixated a point in (a) at the top of the image, the center of the viewpoint shifted to (b). If a point is fixated at the bottom, the center of the viewpoint shifted to (c).

[72] developed an offline method for predicting a user's intended camera zoom level (i.e., magnification and reduction), and reported an accuracy of 65% predicting zooming in, zooming out, or no zoom change for simple shapes in a controlled interface environment. The algorithm relied on previously recorded eye movement data and was not capable of determining zoom intent in real time. Despite these limitations, advances in automatic identification of fixations

and saccades [68] might enable adaptions for dynamic interactive applications.

[103] developed a zooming windowing technique designed specifically for eye-based control. The eye-based typing interface by [97] used zoom and planar transformation of characters. Although these examples were not targeted toward video camera viewpoint control, each demonstrates zoom techniques that may be applicable.

The above examples of gaze-based camera directional and zoom control demonstrate how viewpoint can be automatically adjusted in response to where in the image the user is looking. The control algorithms are simple and do not account for how task conditions and scene dynamics may influence optimal viewpoint. Nevertheless, interactive eye tracking is a feasible and intuitive approach to camera control.

2.6 Confidence Relevant to Robotics

Concepts related to confidence are often linked to human trust in autonomy and allocation of control, or how a human operator uses available levels of autonomy. A distinction can be drawn between *operator* confidence and *robot* confidence. Operator confidence typically refers to the self-assurance of a human in their own ability to perform a task, or trust in a robot's ability to perform autonomously. Research includes the impact of transparency and reliability on operator confidence [104]. Models estimating human self-confidence have been developed for purposes such as automatically choosing between manual and autonomous control [105].

Research related to robot confidence is typically aimed at altering human trust in autonomy or allocating control authority. A common objective is convincing the operator to shift the allocation of control toward autonomy or manual operation as appropriate to optimize performance. For example, a robot may provide visual feedback indicating its self-confidence in order to influence the operator's trust [106]. Alternatively, a model of robot confidence might be used to directly distribute authority, such as setting shared-controller gains to amplify or

attenuate inputs from a teleoperator and ultrasonic sensors [107].

Other research includes a robot expressing its certainty in performing policy learned from a human teacher [108-110], and modeling a robot's confidence in a human co-worker [111] or its ability to predict human actions in a shared environment [112]. A similar concept is algorithm self-confidence, applied for example to a visual classification algorithm [113].

CHAPTER 3: ANALYSIS OF AUTONOMY FOR MULTIROBOT SEARCH TASKS

3.1 Introduction

This chapter describes a user study (Study 1: Levels of Autonomy) conducted to measure multirobot teleoperation task performance at three levels of robot autonomy, and key findings of this study which raised new questions and led to the development of a multirobot test platform to examine these and other questions toward the overall research aim. The platform detailed in Chapter 4 addressed lessons learned from the study and incorporated robot confidence derived from operator attention to facilitate subsequent studies presented in Chapter 5.

3.2 Motivation

Automation is necessary for a human operator to effectively control multiple robots. Research has often focused on how many robots can be operated [42] and methods to do so efficiency [39], [43]. The user study discussed in this chapter was conducted to evaluate task performance at three levels of robot autonomy using an initial test platform with four small wheeled robots [31], [36]. The development and design details of this platform were published by [36]. The study measured the time it took participants to complete certain tasks at three levels of autonomy. With a fixed number of robots, successively higher levels of robot autonomy were expected to improve performance. However, the study yielded a surprising result which led to further research and additional user studies.

3.3 Methods

3.3.1 Test Platform Overview

Figure 3-1 shows the test platform and the control interface window displayed by the test platform software. A camera was positioned above an approximately 7.43 m^2 (80 ft^2) test area, 3.048 m (10 ft) wide by 2.438 m (8 ft) tall, with the optical axis orthogonal to the group plane.

The camera delivered 1280×1024 px video at 10 fps to the control interface. The control interface software application used the ARToolKit augment reality (AR) library [114-117] to detect a unique fiducial marker attached to each robot and compute homographies relating robot poses to pixel coordinates in the application window. Precise localization of each robot enabled the control interface to plan and valid navigation paths, and project AR graphics into the real scene at or near current robot locations. The operator inputted teleoperation commands using a computer mouse or joystick, and reported task completion using a computer keyboard.

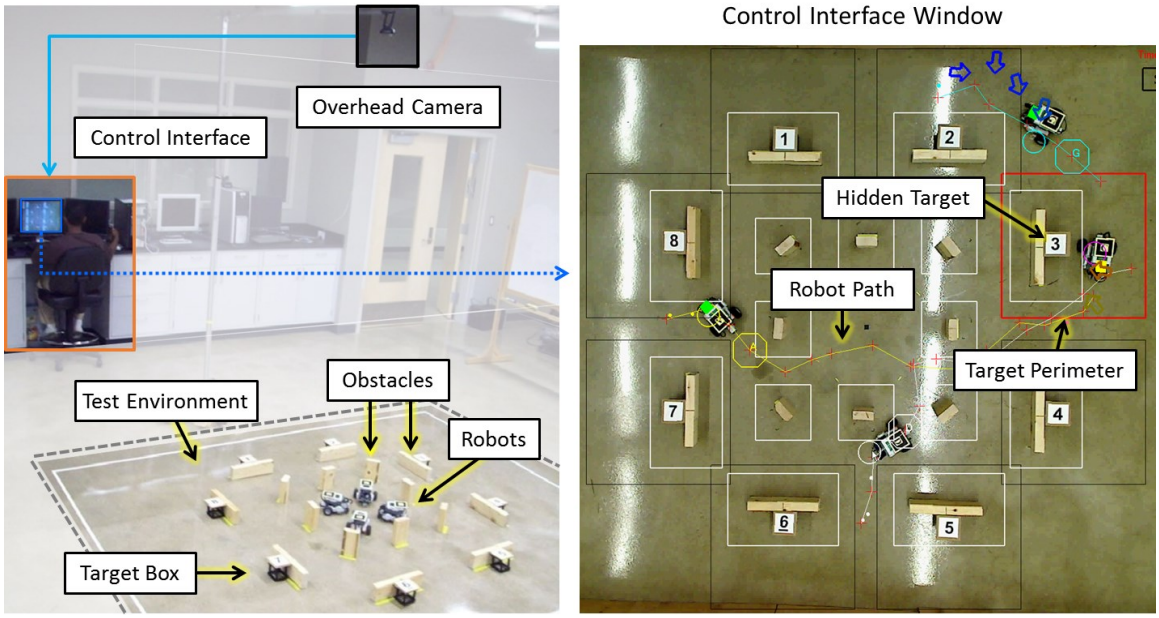

Figure 3-1: The Levels of Autonomy test platform and augmented-reality (AR) interface. Left: An overhead camera supplied video of the test environment. The control interface processed and displayed video frames with overlay graphics. The operator used the resulting AR view to control and receive feedback from multiple robots. Right: The control interface window drew graphics for robot status, navigation paths, and sensor readings. The interface also overlaid graphics related to task completion, such as a red target perimeter line.

3.3.2 Robots and Test Environment

Figure 3-2 shows the robots in the test environment and a diagram of the platform as seen from above, matching the perspective shown by the control interface application. The robots

were equiped with one drive wheel on each side and a free caster at the rear to enable differential steering. They communicated with the control interface via Bluetooth to receive motion commands and send sensor data. A forward facing sensor detected the strength and direction of infrared signals within a 240 degree arc parallel to the ground plane. The robots sent sensor data to the control interface, which rendered color-coded arrows near the respective robot to indicate the sensed infrared signal strength and direction. A dot graphic was displayed if no signal was detect. These graphics persisted in the scene at the point of measurement until the user inputed a new navigation command to the robot.

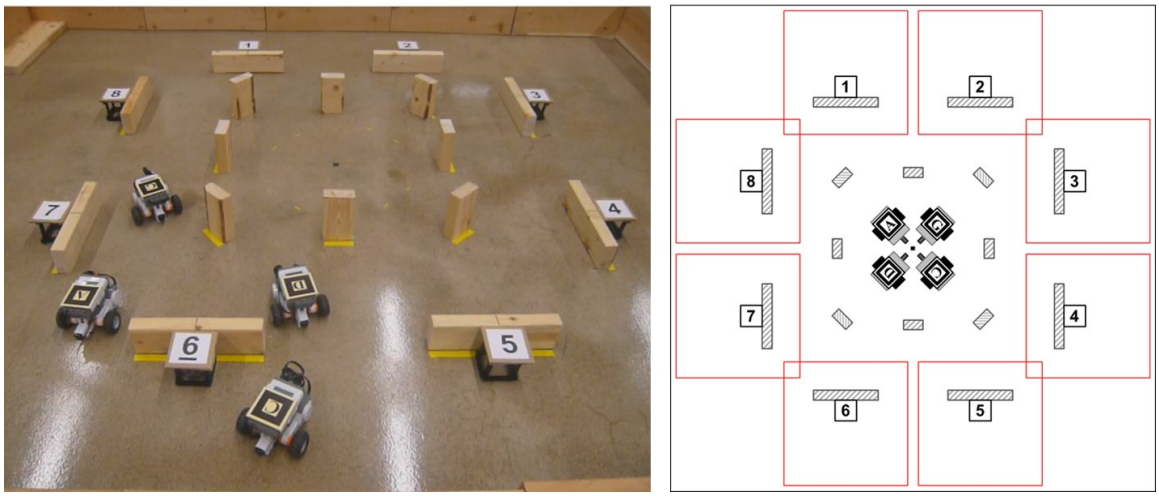

Figure 3-2: Robots and test environment used for the Levels of Autonomy study. Left: For each trial, an infrared beacon was randomly hidden in one of the numbered target boxes. 16 solid obstacles blocked the path and sensor line-of-sight of the robots. Right: This diagram illustrates the red perimeter drawn on the screen around located target.

The test environment contained 16 solid obstacles and 8 numbered target boxes. The robots had to be navigated around the obstacles, which blocked their path and sensor line-of-sight. For each study trial, an infrared beacon was placed in one of the boxes to serve as a hidden target. Participants were instructed to locate which box contained the hidden target. Upon the participant reporting the target location, the control interface drew a red rectangular perimeter line around box (see the illustration on the right in Figure 3-2). The participant then moved all

four robots to within the perimeter in order to complete the trial.

3.3.3 Levels of Autonomy

The robots operated at one of three autonomy levels listed in Table 3-1. The highest level, automatic path generation (*path*), used the A* algorithm [118] to plan a collision-free path around obstacles to the inputted goal point. The robot then automatically followed the planned path until reaching the goal. The intent of path autonomy, along with the AR graphics displayed on the screen, was to allow the operator to focus on higher-level tasks.

The middle autonomy level, automatic vertex generation (*vertex*), planned a single-vertex path to the inputted goal, rejecting goals that resulted in obstacle collisions. Similar to path autonomy, the robot automatically followed the single-vertex path until reaching the goal. In essence, vertex autonomy relied on the operator to plan the overall path, and automatically executed only one path waypoint at time.

The lowest level of autonomy provided no automation (*none*). Instead the operator manually controlled each robot one-at-a-time using a joystick.

Table 3-1: Levels of Autonomy

Level	Input device	Operator input	Automation
path	mouse	goal node coordinates	generate and execute multi-vertex path around obstacles
vertex	mouse	goal node coordinates	generate and execute single-vertex path to goal; reject goals resulting in collisions
none	joystick	forward, reverse, pivot clockwise or counterclockwise	none; robots manually controlled by operator

Levels listed highest-to-lowest autonomy

3.3.4 Procedure

Study participants used the control interface to search for an infrared beacon hidden randomly inside one of eight numbered target boxes. Participants were instructed to determine

which target box contained the beacon, report the box number by pressing the corresponding key on the keyboard, and navigate all four robots to within the red perimeter line drawn on the screen around the target. Each participant completed three practice trials, one for each level of autonomy (path, vertex, and none; see Table 3-1), to familiarize with the controls and search task. The participant then completed nine study trials, three at each autonomy level. The order of the levels was randomized.

3.4 Results

Eighteen individuals from the student and faculty bodies of Wayne State University volunteered to participate in the study. This and all subsequent studies were conducted in accordance with the applicable Human Investigation Committee (HIC) approval found in Appendix B. Each participant completed three trials, one per level of autonomy, all within the same session. The dependent variables were the *search time* to locate the target and the *mission time* to converge on the target (i.e., bring all four robots within the target perimeter).

Figure 3-3 shows the apparent relationship observed between level of autonomy and task performance. As expected, enabling some degree of autonomy appears to have resulted in better average performance than no autonomy. Without automatic path following participants were limited to teleoperating one robot at a time, whereas vertex and path autonomy enabled simultaneous search with multiple robots. The joystick input method required participants to mentally map between the input device and robot orientations, and reorient when switching robots. These orientation costs were not imposed by the vertex and path levels of autonomy, which automatically oriented robots toward each path node.

It is no surprise that equipping some level of robot autonomy is likely to improve performance; however, the comparison between vertex and path levels of autonomy was more interesting. Figure 3-3 illustrates the incremental increase in autonomy from vertex to path

appears related to a *decrease* in performance (i.e., higher search and mission times). Video of the control interface window captured during each study trial was examined post hoc to identify factors which may have been contributed to this counterintuitive result. Fiducial marker tracking appeared reliable and sufficiently accurate to localize and track the robots. Most participants successfully employed multiple robots to simultaneously search different areas or interrogate an area from multiple approaching angles. On the other hand, the videos also provided evidence of potential influences by factors related to the test platform and participants' possible overreliance on automation, the latter of which may help explain the unexpected decrease in performance.

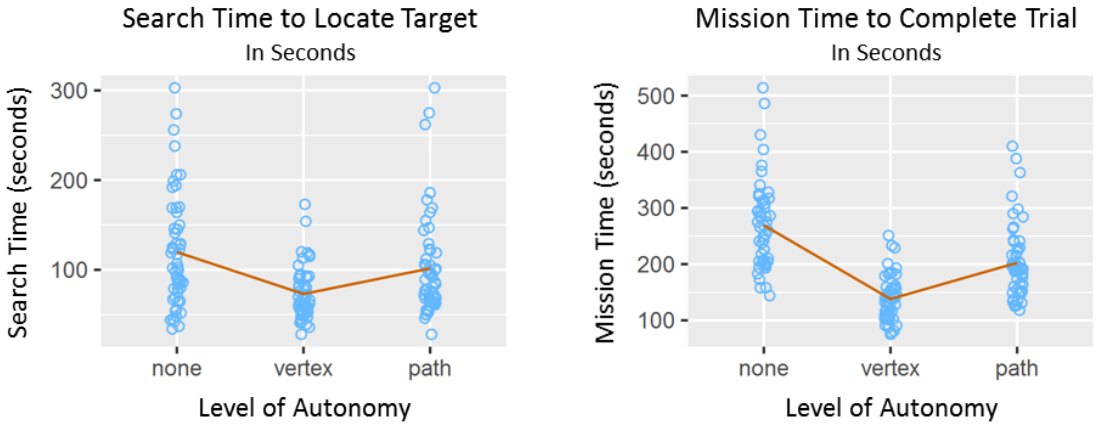

Figure 3-3: Search time and mission time by levels of autonomy (lower time is better). Unexpectedly, the highest level of autonomy (path) did not appear associated with best average task performance (i.e., lowest search time and mission times).

3.5 Discussion

3.5.1 Overreliance on Automation

The control interface videos revealed a pattern of participants generally seeking to exploit the full potential of the available automation. For the trials with full path autonomy, that often meant sending robots to the far corners or edges of the search space, a tactic not possible with vertex autonomy because the placement of obstacles prevented direct paths from the center of

the test environment to the outside edges. This observation is interesting because it suggests the level of autonomy influenced how participants conducted a search, and may indicate a tendency of overreliance on automation.

There were two task elements: (1) locate the hidden target, and (2) converge on the target's location with all four robots. The hidden target was positioned approximately two-thirds the distance from the center of the search space to the outside edge. A frequently observed search strategy was to send one robot to each of the four corners or outside edges of the arena. This *maximum dispersion* strategy could be implemented with a single command per robot when path autonomy was enabled. One or two of the robots often began sensing the target prior to reaching its goal. By this time, the other two to three robots traveled an approximately equal distance in the wrong direction. The best course of action was then to redirect each errant robot with a new goal near the target. However, participants often attended first to the robot(s) nearest the target or using the keyboard to report the target location. In these cases, the robots headed in the wrong direction continued moving farther away, sometimes reaching goals as far from the target as physically possible.

Although participants could have employed a maximum dispersion strategy with any level of autonomy, increasing levels of autonomy enabled robots to travel farther before reaching their goal point and stopping until the operator inputted another goal. Vertex autonomy required more frequent operator interaction, which provided opportunities to correct the course of errant robots sooner. Thus, a maximum dispersion strategy may have negatively affected performance during path autonomy trials more than vertex autonomy trials, one possible explanation for the unexpected average performance decrease observed in Figure 3-3 above.

3.5.2 Potential Test Platform Factors

The control interface videos provide some evidence of factors that may have influenced the

unexpected performance decrease observed with path autonomy versus vertex autonomy. Table 3-2 contains a list of several possible factors related to the platform design and implementation that were observed during the study and in the trial videos. However, none of these factors fully explain the lower average performance observed for the path autonomy trials versus the vertex autonomy trials.

Table 3-2: Potential test platform influences on study results

Category	Factor
Control Interface	Multiple input devices Input and video output latency Communication latency
Path Planning	Computation time Reliability of generated paths
Robot Platform	Communication latency On-board processing latency Non-holonomic vehicle dynamics
Test Environment	Sensor accuracy and reliability Collisions with physical obstacles

Operator input and communication latencies may have contributed to robot over-steer and long stopping distances, which sometimes triggered compensation by the path following algorithm or the participant. Path planning computation time in some instances exceeded the amount of time one might expect a human operator to plan a route of the same complexity. Although automatic path planning generally produced collision-free paths, some paths resulted in the robot colliding with an obstacle and getting stuck, which necessitated operator intervention and caused delays. These collisions may have been due to insufficient clearance afforded by the path planning algorithm when approaching an obstacle at certain angles, or marginal clearance combined with wheel slip and other navigation inaccuracies. Finally, non-holonomic vehicle dynamics appeared to have disproportionately affected trials that used automatic path following. This was especially noticeable for robots following automatically

generated paths. Many navigation challenges seemed related to the 3-wheeled design, with differential steering and a trailing free turning wheel, which resulted in the geometric center being offset toward the rear with respect to the center of rotation. This afforded the rear portion of the robot less clearance when turning, especially during pivot turns (i.e., rotations about the yaw axis).

3.5.3 Operator Attention and Eye Tracking

Table 3-3 compares the initial test platform used for the Levels of Autonomy study with human-robot interface guidelines suggested by [37]. The platform possessed five of six desired qualities, only lacking the ability to assist the operator in deciding level of robot autonomy. The optimum level of automation for a complex system may vary depending on the attentional and workload states of the operator. Thus, it would be beneficial to incorporate the measurement of these states.

Table 3-3: Human-robot interface guidelines and Levels of Autonomy platform

Human-Robot Interface Guidelines [37]	Platform [31], [36]
Provide a map of past robot locations	Yes
Provide fused sensor information	Yes
Support multiple robots in a single display	Yes
Minimize the number of windows	Yes
Spatial information about the robot in the environment	Yes
Assist user in deciding level of robot autonomy	No

Video of the control interface window captured during the study led to the observation that participants may have allocated attention inefficiently, perhaps due to overreliance on autonomy to help manage the complexity the operating four robots simultaneously. This observation and the desire to help the operator make better use of automation led to the following questions:

1) How can operator attention be estimated in real time?

2) Can estimated real-time attention be used to improve performance?

Attention is a cognitive function and thus is difficult to measure directly. Methods typically rely on behavior as a proxy for attention. This can take the form of a conscious action in response to an attended stimulus, such as clicking the “OK” button in a popup window, or an unconscious reaction like turning one’s head toward the source of a loud noise.

As discussed in Chapter 2, eye tracking technology enables physiological measurements linked to various aspects of human cognition, including attention [79-83]. The multirobot platform described in Chapter 4 applied eye tracking to estimate operator attention in real time, and incorporated attention as feedback in an effort to improve search task performance. In addition to estimating real-time attention for the user studies in Chapter 5, the captured eye gaze data were also recorded at 20 Hz in log files. These data are a rich source for future work.

3.6 Conclusion

Automation is necessary for a human operator to effectively control multiple robots; thus, research often focuses on how many robots can be operated [42] and methods to do so efficiency [39], [43]. This chapter describes Study 1: Levels of Autonomy, a user study which measured search task performance with four robots operated at each of three levels of autonomy [31], [36]. The intent of the automation and AR graphics employed by the control interface was to allow the participant to focus on higher-level tasks. With a fixed number of robots, successively higher levels of robot autonomy were expected to improve performance. However, the results revealed performance may actually *decrease* as autonomy increases past some threshold.

Recorded video of the user interface window provided indications that operator overreliance on automation and inattention may be related to the unexpected drop in

performance. This led to the hypothesis that measuring and incorporating attention as feedback can help improve performance. The initial test platform used to conduct the study was not designed to measure operator attention, which limited further examination of this hypothesis. Instead, the platform detailed in Chapter 4 incorporated eye tracking to advance this research, along with additional controls to address several other potential factors which may influence user study results.

CHAPTER 4: MULTIROBOT PLATFORM WITH EYE TRACKING

4.1 Introduction

This chapter details a multirobot test platform with integrated eye gaze tracking which was developed to support (1) the implementation of techniques employing physiological feedback to assess and respond to operator attention in real time, and (2) the evaluation of these techniques in a controlled setting. The presented platform incorporated a number of experimental controls and introduced specific capabilities based on the results of the Levels of Autonomy study in Chapter 3. The resulting system integrated physical robots in a combined live and virtual environment, along with eye tracking and gaze data processing to assess operator attention in real time.

4.2 Motivation

Increasing robot autonomy does not necessarily lead to improved task performance, as illustrated by the user study results discussed in Chapter 5. These results raised three questions in relation to teleoperation of multiple mobile robots:

1) Are the benefits of autonomy diminished by operator overreliance on automation?
2) Can operator attention be estimated in real time and used to improve performance?
3) What can be done to mitigate the potential platform influences identified by the Levels of Autonomy study?

This chapter describes a multirobot platform developed to address these questions and support the evaluation of task performance and efficiency pursuant to the aim and objectives of this research.

Figure 4-1 shows an overview of the platform. An overhead camera captured video of four tracked robots operated in the test environment. The control interface displayed this video, and projected virtual obstacles and targets from its simulation environment as graphical overlays on

the video frames. The interface used an eye tracker to determine where the operator looked on the display. All of this information was combined with operator input, and the control interface issued commands to the robots accordingly via a dedicated wireless network.

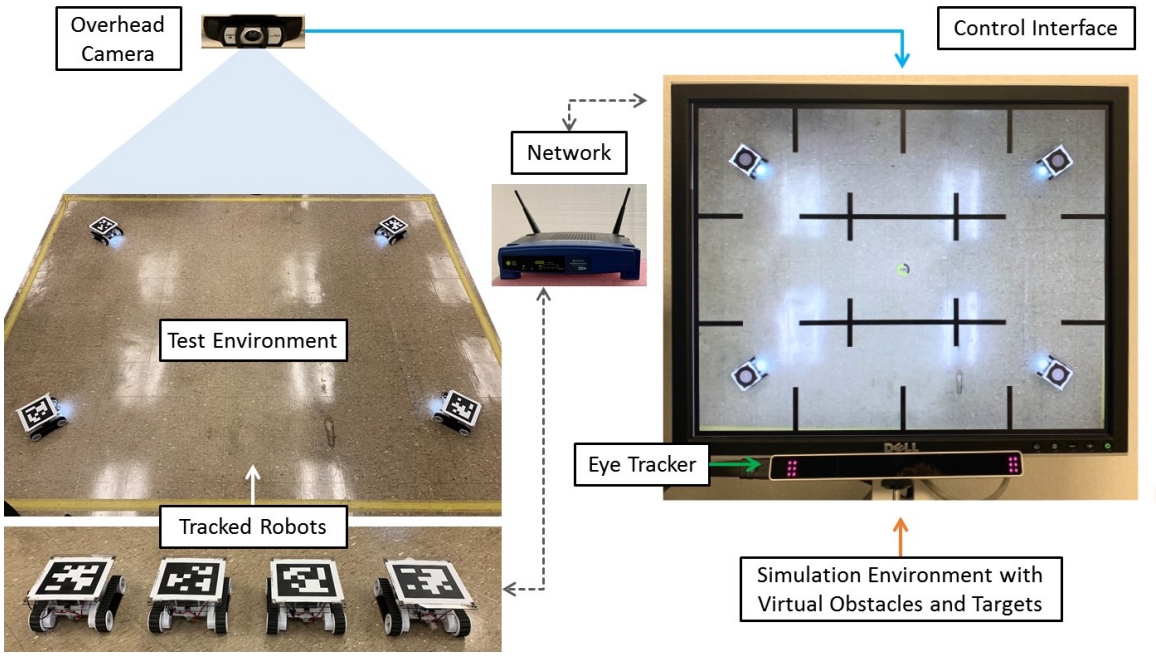

Figure 4-1: Multirobot test platform overview.
An overhead camera supplied video of four tracked robots in the test environment. The control interface rendered virtual objects from the simulation environment on the video and displayed the composite view. An eye tracker monitored the operator's gaze point and eye movements. The control interface communicated with the robots on a dedicated wireless network.

Table 4-1 contains the platform features designed to overcome the limitations observed with other platforms, and to support the real-time estimation of operator attention and workload. The new platform continued to employ physical robots in a controlled laboratory environment, but used computer simulation and augment reality to mitigate challenges with obstacle collisions and imperfect sensing. Virtualization provided additional benefits such as software defined obstacle and target maps that can be determined at runtime and quickly edited during the user study design phase. The platform incorporated eye tracking and gaze data processing in order to measure physiological properties related to attention and workload.

Implications of the simulation environment and the role of physical robots are covered further in the Discussion (see section 4.8).

Table 4-1: Multirobot platform design features to mitigate observed limitations

Category	Limitations	Platform design elements
Control Interface	Multiple input devices Input, video output latency Communication latency	Single common input device Multiple threads, processing framerates Asynchronous TCP, dedicated network
Path Planning	Computation time Reliability of generated paths	Replace path planning with operator input of multiple path vertices
Robot Platform	Communication latency On-board processing latency Non-holonomic vehicle dynamics	On-board Wi-Fi Single-board computer, Linux OS Robust tracked platform, Li-ion power, yaw axis at geometric center
Test Environment	Sensor accuracy, reliability Collisions with physical objects	Virtual targets and sensors Virtual obstacles, robot collision detection
Perception & Cognition	Unknown operator attention, workload, and intent	Eye tracking and gaze data processing
Data Analysis	Limited data to analyze unexpected results	Log robot state and path at 2 Hz Log eye tracking data at 20 Hz Automated session script

4.3 Architecture

Figure 4-2 presents the multirobot platform design architecture. A modular design approach was used to support future expansion, substitution, and other configuration changes as needed. The major components can be organized into four groups: (1) control interface hardware and software, (2) test environment, (3) network, and (4) tracked robot hardware and software. The control interface centralized much of the platform functionality, including the graphical user interface (GUI) and high-level robot control. The tracked robot design included software components for low-level control.

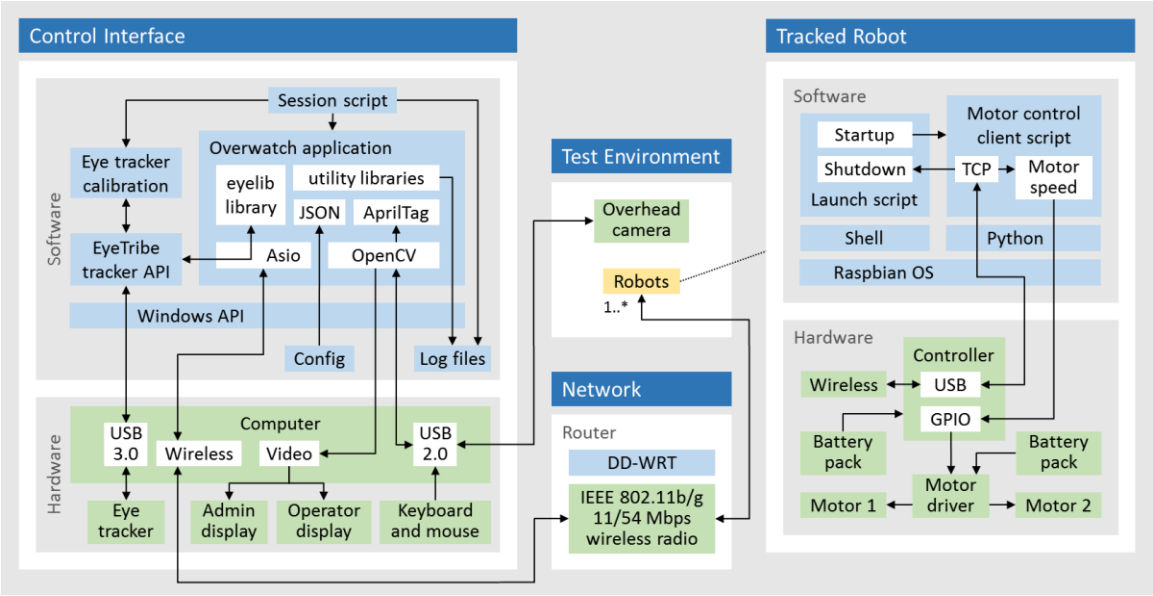

Figure 4-2: Multirobot test platform architecture organized into four groups (dark blue) with hardware (green) and software (light blue) components.

4.3.1 Hardware

The control interface consisted of a general-purpose computer, two displays, keyboard, mouse, eye tracker, and chin rest assembly (see Figure 4-3). The operator display was an LCD monitor set to 1280×1024 px resolution which showed the control interface software application window. The administrator display was a second monitor used by the system administrator to launch software, review study instructions, calibrate the eye tracker, and monitor platform software execution and results. This display and the keyboard were used exclusively by the system administrator. The administrator display was turned away from the operator display so as not to distract the operator. The operator used the mouse to input commands. An eye tracker was positioned below the operator display. A chin rest assembly was constructed to ensure the operator's face and eyes were within the view window of the eye tracker and minimized rotations of the head. See section 4.5.2 for details about the eye tracker and chin rest assembly.

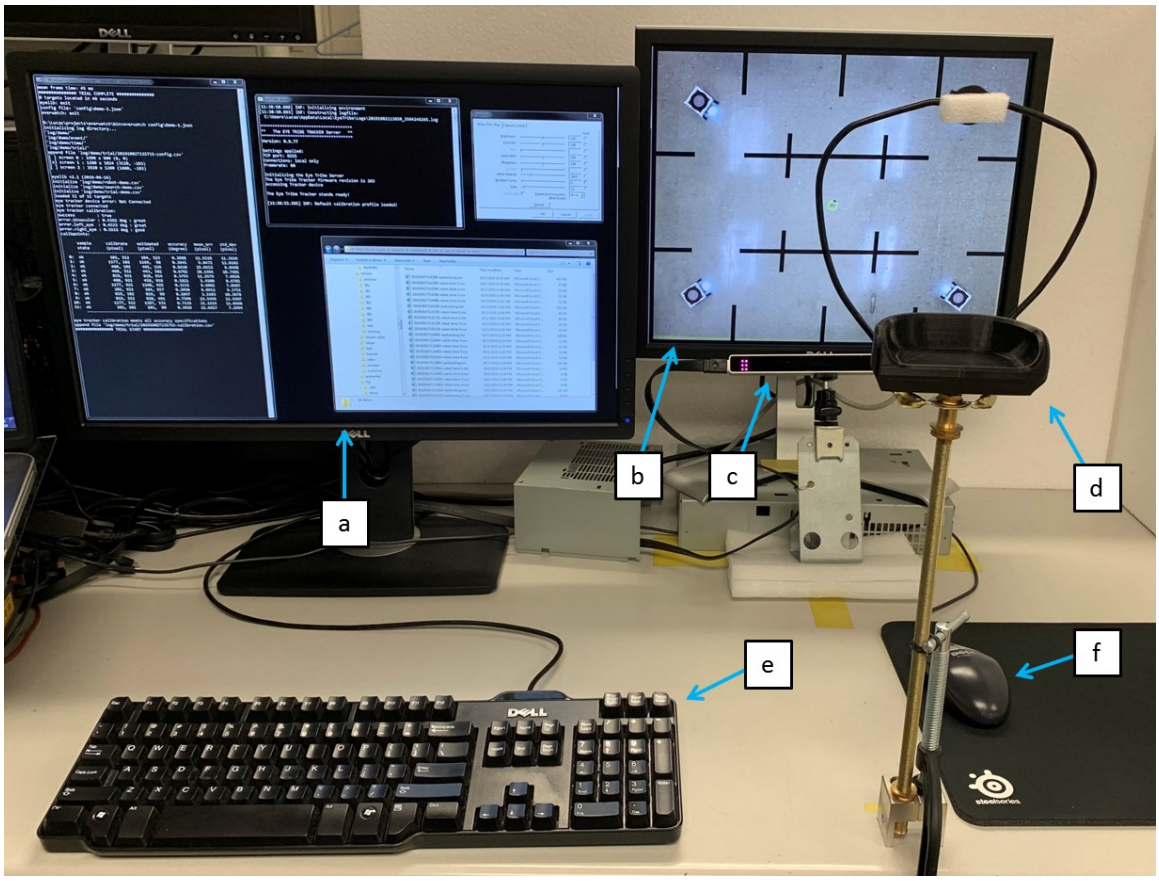

Figure 4-3: Multirobot test platform control interface.
The interface included an administrator display (a), operator display (b), eye tracker (c), chin rest assembly (d), keyboard (e), and mouse (f). The administrator display and keyboard were used exclusively by the system administrator. The operator used the mouse in input commands.

A Linksys WRT54GL Wireless-G router provided a dedicated closed network for the control interface and robots (see the left side of Figure 4-4). The control interface computer and robots connected to the network via the router's IEEE 802.11b/g wireless radio, which was capable of data transfer rates up to 54 Mbps. Open source DD-WRT firmware was installed on the router and a static IP address was assigned to each connected device. The network was designed to accommodate future expansion and was more than adequate to handle the traffic between the control interface and the four robots.

A Logitech C930e Pro video camera was mounted to the ceiling above the test environment

using spring tension rods inserted between two reinforced concrete beams (see the right side of Figure 4-4). The camera delivered video at 1080p (1920×1080 px) to the control interface computer via USB cable. The control interface software processed and displayed the video to provide a view of the test environment which was approximately 3.657 m (12 ft) wide by 3.048 m (10 ft) tall. The specific camera model was selected for its 90-degree field of view, which produced video of the relatively large test environment (11 m^2) with limited distortion. The AprilTag visual fiducial system [119], [120] was used to estimate the location and orientation of the robots in the test environment. A fiducial marker on each robot uniquely identified it and facilitated full 6-DOF localization.

Figure 4-4: Multirobot platform network router (left) and overhead camera (right). The wireless router provided a closed network. The overhead camera was mounted to the ceiling above the test environment using spring tension rods.

4.3.2 Software

Table 4-2 lists the major control interface software components with a summary of their respective purpose and where further details are presented. The primary interface software was the *Overwatch* application (`overwatch.exe`), with eye tracking functionality supported by the *Eyelib* library (`eyelib`). The Session Script was a batch script which facilitated user study

sessions and implemented experimental controls. These three components were developed as part of the presented work. The EyeTribe Server and EyeTribe UI applications provided access to the eye tracker device and calibration utilities respectively. Section 4.4.2 covers software components not listed in Table 4-2 which were on board the tracked robots.

Table 4-2: Multirobot test platform control interface software components

Software	File	Purpose	Details
Eyelib	`eyelib`	Eye tracking library	Section 4.5
EyeTribe Server	`EyeTribe.exe`	Eye tracker device manager	Section 4.5
EyeTribe UI	`EyeTribeUIWin.exe`	Eye tracker calibration	Section 4.5
Overwatch	`overwatch.exe`	Primary interface application	Section 4.6
Session Script	`session.bat`	Facilitate user study sessions	Section 5.4

The block diagram in Figure 4-5 presents a high-level view of the control interface software, their constituent modules, supporting libraries, and application programming interfaces (APIs). The orange circles and lines in the diagram highlight how these components were used together as a workflow. Session Script launched EyeTribe UI to facilitate eye tracker alignment and calibration, then launched Overwatch to run the primary platform application for each trial.

Overwatch contained six modules (darker blue within the gray Overwatch box in Figure 4-5). The `core` and `robot` modules used the Asio library to communicate with the robots. Eyelib contained four modules. The `tracker` module accessed the Eye Tribe API by exchanging TCP messages with the EyeTribe Server application.

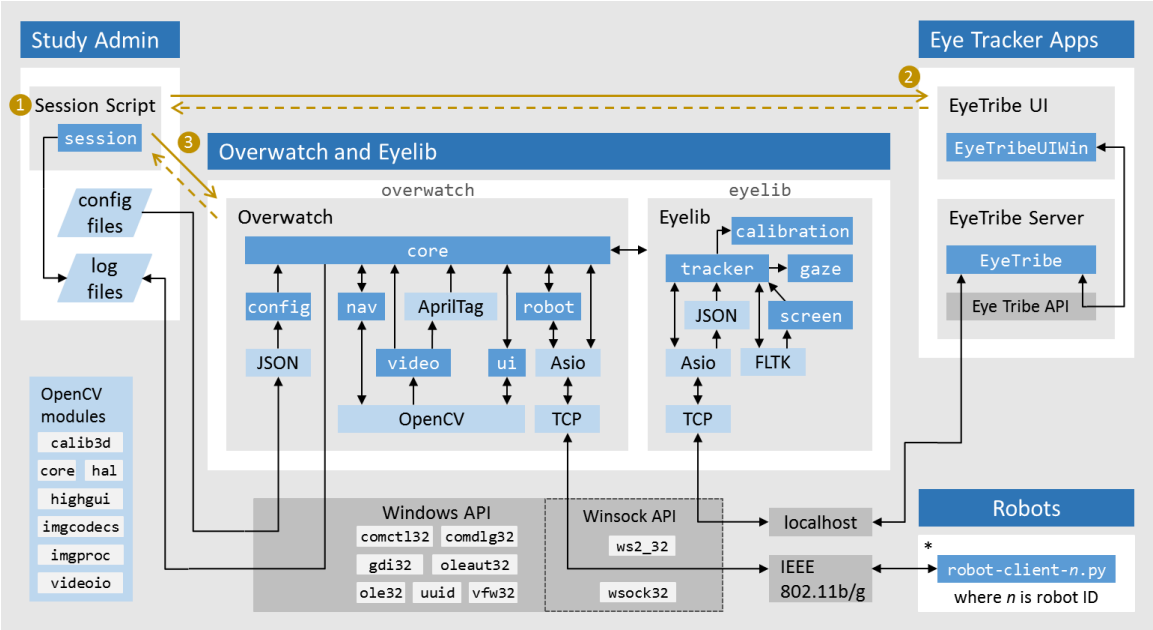

Figure 4-5: Control interface software and network interfaces.
Session Script (1) launched the EyeTribe UI application (2) and Overwatch (3).
Overwatch was launched multiple times during a single session, once per trial.

4.4 Tracked Robots

The four tracked robots shown in Figure 4-6 were built according to a common platform design developed to support this research. This section summarizes the robot hardware and software components. See Appendix A for additional design and implementation details.

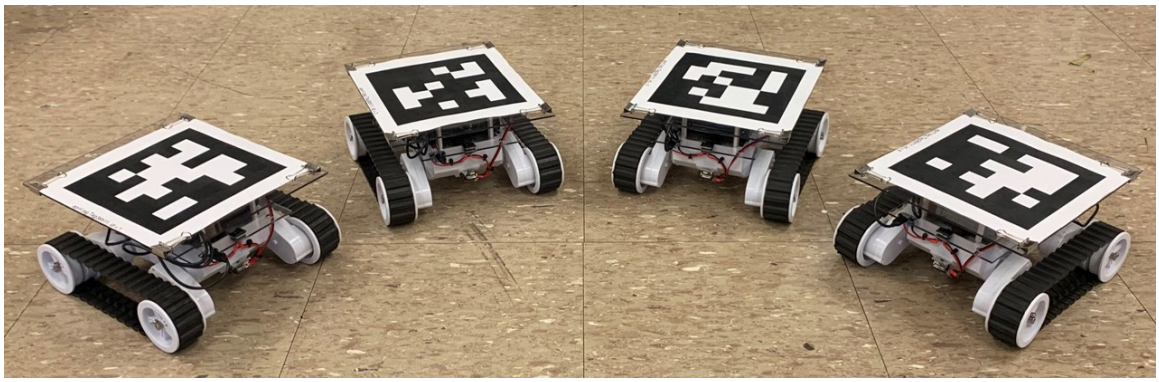

Figure 4-6: Tracked robots.
The chassis provided a stable platform for the onboard controller, motor driver, battery packs, and other electronics. A unique fiducial on each robot was used for localization and tracking.

4.4.1 Hardware

A tracked vehicle configuration with differential steering was selected for the presented tracked robot due to its maneuverability within the relatively small test environment and consistent turning clearances afforded by aligning the yaw axis with the geometric center of the platform. In order to reduce hardware variability and simplify maintenance, commercially available components were integrated to the maximum extent feasible and drill hole pattern templates were used where fabrication was necessary.

Figure 4-7 provides an overview of the major components. A Dagu Rover 5 tracked chassis was used to provide a relatively robust, stable platform with a track and wheelbase of approximately 230 mm (9 in). The chassis came equipped with left and right motor assemblies with integrated gearboxes, wheels, rubber tracks, and electrical wiring. A Raspberry Pi 2 Model B [121] served as the robot controller, with a DRV8835 dual motor driver shield installed on the general-purpose input/output (GPIO) header. An Edimax EW-7811Un USB Wi-Fi module connected the robot to the test platform network.

Two mounting plates were fabricated from 2.36 mm (0.093 in) thick polycarbonate sheets to integrate the components. Four 3.8 cm (1.5 in) aluminum standoffs were used to attached the upper and lower mounting plates to the chassis. The control was mounted between the plates on four short nylon standoffs attached to the lower plate. Four 1.9 cm (3/4 in) wide binder clips were used to clamp a unique AprilTag fiducial to the top surface of the upper mounting plate.

Two lithium-ion battery packs provided power to the robot. A 5.1 Ah pack powered the controller and other digital electronics. A 6.7 Ah pack supplied power for the motors via a USB Micro-B breakout board attached to the lower mounting plate. A single-throw toggle switch was installed in the lower plate between the USB breakout board and a 100 mA USB LED lamp mounted to the front of the robot. The lamp was inserted into the motor power circuit to

prevent the battery pack from shutting down due to low current conditions, and also provided a visual indicator of the robot's orientation. A complete bill of materials with quantities and dimensions can be found in Appendix A.

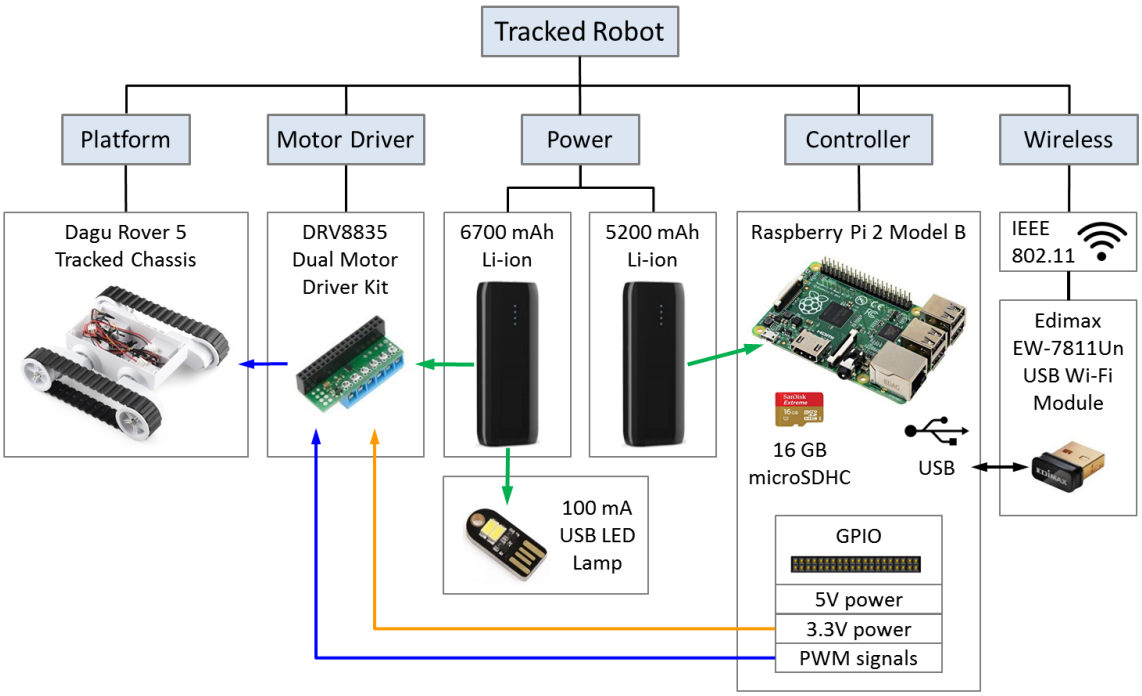

Figure 4-7: Tracked robot major hardware components.
The motor driver was installed on the general-purpose input/output (GPIO) header of the controller. Not shown: Electrical connectors, cables, toggle switch, polycarbonate mounting plates, fasteners, and other mounting hardware. See Appendix A for a complete bill of materials.

Figure 4-8 contains a schematic diagram of the motor driver circuit. The motor driver board could optionally supply power to the controller via the GPIO header, but this feature was not used. The controller was instead powered by a separate battery pack, which also provided power to the H-bridge integrated circuit via the GPIO's regulated 3.3 V pin.

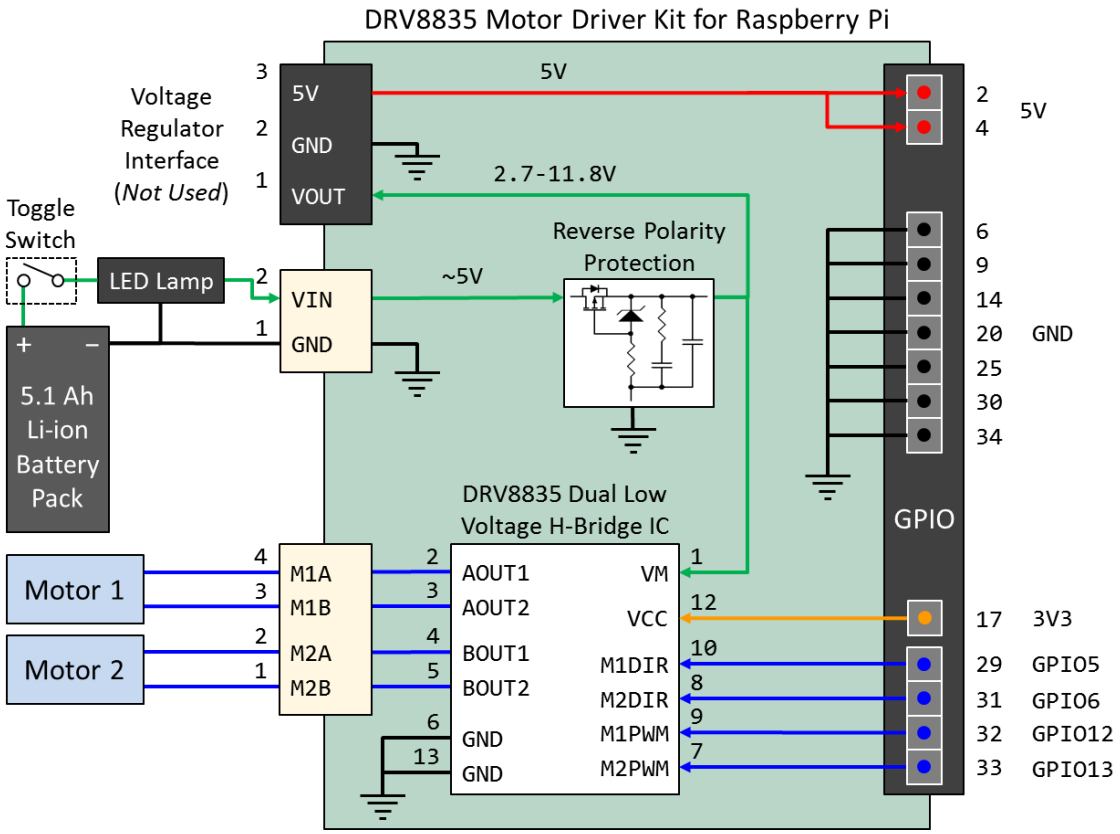

Figure 4-8: Schematic diagram of the tracked robot motor driver circuit.

4.4.2 Software

The Raspbian operating system was installed on the controller to support the software onboard the robots. Table 4-3 contains a summary of software component.

Table 4-3: Tracked robot software

Software	Purpose
`launch-robot-#.sh`	Shell script to launch robot-client at startup and shutdown OS upon exit
`robot-client-#.py`	Python script for TCP I/O and motor commands
`Pololu_drv8835_rpi`	Python library for DRV8835 dual motor driver
`WiringPi2-Python`	Functions for managing IO expanders
`Python`	Script language interpreter
`WiringPi`	GPIO access library for the BCM2835 SoC

'#' in script names refers to the robot number (1, 2, 3, or 4)

Robot functionality was distributed between the onboard controller and the centralized control interface software. Onboard software was minimal because the control interface was responsible for motion planning and sent motor speed values to the robot via TCM messages. Figure 4-9 illustrates the onboard controller software and interfaces.

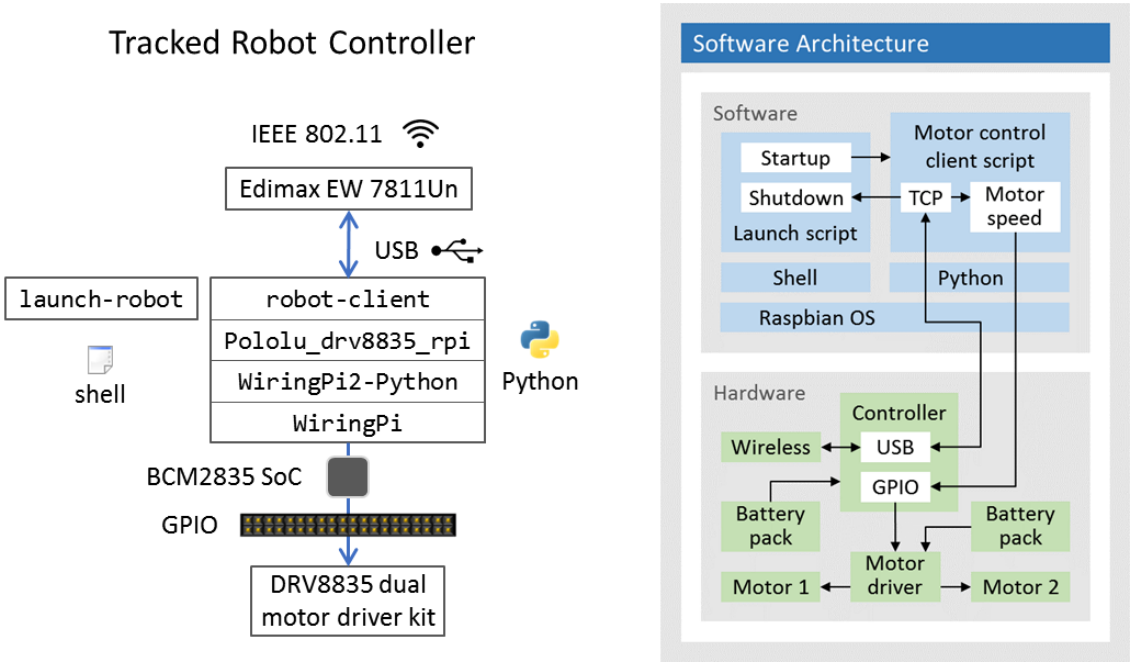

Figure 4-9: Tracked robot software components and interfaces.

`launch-robot` was a shell script which managed software startup and shutdown. `robot-client` was a Python script which processed TCP communication from the control interface and issued commands to the motor driver via the `Pololu_drv8835_rpi` library. These scripts contained a unique identification number for each robot and were named accordingly. For example, `launch-robot-1` and `robot-client-1` were installed on robot 1.

A `cron` task was scheduled on each robot to execute `launch-robot` each time the controller booted (see Figure 4-10). `launch-robot` simply launched `robot-client`, waited for it to complete, then issued a shutdown command to the operating system (OS). In addition to

processing motor commands, `robot-client` listened for a shutdown command to be issued by the control interface. Upon receiving the shutdown command, `robot-client` stopped processing and returned execution back to `launch-robot`, which then issued a shutdown command to the controller OS. Thus, the onboard software ensured the main `robot-client` script always ran when the robot was powered on, and an orderly shutdown occurred before the robot was powered down.

```
#!/bin/sh
# launch-robot-1.sh
echo `date +"%Y-%m-%d %H:%M:%S"` `hostname`
#sleep 20
cd /
cd /home/pi
echo `date +"%Y-%m-%d %H:%M:%S"` robot-client-1.py
sudo python3 robot-client-1.py $1
sudo shutdown -h now
```

```
@reboot sh /home/pi/launch-robot-1.sh >/home/pi/log/cronlog 2>&1
```

Figure 4-10: Tracked robot software launch script (top) and `cron` task (bottom).

4.5 Eye Tracking and Data Processing

4.5.1 Initial Testing

A webcam-based system was developed to assess the feasibility of integrating eye tracking technology in the multirobot test platform. A USB connected visible and near-infrared (VNIR) camera was designed and assembled utilizing the camera board from a Creative Live Cam Socialize HD webcam. The camera board was modified with a new lens mount to accept M12 threaded lenses, and placed in a custom housing with a 1/4-20 UNC threaded receptacle for mounting. A 25 mm C mount adapter was fabricated in order to install a 720 nm IR passing filter. The camera and two 30-LED infrared lamps at 850 nm wavelength were mounted to a rigid frame with standard 1/4-20 UNC threaded studs. A ball head was used to enable alignment of

the camera for an optimal view of the operator's eyes. The resulting hardware assembly was positioned on a desk below a computer monitor and facing the operator.

Figure 4-11 shows the assembled hardware and screen captures from initial testing. The ITU Gaze Tracker [122], [123] library was used to process video frames from the eye tracker camera. The system tracked pupil and corneal reflections of light from the IR lamps, using an interpolation-based technique to map eye features to the point of visual gaze. The results demonstrated the feasibility of incorporating eye tracking technology in the multirobot platform.

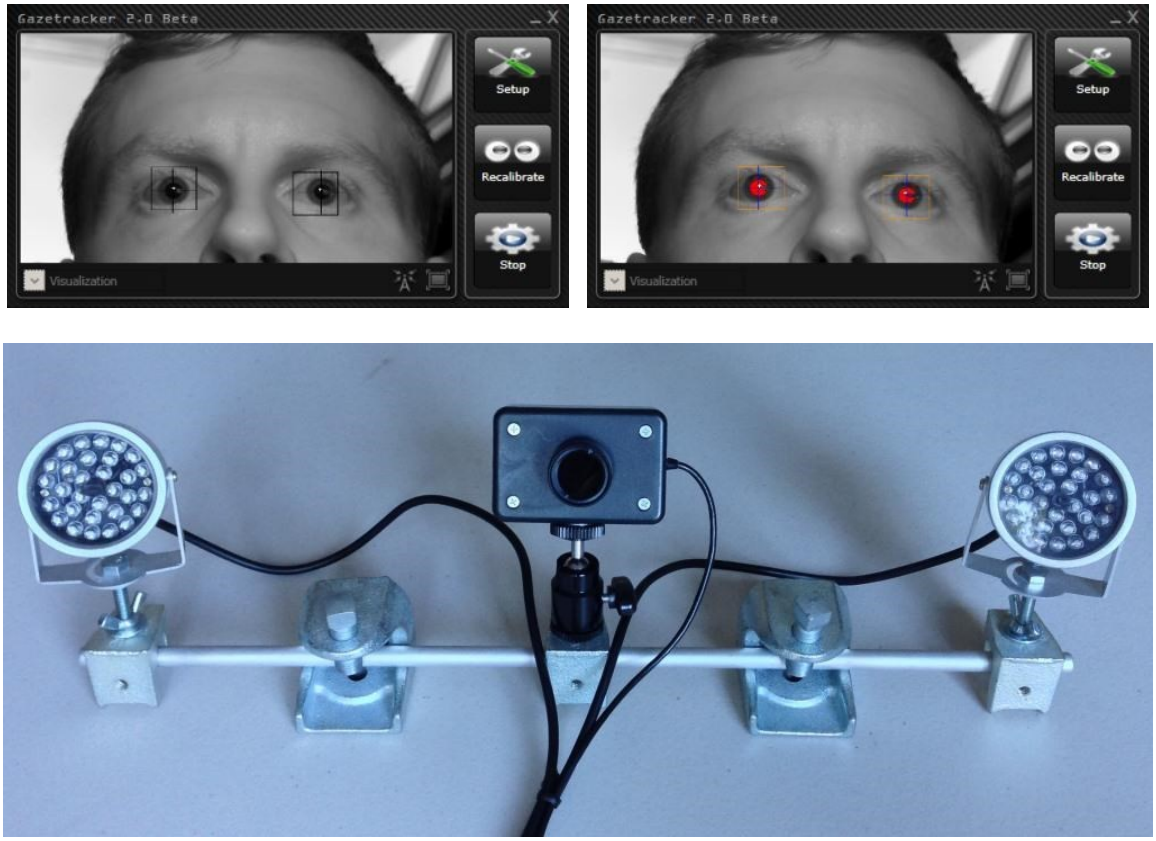

Figure 4-11: Gaze Tracker software [122], [123] (top) and test eye tracking hardware (bottom). Visualization graphics could be optionally enabled in the software to illustrate the image processing steps (top left and right). For testing, a near-infrared camera was built and attached to a rigid frame with 850 nm wavelength near-infrared lamps to the left and right (bottom).

4.5.2 Hardware

The multirobot test platform used an Eye Tribe Tracker ET1000 (The Eye Tribe Aps) eye tracker. The device hardware contained an integrated camera and infrared illumination in a compact package 20 × 1.9 × 1.9 cm (7.9 × 0.75 × 0.75 in). Table 4-4 contains technical specifications for the ET1000. Placement of the eye tracker near the display was important to obtain acceptable calibration results. Keeping the head still with minimal rotations was also important during calibration, although the device was less sensitive to head rotation once calibrated. A chin rest assembly was constructed to ensure the operator's face and eyes were within the view window of the eye tracker and to minimized rotations of the head.

Table 4-4: ET1000 eye tracker technical specifications

Specification	Value
Sampling rate	30 Hz or 60 Hz
Accuracy	0.5° – 1°
Spatial Resolution	0.1° (RMS)
Latency	< 20 ms at 60 Hz
Calibration	9, 12, or 16 points
Operating range	45 – 75 cm
Tracking area	40 × 30 cm at 65 cm distance (30 Hz)
Screen sizes	Up to 24-inch diagonal
API/SDK	C++, C#, and Java included
Data output	Binocular gaze data
Dimensions (W/H/D)	20 × 1.9 × 1.9 cm (7.9 × 0.75 × 0.75 in)
Weight	70 g
Connection	USB 3.0 Micro-B SuperSpeed

Figure 4-12 shows the eye tracker and chin rest assembly. The tracker was mounted on a ball head attached to a metal bracket below the operator display. The ball head enabled quick alignment of the device for an optimal view of the operator's face and eyes. The eye tracker and control interface computer were connected by a USB 3.0 cable with Micro-B SuperSpeed and Standard-A plugs respectively.

The chin rest assembly featured a plastic chin rest cup designed and 3D-printed in the lab, and a forehead rest and support rod constructed from repurposed brass-finish light fixture hardware (see the right side of Figure 4-12). A short hollow threaded lamp pipe was inserted into a hole at the bottom of the chin rest cup to facilitate mounting. The hole was tapped by carefully turning the pipe in order to form threads. A long hollow threaded lamp pipe served as the vertically-aligned support rod. The rod was secured to the control interface desk with two aluminum brackets and a C-clamp. The chin rest cup was attached to the top of the rod using threaded couplings, hex nuts, and washers.

The forehead rest was adapted by bending a lamp shade harp, painting it flat black for a non-reflective matte finish, and attaching a replaceable foam pad to the threaded finial stud. The harp was bent such that the foam pad came into contact with the forehead. The forehead rest was attached to the support rod with a lamp shade saddle. The saddle was bent to position the forehead rest closer to the display than the chin rest cup and support rod.

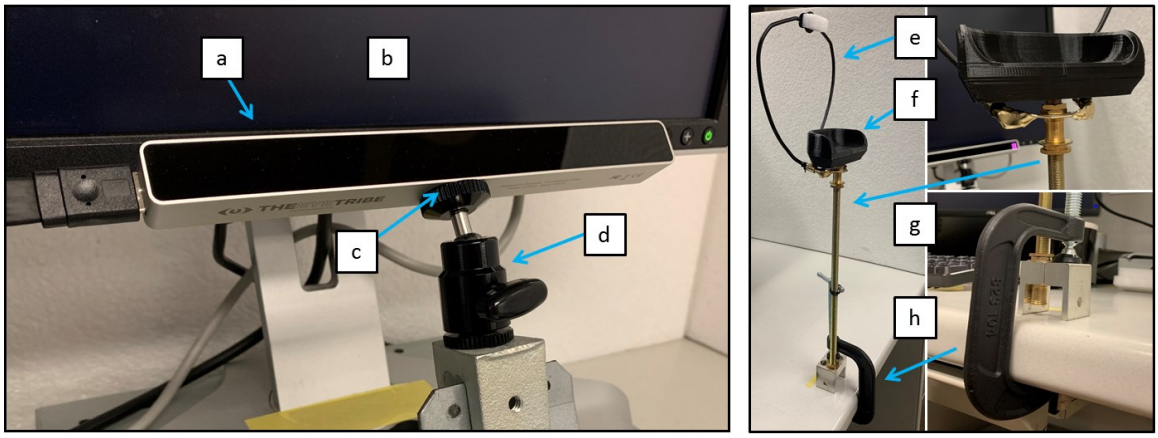

Figure 4-12: Eye tracker (left) and chin rest assembly (right).
Left: The eye tracker (a) was positioned just below the operator display (b). The housing had a standard 1/4-20 UNC threaded receptacle (c) which was used to mount the device on a ball head (c). Right: The chin rest assembly had forehead (e) and chin (f) rests held up by a threaded lamp pipe (g), which was attached to the control interface desk with a C-clamp (h).

4.5.3 Software

The EyeTribe Server console application (`EyeTribe`) initialized the eye tracker device and provided access to device settings, calibration procedure and results, and streaming eye gaze data via the Eye Tribe Tracker API. Client applications accessed the API via JavaScript Object Notation (JSON) formatted TCP messages exchanged asynchronously with the EyeTribe Server. Streaming data included raw and smoothed gaze point coordinates, pupil size, and normalized pupil coordinates. These data were available for the left and right eyes, and as composite values for both eyes.

The EyeTribe UI client application (`EyeTribeUIWin`) facilitated configuration and calibration of the eye tracker. Figure 4-13 contains a screenshot of the EyeTribe UI main window. Figure 4-14 shows how the application was used to align the eye tracker device and execute the automated calibration procedure.

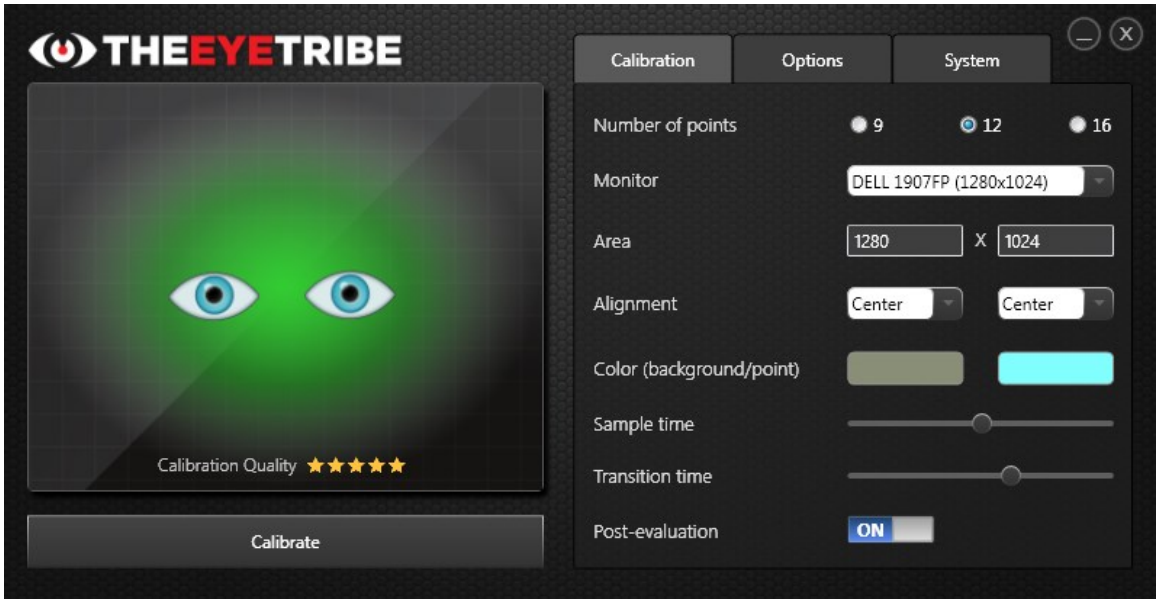

Figure 4-13: EyeTribe UI main window.
The left panel depicted the viewing window of the eye tracker device. This panel was used to physically align the device prior to calibration. The Calibrate button started the automated calibration procedure with the number of calibration points specified on the right.

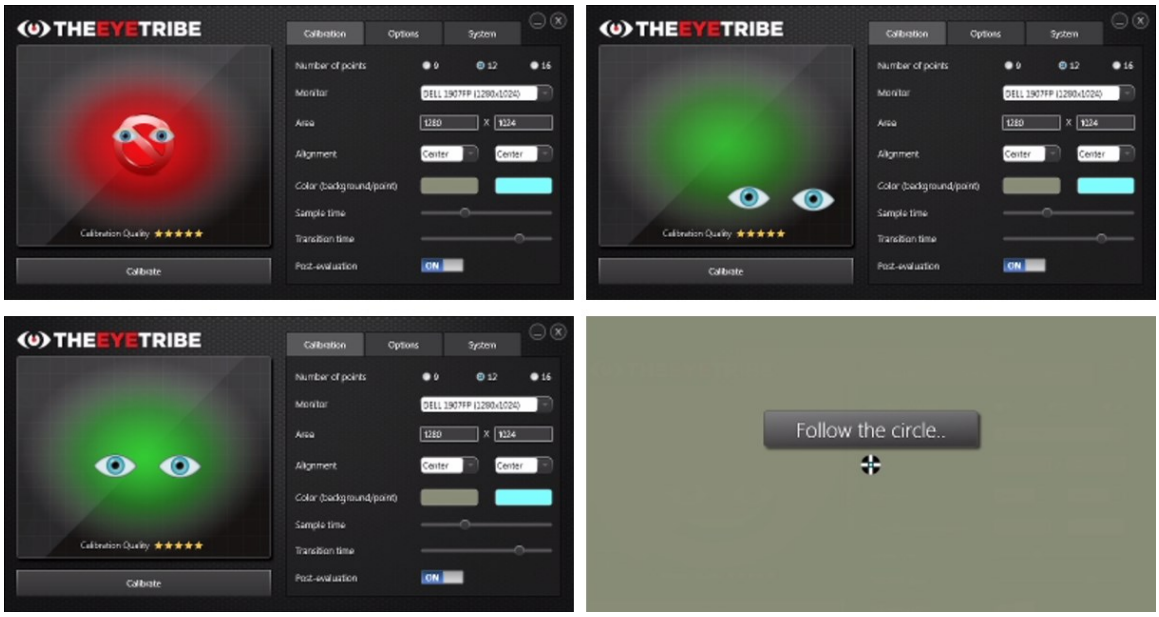

Figure 4-14: EyeTribe UI tracker alignment and calibration.
The left panel graphic was red if the eye tracker did not detect eyes (top-left) and depicted the position of the eyes relative to the device viewing window (top-right). The device was physically aligned until the eyes were centered in the panel (bottom-left). During the automated calibration procedure, the operator watched as the calibration circle shifted and dwelled at each calibration point (bottom-right).

The EyeTribe Server and EyeTribe UI applications were both included in the Eye Tribe software development kit (SDK) from the manufacturer of the ET1000 tracker. The Eye Tribe SDK also included a reference implementation of the publicly available open API. This reference code was not incorporated into the multirobot test platform. Instead, the API was implemented along with other features in the Eyelib (`eyelib`) eye tracking library.

Eyelib was a C++ library developed as part of the presented work to define API-agnostic gaze data structures and related functions; fixation detection algorithms, including dispersion threshold and velocity threshold; consistent interfaces for blink, fixation, pupillometry, and saccade measurements; and access to screen (i.e., computer display) properties. Figure 4-15 presents the modular design approach used to encapsulate and expose subsets of functionality to client applications. The interfaces were designed to support the integration of alternative eye

tracker APIs if needed in the future, without altering the client application.

The Eyelib source code contained 3,967 lines of code spread across 45 files. The library leveraged the Asio library [124] for asynchronous TCP communication and JSON for Modern C++ (nlohmann/json) [125] to serialize and deserialize messages in JavaScript Object Notation (JSON) format. The `tracker` module used the Fast Light Toolkit (FLTK) GUI library to optionally display a window showing raw and smoothed gaze points on the screen, or perform an automated calibration procedure similar to the EyeTribe UI application. The `screen` module used FLTK to obtain information about available displays.

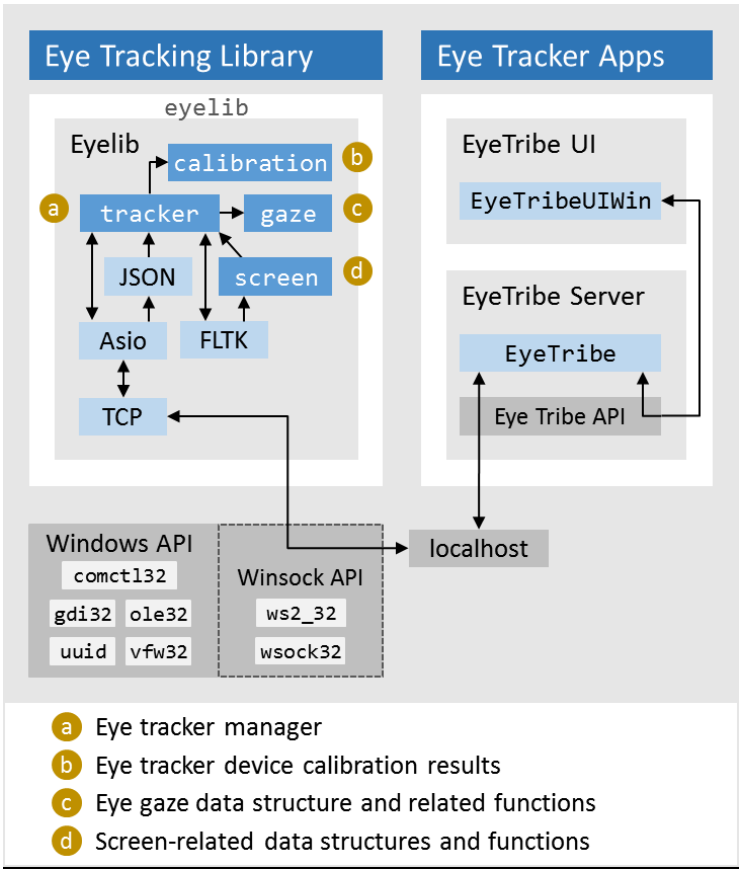

Figure 4-15: Eyelib eye tracking library software block diagram.
Eyelib contained four modules, marked in the diagram with circle containing a letter. The `tracker` module provided core functionality, including all access to the Eye Tribe Tracker API.

4.6 Overwatch Software Application

4.6.1 Architecture

The Overwatch application (`overwatch.exe`) was the primary software component of the multirobot test platform and the graphical user interface (GUI) used to control the robots. It received and processed all streaming video from the overhead camera, user input from the keyboard and mouse, streaming eye gaze data from the eye tracker, and TCP connection requests from the robots. The application displayed processed video frames and computer-generated graphics on the operator display, and outputted data to log files. Figure 4-16 highlights some of the graphics rendered in the application window.

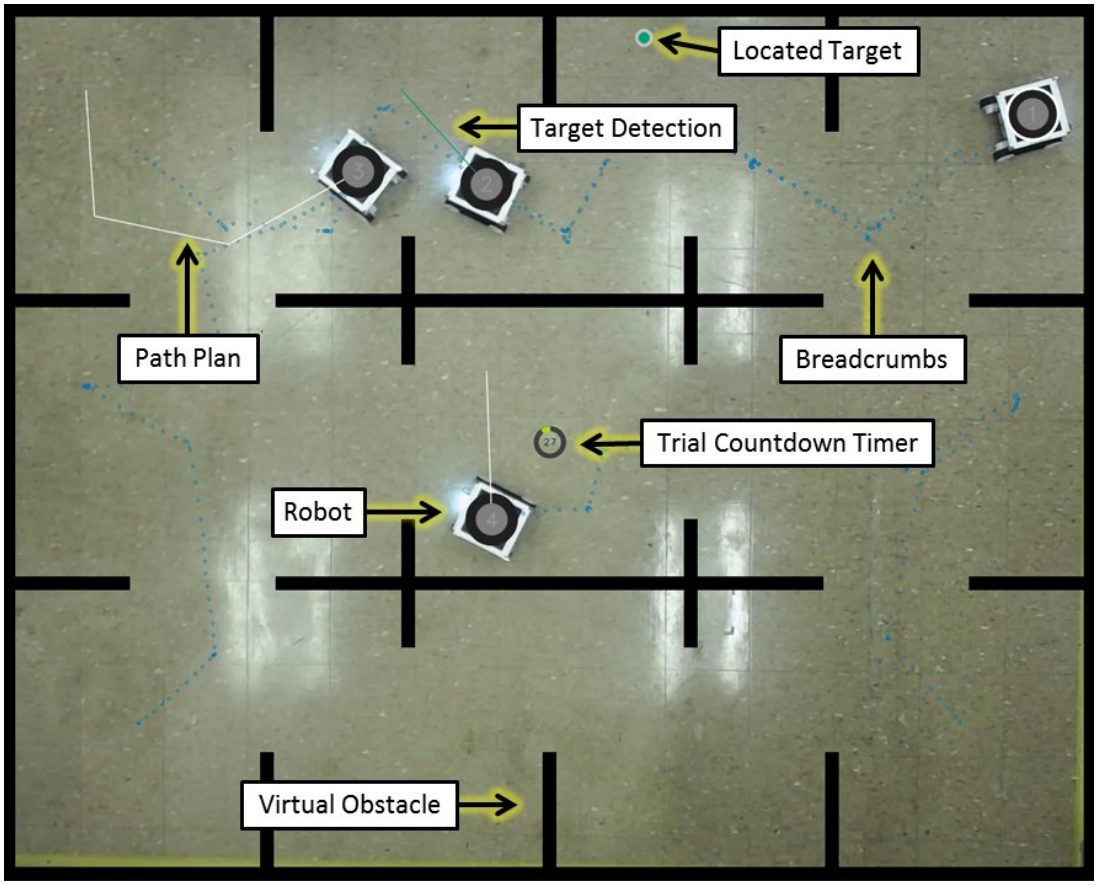

Figure 4-16: The Overwatch application window displayed video with graphical overlays. Rendered graphics included robot paths, virtual obstacles and targets, and a countdown timer showing how many seconds remained during a user study trial.

The application was a multi-threaded program written in C++. The source code contained 8,906 lines of code spread across 98 files. It was divided into six modules, identified by the orange circles in Figure 4-17. These modules logically grouped common functionality and programming interfaces. For example, the `robot` module metaphorically represented the properties and behaviors associated with a physical robot operating in the real world.

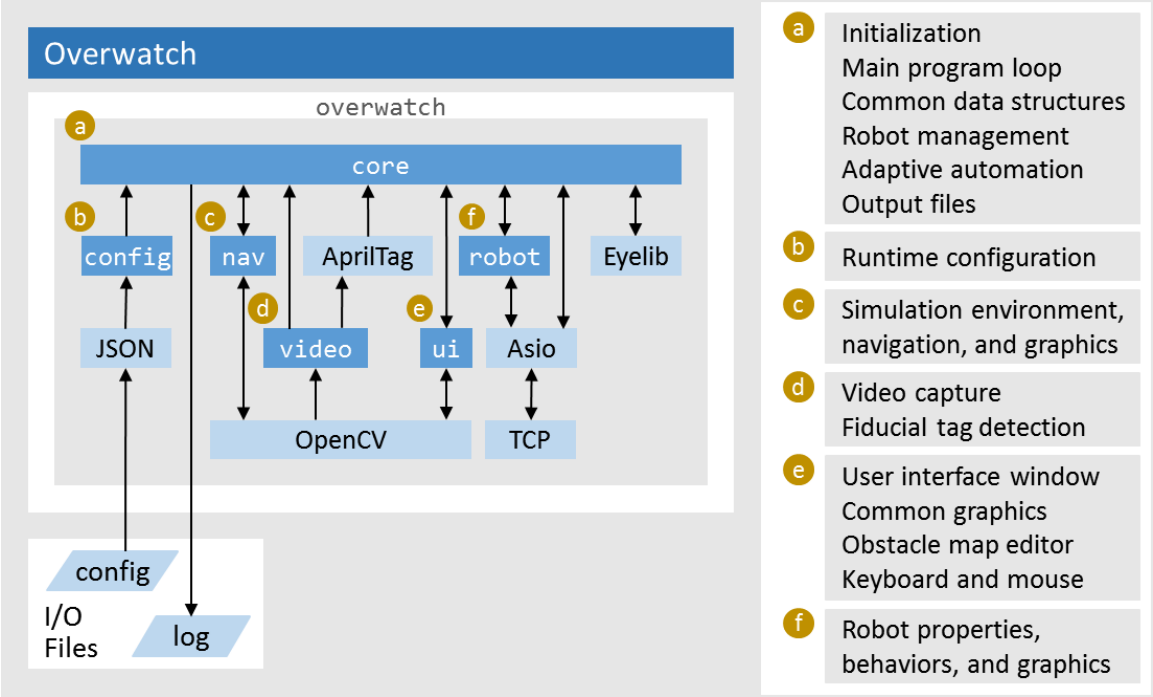

Figure 4-17: Overwatch software modules and supporting libraries.

The `core` module was central hub of the application. It initialized the application, registered event handlers, spawned a dedicated fiducial tag detection thread for robot tracking, implemented the main program loop, and released resources upon exit. The main loop captured and processed video frames, called functions to render graphics, showed the processed frames in the window, and processed user input. Common data structures and related functions defined by the `core` module supported eye tracking metrics and eye tracker status, log file output, aggregate time duration data, search tasks and target objects, and basic date and time

functions.

The `core` module also centralized management of robot objects by distributing obstacle and target configurations, receiving asynchronous TCP connection requests from the physical robots, forwarding keyboard and mouse input, detecting and intervening to avoid robot collisions, and layering robot graphics drawn on the screen. Robot management also included the adaptive automation algorithm used to assign a confidence value to each robot. Rather than delegating this algorithm to the robots, it was implemented centrally in order to support future work involving mutual confidence shared by two or more robots, and as a practical matter to enable efficient data processing.

The `config` module parsed JSON formatted configuration files to obtain a wide variety of application settings defined at runtime. Table 4-5 summarizes the types of configuration files Overwatch could process. These files enabled changes to obstacle and target maps, robot speeds, and other settings without recompiling the software. This greatly facilitated both the development of study parameters and within-session test platform reconfiguration to change study conditions between trials.

The `nav` module provided a simulation environment with virtual obstacles and targets, paths and breadcrumbs to navigate and track movement, and collision detection. Each virtual object was defined by a location and dimensions in video image coordinates, and could be drawn over captured video frames to produce augmented reality graphics.

Table 4-5: Types of Overwatch configuration files

Group	Purpose
`condition`	Study condition codes
`eyetrack`	Eye tracking options, maximum error values, and device communication settings
`log`	Data logging options
`obstacle`	Coordinates and dimensions of virtual obstacles, and obstacle exclusion zone margin
`robot`	Robot communication, confidence, fiducial tag, graphics, motion, and virtual sensor settings
`session`	Configurations for each study trial within session
`target`	Search target coordinates
`task`	Search task settings
`tutorial`	Tutorial steps with optional graphics and instructional text at specified coordinates
`video`	Camera device ID, resolution, crop size
`window`	Application window screen ID, position, size, and splash background color

The `robot` module contained data structures and algorithms that defined the properties and behaviors of robots in the simulation environment including confidence level and value, pose (position and orientation), path vertices, collision state, and input and motion states (see Table 4-6), and virtual sensors to detect targets. These relied heavily on functionality from the `nav` module. The `robot` module performed motion planning to translate virtual paths into motor commands, and sent these commands via TCP to the physical robots. It projected computer-generated graphics on captured video frames to represent robot properties and behaviors in context of the real test environment. This module also recorded various event and state data, and output these to log files for analysis.

Table 4-6: Robot input and motion states

State	Description
`autonomous`	Automatically moving to next path vertex
`deactivated`	Not accepting input or recording data
`input_path`	Accepting path vertex input from operator
`input_teleop`	Accepting teleoperation input from operator
`wait_collision`	Movement paused by automatic collision detection
`wait_idle`	Waiting idle for input from operator
`wait_pause`	Movement pause by operator (not used)
`wait_target`	Waiting near a detected target

The `ui` module included the user interface window, common graphics, an obstacle map editor for developing and exporting obstacle configuration files, and keyboard and mouse input. Figure 4-18 displays the colors defined for the user interface. In order to provide contrast between different graphical elements while enhancing accessibility, a color palette was designed based on a 7-color palette adapted for color blindness [126].

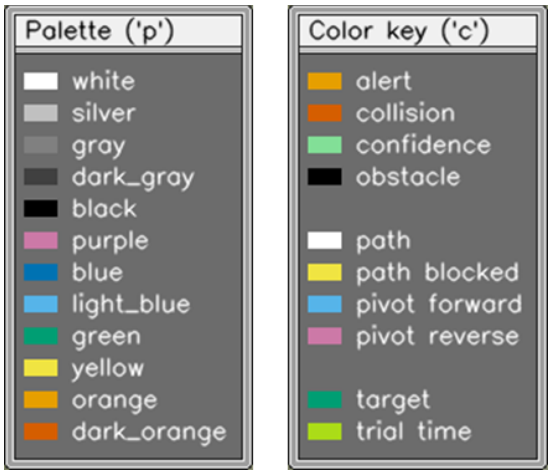

Figure 4-18: Overwatch user interface color palette and key.
Left: The color palette was designed with accessibility in mind, and provide a base set of colors available in the software code. Right: Colors were modified or added to suite specific needs, including contrast with the test environment.

The `video` module initialized video capture and provided fiducial tag detection enabled by the AprilTag [119], [120] library. The main program loop–contained in the `core` module–passed

every other video frame to the tag detection method for processing. This reduced framerate was implemented to allow sufficient time for the detection algorithm while still rendering graphics and displaying video in the window at the full framerate.

4.6.2 Simulation Environment

The simulation environment created by the Overwatch application supported an arbitrary number of virtual obstacles and targets. Figure 4-19 illustrates how the number, location, and size of these objects can dramatically change the appearance of the test environment as viewed in the application window. The obstacle and target maps used for training and user study trials were carefully designed to provide the appropriate level of difficulty and consistency.

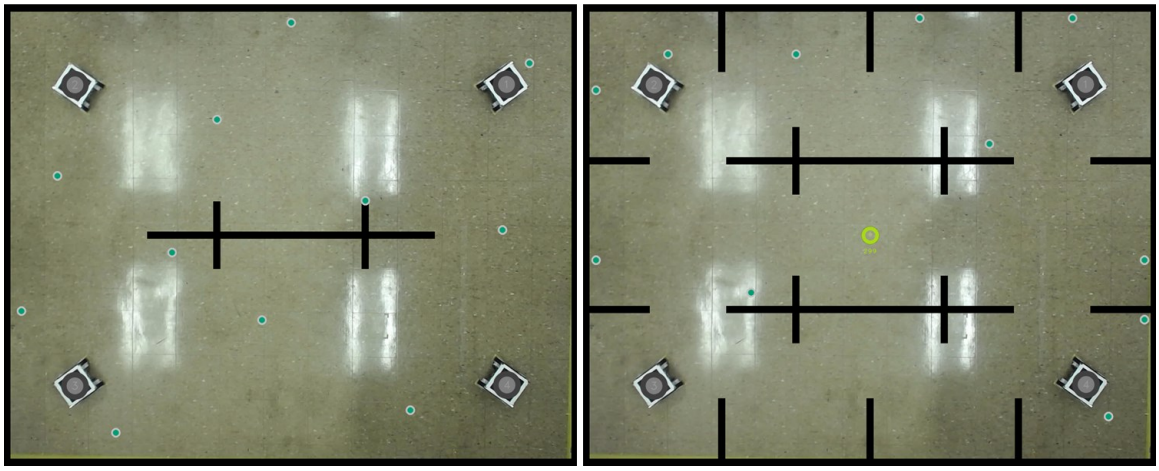

Figure 4-19: Example obstacle and target configurations.
Left: Fewer obstacles created more open spaces, which were used for operator training.
Right: More obstacles provided more challenges spaces to navigate and locate targets.

Obstacles and targets were defined by JSON configuration files which were specified at runtime as arguments when the application was launched. Figure 4-20 contains an example JSON object with a group named `border` for identification purposes which defines four obstacles. An arbitrary number of group objects like the example each contained an array named `rectangles` with an arbitrary number of obstacles specified as shown in the example. Each obstacle was defined by a `center` object with `x` and `y` values specifying the center point

coordinates, and a `size` object with `w` and `h` values specifying width and height. All values were in pixels. Targets were similarly defined by an arbitrary number of arrays named `coordinates` which contained an arbitrary number of objects with `x` and `y` values specifying target coordinates in pixels.

Configuring virtual objects at runtime avoided recompiling the Overwatch application to change obstacles and targets during development, testing, and designing user studies. Runtime configuration enabled the use of counterbalanced sets of targets for the user studies presented in Chapter 5.

```
{ "group": "border",
  "rectangles":[
    {"center":{ "x":    8, "y":  512 }, "size":{ "w":   16, "h": 1024 }},
    {"center":{ "x": 1272, "y":  512 }, "size":{ "w":   16, "h": 1024 }},
    {"center":{ "x":  640, "y":    8 }, "size":{ "w": 1280, "h":   16 }},
    {"center":{ "x":  640, "y": 1016 }, "size":{ "w": 1280, "h":   16 }}
  ]
}
```

Figure 4-20: Example JSON configuration for four virtual obstacles.
Virtual obstacles were defined by their center point (x, y) and dimensions (w, h).
Targets were similarly defined by their center point, but did not have size dimensions.

4.6.3 Robot Motion Planning and Collision Avoidance

The motion planning algorithm determined forward motions, turns, and pivot turns (rotations about the yaw axis) necessary to navigate a robot to its path vertices sequentially within configurable tolerances. The robot automatically executed these maneuvers using configurable motor speed values for each type of maneuver. Similar to the virtual obstacles and targets described in the previous section (4.6.2), motor speed values were defined at runtime using JSON formatted configuration files.

Figure 4-21 presents the robot collision detection algorithm, which monitored the location and motion of each robot and used this information to detect and respond to imminent

collisions between pairs of robots based on a number of rules. The algorithm automatically suspended the motion of robots to avoid collisions. The robots remained suspended until the collision was resolved by the operator (see 4.6.4.4 Resolving Collisions).

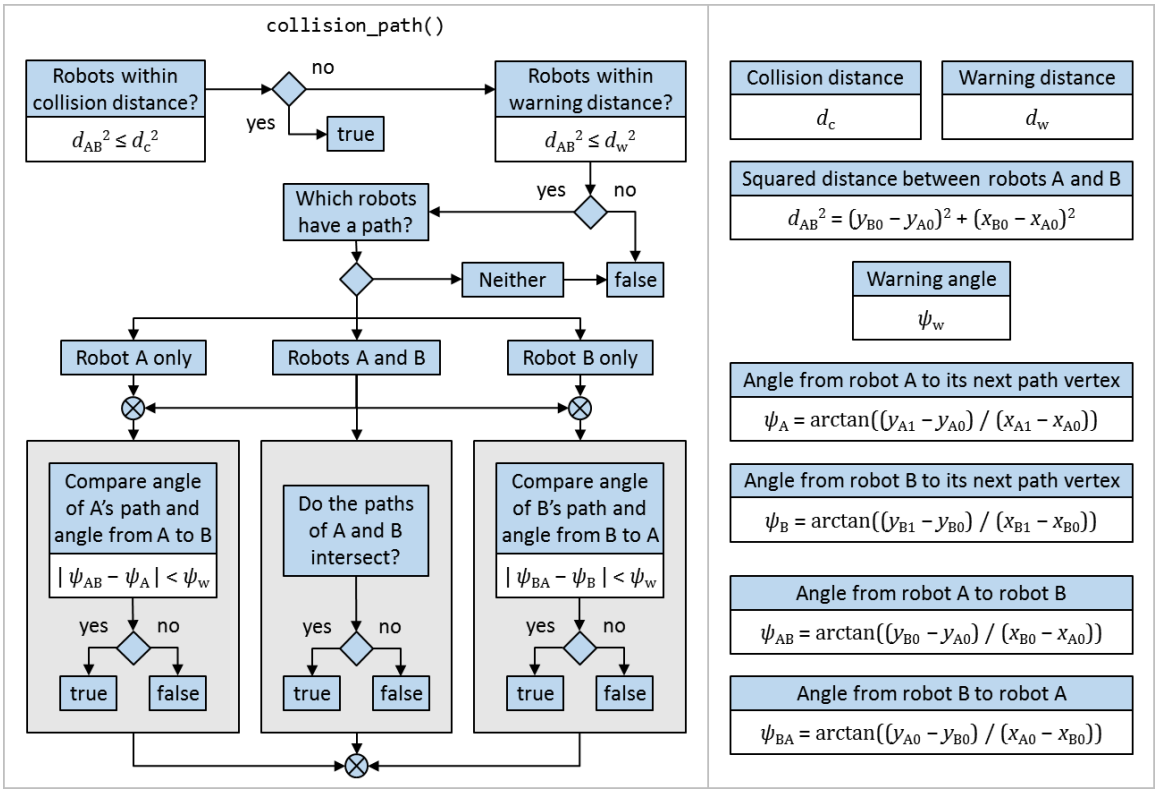

Figure 4-21: Robot collision detection algorithm.
A robot which had a navigation path was presumed to be in motion unless suspended by the collision detection algorithm.

4.6.4 Operator Input Commands

Table 4-7 contains operator input commands used to control the robots and completed search tasks. For simplicity, short labels were used for reference. Although all of the input methods can be broadly described as teleoperation, the *teleop* label was used to distinguish this method from the more automated *path* and *pivot* methods.

The operator inputted all commands using the computer mouse of the control interface to perform one or more of the following actions: move the mouse to position the on-screen cursor

(move), press and release the left mouse button (click), click twice in quick succession (double-click), press and release the right mouse button (right-click), or press and hold the left mouse button then move the mouse (drag).

Table 4-7: Operator input commands used to control robots

Input	Instructions
path	1. Click robot, move mouse to desired waypoint 2. Click to add waypoint to path and continue input 3. Double-click last waypoint to finish input 4. Right-click to cancel input
pivot	1. Double-click robot to rotate about the yaw axis 2. Robot rotates clockwise up to 360° or until stopped
stop	Click moving robot to stop and cancel path
teleop	1. Position cursor on robot 2. Press and hold down left button 3. Drag mouse 4. Robot will pivot and drive until reaching pointer 5. Release button to quit input
target	Click the intersection of target detection lines

4.6.4.1 Path Input

The *path* input method enabled the operator to input a navigation path for a robot. The operator planned and inputted a path as a series of one or more waypoints (i.e., path vertices). The operator positioned the mouse cursor over an idle robot (not in motion), then clicked to select it for path input. The operator then moved the cursor and clicked a desired waypoint to add a path vertex with the waypoint coordinates. The operator could input any number of vertices by clicking additional waypoints, or right-click at any time to cancel the path and exit path input mode. The operator double-clicked the last vertex to terminate the path at that point and complete input. The last vertex could also be clicked once to input it, then clicked a second time to complete input and terminate the path.

After path input was completed, the robot automatically executed forward motions, turns,

and pivot turns (rotations about the yaw axis) to visit each path vertex and stopped at the last vertex. The robot automatically paused if the motion planning algorithm determined a collision with another robot was imminent. While paused, the robot stopped moving but could resume motion if the collision was resolved (see 4.6.4.4 Resolving Collisions). The operator could click the robot while in motion or paused to stop it and cancel the path.

Figure 4-22 shows the graphics drawn with respect to the robot while in path input mode. Color-code circles and white lines represented path vertices (waypoints) and edges respectively. The center of the robot was considered the first path vertex. A circle was drawn at each subsequent vertex to highlight it. Each inputted vertex was white and continued to be drawn for the duration path input. A pending vertex (i.e., not yet inputted) was blue if valid or yellow if invalid. A vertex was invalid if it was too close to one or more obstacle (see 4.6.4.4 Resolving Collisions). A yellow line was also drawn around obstacles for which the vertex was invalid. After the operator completed path input by double-click the last vertex, a thinner line was drawn between vertices without the vertex circles.

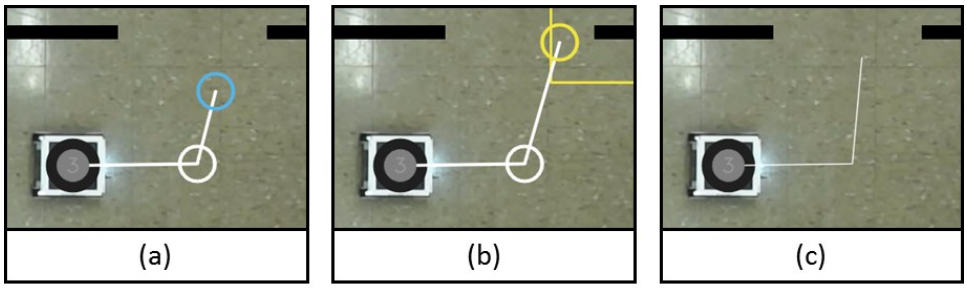

Figure 4-22: Overwatch path input commands.
Color-code circles and white lines represented vertices and edges respectively. Inputted vertices were white. A pending vertex was blue if valid (a) or yellow if invalid (b). A yellow line was also drawn around obstacles to which an invalid vertex was too close. A thinner line with no circles was drawn after path input was completed (c).

The path input method served two purposes. First, it avoided potential latencies and path inaccuracies of automatic path planning by making the operator responsible for planning and

inputting valid path vertices. Second, the cognitive effort and focus required to plan and input paths were desirable as contributions to the user study designs. The idea was to provide a level of autonomy that still required the operator to divide attention among the robots, switch between frames of reference, and integrate information from multiple sources. In other words, the path input method presented here was designed to introduce a common level of multirobot teleoperation challenges experienced by study participants across all test conditions.

4.6.4.2 Pivot Input

The *pivot* input method allowed the operator to command a robot to rotate about its yaw axis up to 360 degrees in a clockwise direction as viewed from above the robot. The operator positioned the mouse cursor over an idle robot (not in motion), then double-clicked to issue the command. As with the autonomous execution of path input, the operator could click the robot while in motion to stop pivoting.

This method was implemented to facilitate reorienting a robot to face a desired direction, and to search the immediate area for hidden targets by “sweeping” the sensor around the robot’s current location. Variations to enable pivoting counterclockwise were considered, such as using the right mouse button or holding the shift key. In the interest of minimizing input command complexity, only clockwise pivot was implemented.

4.6.4.3 Teleop Input

The *teleop* method used the mouse cursor as a dynamic goal point. The operator positioned the cursor over a robot, and pressed and held the left mouse button to start teleop input. The operator then moved the cursor to a desired goal point. The robot pivoted and moved forward as needed to reach the goal. The operator could drag the mouse to dynamically shift the goal point, even while the robot was in motion. The robot stopped upon reaching the goal (i.e., mouse cursor) or when the operator released the button.

Figure 4-23 shows the graphics drawn with respect to the robot while in teleop mode and their meaning. A white line was drawn from the center of the robot to the goal point. Robot motion corresponded with the intersection of this line with a color-coded ring that appeared upon the start of teleop input. The robot moved forward or reverse when the line intersected the white arc at the front or back of the robot respectively. The robot performed a pivot turn (rotation about the yaw axis) when the line intersected a blue or purple arc. The blue arcs pivoted the front of the robot toward the goal, while the purple arcs pivoted the back of the robot toward the goal.

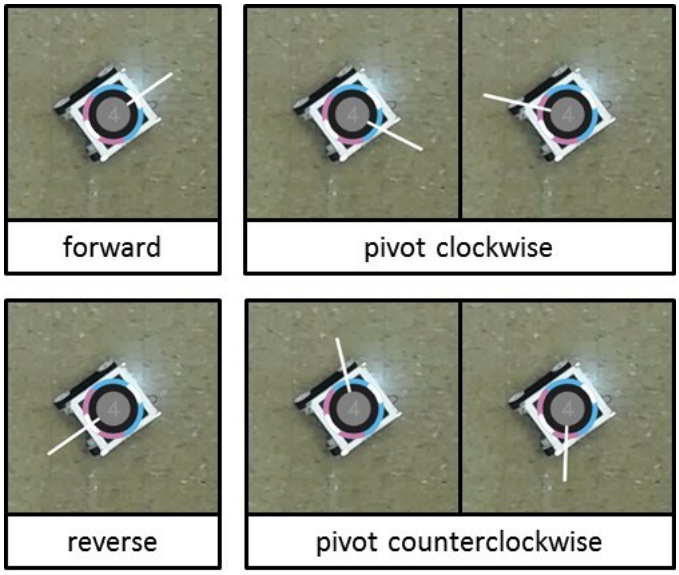

Figure 4-23: Overwatch teleoperation input commands.
The front of the robot in these pictures was facing the top-right corner of the picture.

The intent of having two pivot directions was to make the controls more intuitive. For example, if the angle of the line increased or decreased as a robot moved forward or reverse toward a goal point, the line could eventually cross into a pivot arc. When this happened, one would expect the robot to pivot toward the goal and then resume moving forward or reverse. Pivoting toward the goal meant aligning the front of the robot toward the goal if it was originally moving forward, or the back of the robot if it was moving in reverse. Thus, the expected

direction of rotation depended on whether the goal was in front of or behind the robot. The blue arcs were slightly longer than the purple arcs to bias pivoting in the “forward” direction.

The teleop input method provided a means of maneuvering within relatively tight spaces in the test environment, and enabled the operator to resolve two types of collisions without intervention by the system administrator. These collisions and how teleop is used to resolve them are discussed in the next section.

4.6.4.4 Resolving Collisions

Obstacle collisions were detected and avoided during operator input. Each obstacle was surrounded by an *exclusion zone*, the boundary of which was a configurable distance away from the obstacle’s edges. The path input algorithm rejected any point inside the boundary. Once a valid path was inputted however, the robot operated within motion constraints but was otherwise free to execute the path without any constraints on its location. Because a robot could and often did cross into an exclusion zone while operating autonomously, it would occasionally end its path within the boundary. When this happened, the operator was unable to input another path for the robot because the first path vertex (the robot’s location) was inside the exclusion zone and thus an invalid vertex.

Teleop input was configured with a greater tolerance than path input when near obstacles. This allowed the operator to use teleop input to click and “drag” a robot out of the path input exclusion zone, thus resolving the obstacle collision. Although the operator was still prevented from moving the robot closer to an obstacle during teleop input, the algorithm did allow pivoting. This enabled the operator to use the full range of input angles around the robot, even when in close proximity to an obstacle.

Teleop input also helped resolve robot collisions. The motion planning algorithm automatically suspended a robot’s motion to avoid collision with another robot. A suspended

robot stopped moving but retain its path. Once the collision was resolved, the robot then resumed execution of the path. As shown in Figure 4-24, the operator resolved a collision by moving one of the robots away from the other robot. The collision algorithm would not allow a suspended robot to receive path input, but did allow teleop input. An operator could click and "drag" one of the (nearly) collided robots away from the other. The latter robot would then resume its path or, if idle, be able to receive path input.

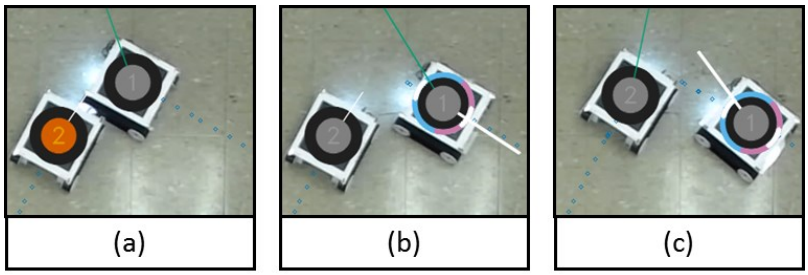

Figure 4-24: Teleop input was used to resolve robot collisions. An orange filled circle indicated the collision detection algorithm suspended a robot (a). The operator had to use teleop input to move one of the robots away to resolve the collision (b). Afterward the other robot would resume its path (c).

4.6.4.5 Target Localization

The operator was responsible for locating hidden virtual targets as a search task. A target had to be detected by two or more robots before it could be located. Fixed-length green lines from the robot through the target identified the direction of the target but not its precise location. The intersection of multiple detection lines provided the operator with the required information. Figure 4-25 shows a target detection and localization sequence. The operator positioned two robots near the detected target to reveal its location at the intersecting green lines. The operator clicked the intersection to report the target location. A green and gray circle was drawn to indicate the target was located. This circle persisted until the trial timer expired or the Overwatch application was closed.

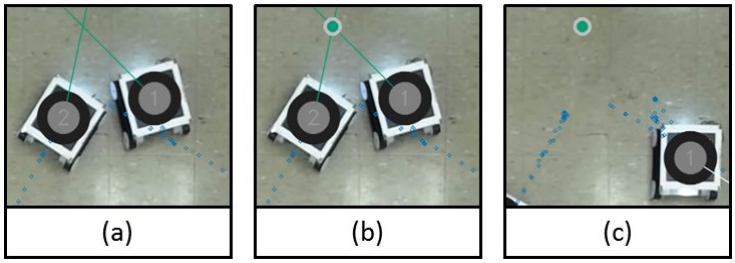

Figure 4-25: Target detection and localization.
A target had to be detected by two or more robots before it could be location (a).
The operator reported the location by click the intersection of the green line,
after which a green and gray circle appear to indicate the target was located (b).
The target marker persisted after the robots moved away (c).

4.6.5 Administrator Commands

In addition to the operator input commands, the control interface accepted a number of administrator commands used to manage the test platform during user studies and to access special features during development and system testing. These commands are found in Table 4-8. Of these, only the robot shutdown and tutorial commands were necessary during normal operation of the platform.

The robot shutdown command triggered a TCP message telling the target robot to halt motion, stop processing further input, and perform an orderly shutdown of the onboard operating system. The combination of key presses and mouse input greatly reduced the likelihood of accidentally shutting down a robot. The command started with pressing and holding the `Ctrl` and `Alt` keys, similar to the `Ctrl` + `Alt` + `Del` sequence familiar to users of the Windows operating system. Mouse input was used to enable selecting a specific robot to shut down. Finally, the command was completed by pressing and releasing the right mouse button, which was used in lieu of the left button as an added measure to prevent accidently issuing the command.

The tutorial commands were used to advance one step or go back one step while the test

platform was in tutorial mode. This mode was only used at the beginning of a user study session to familiar participants with the operator input controls. Other administrator commands included inputs to view help and information panels. These were primarily used for system development and testing. The robot pause command was enabled to allow the administrator to stop a robot if an anomaly occurred or to prevent damage to a robot. This command was not used during any user study trial. Finally, the map editor commands were used during platform development to design obstacle maps. Details pertaining to map editor functionality are not presented here because there are not directly relevant to the research aim and objectives.

Table 4-8: Administrator commands used manage the test platform and access special features

Command	Method	Instructions
help	keyboard	Press `Space` to show/hide robot commands Press F1 to show/hide admin commands
map editor	key + mouse	Hold `Alt` key to display mouse coordinates Hold `Alt` + left-click and drag to create and size object Hold `Shift` + `Alt` + left-click and drag to copy object Hold `Ctrl` + `Alt` + right-click to delete object
robot pause	keyboard	Press number key to pause/resume respective robot
robot shutdown	key + mouse	1. Press and hold `Ctrl` + `Alt` 2. Right-click robot
tutorial	keyboard	Press right arrow key to advance to next step Press left arrow key to go back to previous step
view panels	keyboard	Press `C` or `K` to show/hide color key Press `P` to show/hide color palette Press `S` to show/hide status panel

4.6.6 Status Panel

The window displayed a status panel before and after each user study trial. The purpose of this panel was to inform the system administrator about connection and calibration status of the eye tracker, and the connection and fiducial tag tracking status of each robot. Figure 4-26 illustrates the color-coded status conditions. All icons were blue if no problem was detected. The eye tracker icon used gray to indicate connection issues and red to indicate the calibration

results were outside the configured limits. There were two icons next to each robot; a square icon for fiducial tag tracking and a wireless symbol icon for TCP connection, with gray used to indicate problems with robot localization or communication respectively.

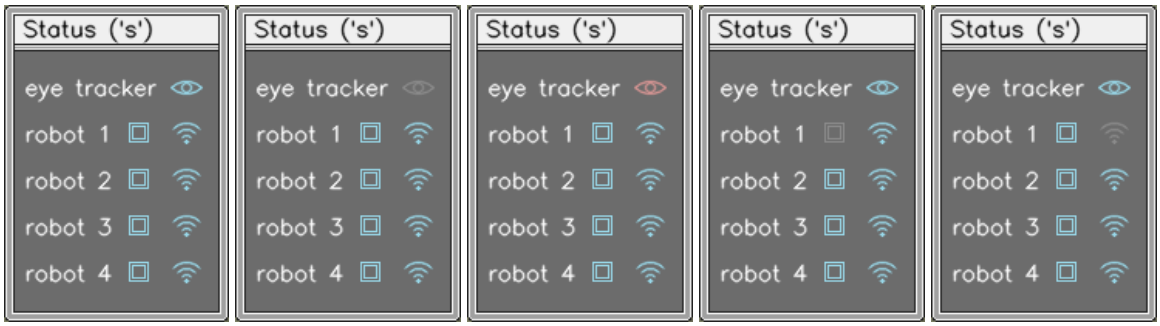

Figure 4-26: Status panel with eye tracker, robot tracking, and robot connection icons. Blue icons indicated ideal eye tracker and robot conditions (a). The panel used gray to signal problems with the eye tracker connect (b) robot localization (d) and robot TCP connections (e). The eye tracker icon used red to indicate calibration results outside the configured limits (c).

4.6.7 Data Logging

During each user study trial, the Overwatch application collected, processed, and outputted a large volume of data to several log files in comma separated value (CSV) format. The name and relative directory path of these files was determined at runtime based on the options specified by session configuration file. The most important data were trial-level state and performance information appended to three session logs at the conclusion of each trial:

1. The trial log recorded trial identification and task performance.
2. The robot log recorded aggregate state, distance, and confidence.
3. The search log recorded detailed search task results.

These data along with detailed robot state information recorded at 2 Hz supported the analyses detailed in Chapter 5. In addition, Overwatch also produced event logs, recorded eye tracker calibration and runtime configuration, and captured eye gaze data at 20 Hz.

4.7 Host System and Build Environment

The control interface computer was equipped with an Intel Core i7-4500U quad-core CPU at 1.8 GHz (4MB cache, up to 3.0 GHz single core), 100 MHz front-side bus, 8 GB DDR3L SDRAM up to 1600 MHz, and Microsoft Windows 7 Professional 64-bit operating system. This machine also hosted the development environment used to author, integrate, build, test, and release the control interface software.

The Overwatch application (`overwatch.exe`), Eyelib static library (`eyelib`), and other static library dependencies were written in C++ and built using MinGW-w64 version 5.3.0 and Code::Blocks Integrated Development Environment (IDE) version 13.12. The MinGW-w64 toolchain included Windows ports of the GNU Compiler Collection (GCC) compiler and GNU Binutils, supported by Windows-specific header files and static import libraries.

Table 4-9 lists software dependencies required in order to build the control interface software. The principal external dependencies were the AprilTag [119], [120] library used for robot localization and tracking, Asio [124] for asynchronous TCP communication, JSON for Modern C++ (`nlohmann/json`) [125] used to parse configuration files, and OpenCV (Open Source Computer Vision Library) [127]. OpenCV libraries were used by Overwatch to capture and process video frames, capture keyboard and mouse input, create the application window, render graphics, and display processed video frames on the screen.

Table 4-9: Control interface software dependencies

Dependency	Library	Description
AprilTag	`apriltag2`	Fiducial marker tracking and homography
Asio	`asio`	Asynchronous TCP communication https://think-async.com/Asio/
Eigen	`Eigen`	C++ template library for linear algebra http://eigen.tuxfamily.org/ Used by AprilTag
eyelib	`eyelib`	Eye tracking library
FLTK	`fltk`	Fast Light Toolkit (FLTK); Required by eyelib library
JSON for Modern C++	`nlohmann/json`	JavaScript Object Notation (JSON) Niels Lohmann https://github.com/nlohmann/json
OpenCV 3.0.0	`core`	Core Functionality
	`calib3d`	Camera Calibration and 3D Reconstruction
	`hal`	Hardware Acceleration Layer
	`highgui`	High-level GUI and Media I/O
	`imgcodecs`	Image File Reading and Writing
	`imgproc`	Image Processing
	`videoio`	Media I/O
OpenCV 3rd-party	`libjpeg`	Reading/writing JPEG images
	`libpng`	PNG reference library
	`libtiff`	Reading/writing TIFF files
	`zlib`	LZ77 data compression
utility libraries	`utl`	Header-only utility libraries
Windows API	`comctl32`	Common Control Library
	`comdlg32`	Common Dialog Box Library
	`gdi32`	Graphics Device Interface (GDI)
	`ole32`	Component Object Model
	`oleaut32`	OLE Automation
	`uuid`	Universally Unique Identifier
	`vfw32`	Windows Multimedia
	`ws2_32`	Winsock 2
	`wsock32`	Winsock 1

4.8 Discussion

4.8.1 Results

The presented multirobot test platform incorporated a number of experimental controls for the user studies in the next chapter. The tracked vehicle design resulted in four maneuverable robots with ample power and consistent turning clearances afforded by aligning the yaw axis with the geometric center of the platform. This mitigated potential user study variability due vehicle dynamics. A dedicated wireless network ensured low-latency communication between the robots and the control interface. A 90-degree field of view from the overhead camera produced video of the relatively large test environment (11 m^2) with limited distortion.

The control interface software made extensive use of configuration files and automation to control many aspects of user study administration and data collection. Automated study sessions ensured the delivery of study information and instructions, facilitated consistent calibration and training, and accurately executed study trials using predefined configuration options and counterbalanced conditions. Automatic data logging ensured study results were accurately recorded and formatted to support analyses.

The principal control interface software component was the Overwatch application, the central hub around which the rest of the platform was built. This was supported by the Eyelib library, which managed and provided access to eye tracking functionality. Modern C++ language features and programming tools were leveraged to produce reliable software that operated consistently with minimal latency. Colors were carefully selected to design a palette that supported the research objectives and accessibility.

Figure 4-27 shows summary source code statistics from the `overwatch` and `eyelib` projects. Special attention was given to documenting the code for both future work and reusability for other work. The source files contained extensive comments to document the code. Doxygen

[128] was used to generate documentation in HTML format using tags and markdown within the comments.

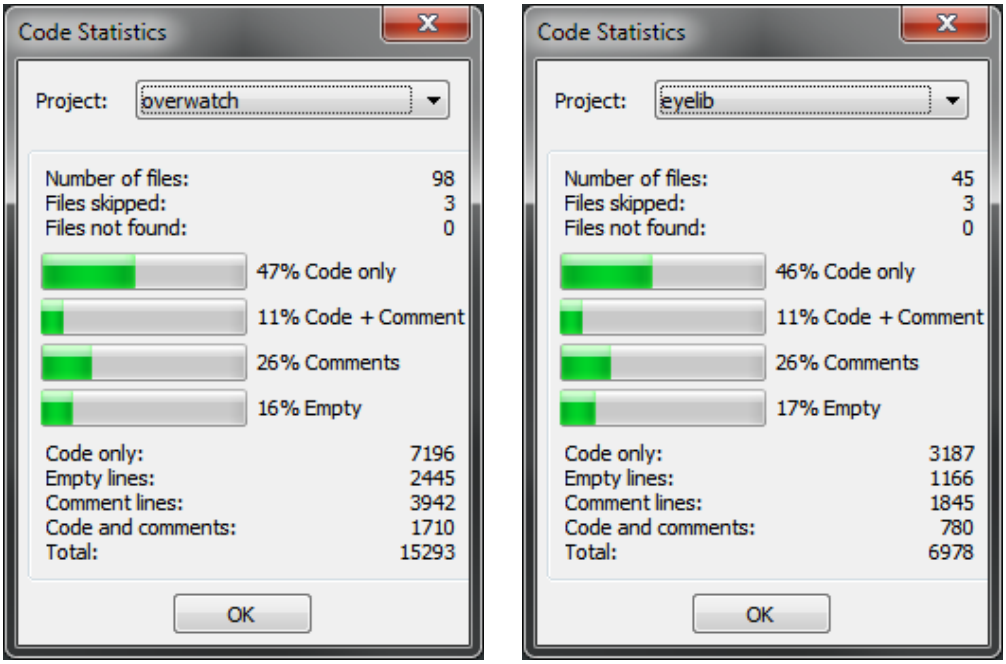

Figure 4-27: Overwatch and Eyelib software code statistics.
The `overwatch` source contained 8,906 lines of code spread across 98 files (left).
The `eyelib` source contained 3,967 lines of code spread across 45 files (right).
Note: These figures did not include AprilTag, Eigen, JSON, and other third-party libraries, nor did they include custom general-purpose header-only utility library files

4.8.2 Simulation Environment

The test platform integrated physical robots in an open obstacle-free test environment with virtual objects and sensing in a computer simulation environment, and employed augmented-reality to present a composite view of these physical and virtual objects in the real-world. The operator was given path planning responsibility, but the software used accurate estimates of robot location from fiducial tag localization and perfect knowledge of obstacle location and size to validate each path vertex inputted before accepting it. The motion planner and robots were responsible for executing planned paths within operating constrains, but did not have to consider and were not affected by the virtual obstacles. Search targets were also virtualized, and

a virtual sensor to detect these targets was assigned to each robot.

This simulation approach yielded several advantages:

1) Virtualization avoided accidentally disturbing physical objects during study trials.
2) The obstacle-free test environment eliminated possible effects of obstacle interference.
3) Path input validation reduced the likelihood of effects due to unreliable path planning.
4) Precise obstacle maps enabled control of obstacle pattern symmetry.
5) Target virtualization facilitated randomization of target placement.
6) Virtual target sensing eliminated possible effects of sensor accuracy and reliability.
7) Predefined obstacle and target maps enable repeatable user study conditions.
8) Virtualization enabled rapid reconfiguration to change study conditions between trials.

4.8.3 Physical Robots

While virtualizing the entire test platform was considered, there are advantages to maintaining a physical test environment and robot platforms. Physical robots, even in a laboratory setting, provided realism that may be difficult to achieve in a full simulation. It may be more cost effective and time efficient to implement a physical solution. For example, mounting a video camera on the ceiling and placing tape lines on the floor were all that was required to establish the test environment for this research. A pragmatic engineering approach was used to leverage the strengths and mitigate the weaknesses of physical and virtual elements in order to design and implement the platform presented here.

The presence of physical robots and other hardware also contributed to recruitment of study participants and retention of participants who returned for multiple sessions. The robots generally triggered curiosity and questions from participants and other visitors to the lab, including prospective undergraduate students.

4.8.4 Eye Tracking

Eye tracking capability was integrated into the test platform in order to support the real-time estimation of operator attention. Captured eye gaze data were processed online to meet this objective. Further details about this processing are found in Chapter 5. In addition, these data were also output to log files at 20 Hz to facilitate future work such as post-hoc analysis of pupil size measurements.

4.9 Conclusion

The presented work has provided several key components of teleoperation and control for multirobot systems. The control interface software enabled a single human operator to control a team of ground robots in real time in a collaborative manner, balancing automation and human interaction to perform search tasks in a challenging environment. This work provides an easily translatable augmented-reality interface capable of coordinating both ground and aerial robots in complex environments for applications in a variety of domains including space exploration, border security, homeland security, defense, search and rescue, and first responder events in hazardous conditions [5], [30-35].

The resulting system integrated physical robots in a combined live and virtual environment. Integrated eye gaze tracking and data processing were employed to assess operator attention in real time. This enabled the development of techniques incorporating physiological feedback and the evaluation of these techniques presented in the next chapter.

CHAPTER 5: ROBOT CONFIDENCE AND TASK PERFORMANCE

5.1 Introduction

This chapter details techniques applying physiological feedback to human operation of multiple mobile robots, and user studies to evaluate these techniques in a controlled setting with physical robots in a combined real and virtual environment enabled by the test platform described in Chapter 4. Specifically, a model of *robot confidence* is presented which estimated operator attention to derive a confidence value, which was then used to select robot behaviors according to specified threshold values. This model and a number of adaptive behaviors were implemented in the test platform software. Three user studies were conducted to examine the effects of these behaviors in relation to search task performance and efficiency.

5.2 Motivation

Human supervision and control of multiple mobile robots involves divided attention, multiple frames of reference, and the integration of information from many sources. The ability to assess the cognitive state of the operator in real time and use this information as feedback could lead to new methods of evaluating and interacting with multirobot systems.

5.3 Robot Confidence and Behavior

5.3.1 Conceptual Overview

The overall aim of this research is to develop techniques incorporating operator attention as input for teleoperation interfaces in order to enable effective and efficient control of multiple mobile robots. Pursuant to this aim, a robot confidence model with indicators of operator attention as input was defined and implemented to vary robot behavior in the multirobot test platform. The term *robot confidence* was used as a metaphor to describe the mapping of attention-related inputs to robot behaviors.

Figure 5-1 contains a conceptual diagram of the generalized robot confidence model defined in the next section. This diagram illustrates how the model transforms indications of operator attention into adaptive robot behaviors. Human attention is a cognitive function, making it difficult to measure. The simple sense-think-act model on the left side of the figure depicts selective attention applied to perception, processing information, and responding to the processed information. Although the processes of cognition were not incorporated into the model, they are shown in Figure 5-1 to provide context. Also note the model operates on discrete time as described later.

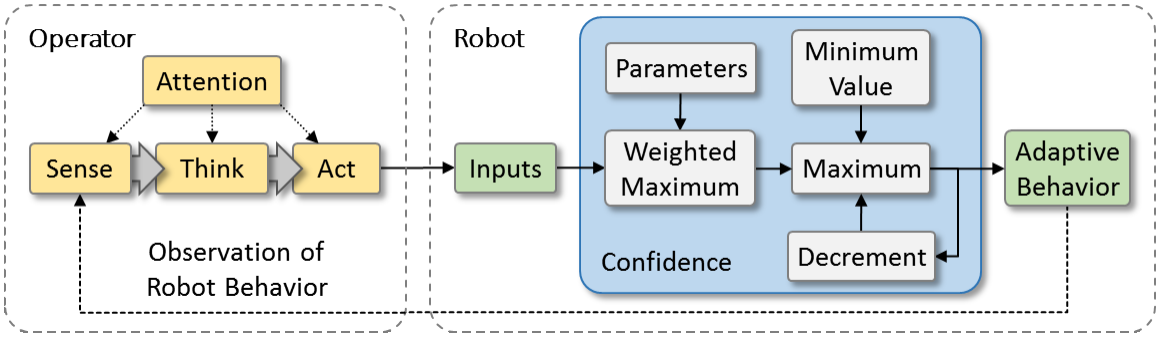

Figure 5-1: Telerobot confidence and behavior model.

A number of Operator actions (Act) may be outward indications of Attention, including eye gaze fixations near the robot and input commands issued to the robot. These are generalized in the diagram as the green Inputs block. A Weighted Maximum described in the next section of this chapter combines these inputs with confidence Parameters to produce a single confidence input. This is compared along with a Minimum Value and the Decrement block result, and the Maximum is taken as the confidence value for the current timestep. The resulting confidence value is used to determine the robot's Adaptive Behavior, and is also decreased by Decrement to provide feedback for the next timestep. Finally, Observation of Robot Behavior by the Operator can potentially influence Attention and future actions.

5.3.2 Generalized Confidence Model

For n confidence model inputs in $\boldsymbol{x}$ and corresponding parameters in $\boldsymbol{p}$, Equation 5-3 computes a maximum weighted input u, where $\boldsymbol{p} \circ \boldsymbol{x}$ is the Hadamard (element-wise) product of $\boldsymbol{x}$ and $\boldsymbol{p}$. Equation 5-4 defines robot confidence c_k at timestep k, where c_d is a confidence decrement subtracted from the previous confidence value c_{k-1} and c_min is a minimum confidence value (e.g., 0).

$$\boldsymbol{x} = [x_1 \dots x_n] \tag{5-1}$$

$$\boldsymbol{p} = [p_1 \dots p_n] \tag{5-2}$$

$$u = \max(\boldsymbol{p} \circ \boldsymbol{x}) = \max([p_1 x_1 \dots p_n x_n]) \tag{5-3}$$

$$c_k = \max(u_k, c_{k-1} - c_\mathrm{d}, c_\mathrm{min}) \tag{5-4}$$

Figure 5-2 illustrates confidence during a notional sequence of inputs and parameter values. This example employs an input vector $\boldsymbol{x} = [x_1 \ \ x_2]$ with two binary inputs $x_1, x_2 \in [0,1]$, two associated confidence parameters $\boldsymbol{p} = [p_1 \ \ p_2] = [25 \ \ 10]$, confidence decrement $c_\mathrm{d} = 10$, and minimum value $c_\mathrm{min} = 0$.

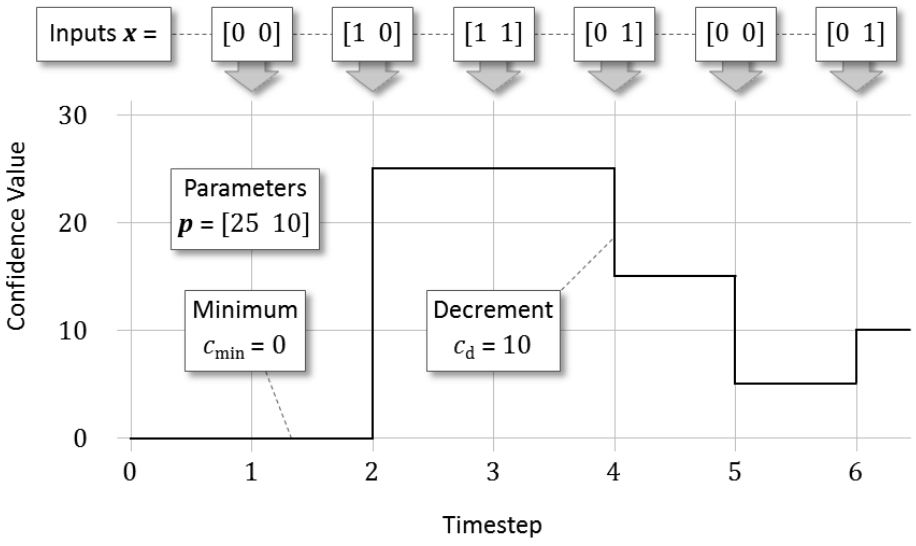

Figure 5-2: Illustration of robot confidence given notional parameter values and inputs.

Starting with initial robot confidence $c_0 = 0$, the example in Figure 5-2 depicts:

1. At timestep 1, Equation 5-3 with model input $\boldsymbol{x} = [0\ \ 0]$ results in a maximum confidence input of $u = \max([p_1 x_1\ \ p_2 x_2]) = \max(25 \cdot 0, 10 \cdot 0) = 0$. Equation 5-4 then yields robot confidence $c_1 = \max(u, c_0 - c_\mathrm{d}, c_\mathrm{min}) = \max(0, 0 - 10, 0) = 0$. Note that taking the maximum value prevents confidence values below c_min.
2. At timestep 2, $\boldsymbol{x} = [1\ \ 0]$ results in $u = \max(25 \cdot 1, 10 \cdot 0) = 25$ and robot confidence $c_2 = \max(25, 0 - 10, 0) = 25$.
3. At timestep 3, $\boldsymbol{x} = [1\ \ 1]$ results in $u = \max(25 \cdot 1, 10 \cdot 1) = 25$ and robot confidence $c_3 = \max(25, 25 - 10, 0) = 25$. Note that multiple instances of the same input u at consecutive timesteps *sustain* confidence rather than increase it.
4. At timestep 4, $\boldsymbol{x} = [0\ \ 1]$ results in $u = \max(25 \cdot 0, 10 \cdot 1) = 10$ and robot confidence $c_4 = \max(10, 25 - 10, 0) = 15$. Because $(c_3 - c_\mathrm{d}) > u$, the resulting confidence value is the decremented previous confidence value rather than the maximum weighted input.
5. At timestep 5, $\boldsymbol{x} = [0\ \ 0]$ results in $u = \max(25 \cdot 0, 10 \cdot 0) = 0$ and robot confidence $c_5 = \max(0, 15 - 10, 0) = 5$.
6. At timestep 6, $\boldsymbol{x} = [0\ \ 1]$ results in $u = \max(25 \cdot 0, 10 \cdot 1) = 10$ and robot confidence $c_6 = \max(10, 5 - 10, 0) = 10$. Note that confidence does not increase by the value of the maximum weighted input (10). Instead, it only increases by 5 to reach a value of 10.

A maximum value was used for Equation 5-3, and again for Equation 5-4. These were incorporated in order to accommodate eye gaze fixations as input. The duration of a single fixation could be less than 100 milliseconds, and multiple fixations on or near an object of interest can occur within the time span of other forms of input, such as a mouse button press

and release. Other methods of aggregating inputs, such as a weighted sum, may result in fixations and other higher frequency inputs having a stronger influence on confidence than lower frequency inputs.

Although confidence parameters in $\boldsymbol{p}$ could be used offset this imbalance, a second bias is introduced by the inherent variability of fixation events. Focused attention is not necessarily accompanied by a single fixation. Instead, multiple fixations are likely. Attention for a given duration may result in a relatively small number of long fixations, or a higher number of short fixations. Individual operator differences and the design of fixation detection algorithms also contribute to variability in fixation counts and durations. A weighted sum or similar method would produce biased outputs favoring a high number of short fixations over fewer long fixations. Similar biases may also occur if other physiological properties were included as input, such as pupil diameter.

The maximum-value approach makes selective use of the available information to determine the confidence value. More sophisticated methods might be used to address potential biases and take full advantage of the available information. This researched started with a simple model in order to avoid complexity and over-optimization before the potential utility of the presented techniques were evaluated. The results detailed later in this chapter justify future work to potentially build on the model, as discussed in the next chapter.

The operator could attend to things other than a robot. For example, looking for navigation breadcrumbs to determine which areas of the test environment have been searched. The confidence model does not account for attention not directed at a robot. Instead, if the operator attends to something besides a robot, the confidence of all robots decreases accordingly.

5.3.3 Model Implementation

For this research, attention directed at a robot was of interest. In reference to Figure 5-1, that occurs when operators observe (Sense), process information about (Think), or interact (Act) with a given robot. The multirobot test platform and specific operator tasks implemented were designed to fully occupy the operator's vision and demand frequent input. In other words, the system left little to think about a robot without simultaneously looking at or issuing commands to it. Thus, fixations near a robot and input commands were considered proxies for attention.

Algorithm 1 describes how the robot confidence model was implemented in the multirobot test platform using eye gaze fixation and user input as indicators of operator attention. The algorithm was a condensed version of Equation 5-3 and Equation 5-4 with confidence decrement $c_{\mathrm{d}} = \Delta t = (t_k - t_{k-1})$, minimum confidence $c_{\min} = 0$, and confidence parameters $\boldsymbol{p} = [p_{\mathrm{g}} \;\; p_{\mathrm{u}}]$, where p_{g} and p_{u} corresponded to fixations and user input respectively.

Algorithm 1: Robot confidence implementation
Input: Number of robots $R \in \mathbb{N}$, robot index $r \in \mathbb{N}$, confidence $c_r \in \mathbb{N}$, eye gaze fixation parameter $p_{\mathrm{g}} \in \mathbb{N}$, user input parameter $p_{\mathrm{u}} \in \mathbb{N}$

```
for r = 1 to R
    if user input received for r then
        c_r ← p_u
    else
        Δt ← time duration since last update in seconds
        c_r ← max(c_r – Δt, 0)
        if operator fixated r then
            c_r ← max(c_r, p_g)
        end if
    end if
end for
```

The confidence value of each robot was updated at regular time intervals. If the robot received user input during the interval, its confidence was set to p_{u}. Otherwise, confidence was decremented by the number of seconds since it was last updated (Δt), or set to 0 if confidence $< \Delta t$. The algorithm then set confidence to p_{g} if the robot was fixated and confidence $< p_{\mathrm{g}}$,

otherwise confidence remained at the decremented value. Thus, robot confidence increased due to operator attention (fixations and user input), and otherwise decreased over time until the robot was attended again.

Equation 5-5 describes the model implemented by Algorithm 1. For a given robot at timestep k, confidence was potentially influenced by two binary indicators of attention: eye gaze fixation near the robot $x_{\mathrm{g}} \in [0,1]$, and user input to the robot $x_{\mathrm{u}} \in [0,1]$. All other variables are defined above.

$$c_k = \begin{cases} \max\left(c_{k-1} - c_{\mathrm{d}}, p_{\mathrm{g}}\right), & x_{\mathrm{g}} \wedge \neg x_{\mathrm{u}} \\ p_{\mathrm{u}}\,, & x_{\mathrm{u}} \\ \max(c_{k-1} - c_{\mathrm{d}}, c_{\min}), & \neg x_{\mathrm{g}} \wedge \neg x_{\mathrm{u}} \end{cases} \tag{5-5}$$

Although user input could indicate attention on its own, the operator was unlikely to command a robot without an accompanying fixation. However, because user input indicates focused attention while fixations may or may not, the model implementation accommodated assigning a higher confidence value for the former. This was accomplished by the first $\max$ function in Equation 5-5.

Algorithm 1 was defined in and used by the `core` module of the Overwatch application. A number of related data structures were defined in the `config` module because the confidence value parameters for eye gaze fixation and user input, as well as the robot behaviors which could optionally be affected by confidence, were all configurable at runtime. These structures were used by the `robot` module. Figure 5-3 contains a simplified diagram of relevant structures, properties, and relationships. Each robot had its own `Confidence` object which not only maintained its current confidence level and value, but also keep a running sum total duration the robot was operated at each confidence level. Each robot also had a copy of the confidence model parameters, and parameters related to alert and motor speed behaviors.

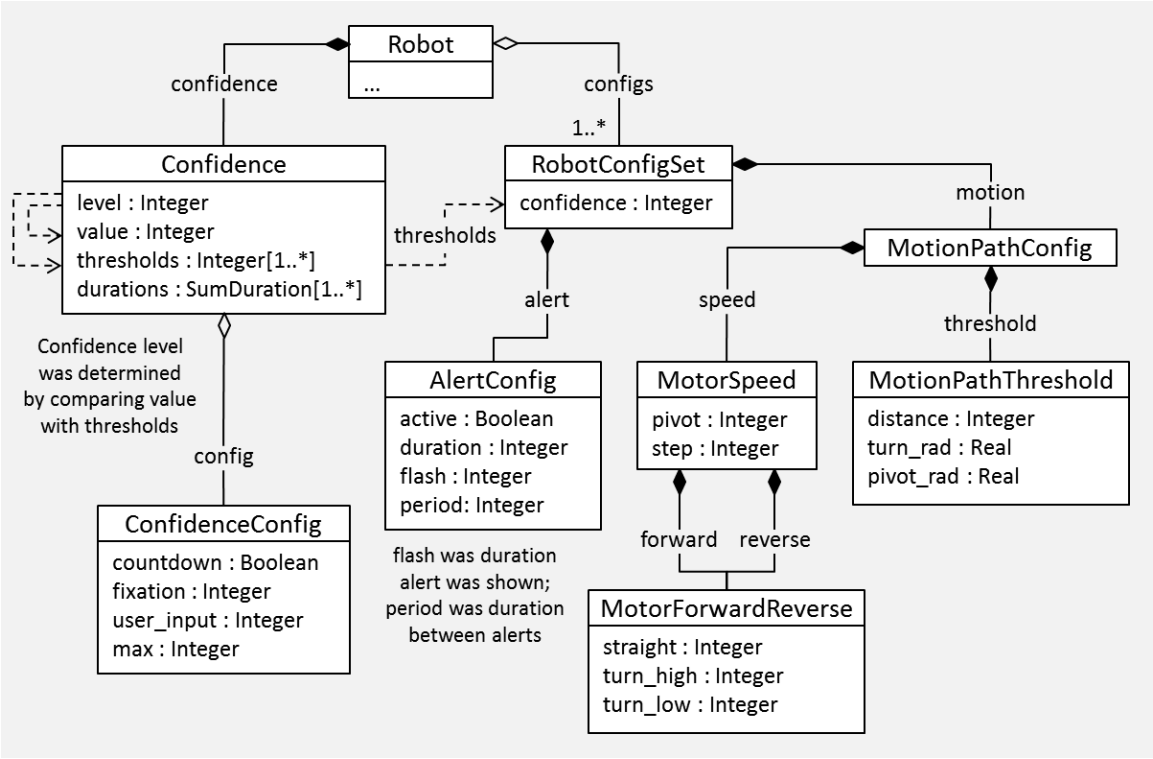

Figure 5-3: Structure diagram of robot confidence and behavior modification as implemented in the multirobot test platform.

5.3.4 Robot Behavior

On the far right of the diagram in Figure 5-1 is a single block labeled Adaptive Behavior. This block represents any number of behaviors that could be affected by confidence. The behaviors implemented for this work were a visual alert graphic and robot velocity (referred to as speed in Figure 5-3 above). The visual alert was a bright orange circle drawn at the robot's location. The alert could be configured to flash for an optionally specified duration, or be shown continuously on the screen. This behavior was intended to drawn the operator's attention toward a robot during a low-confidence state. The robot velocity behavior optionally changed motor speed and velocity-related motion planning parameters in response to robot confidence.

Alert and velocity behaviors could be assigned to specified confidence *levels*. A confidence level was simply a threshold confidence value which could trigger some change in behavior. This

thresholding approach was selected to minimize complexity before the potential utility of confidence and behavior modeling was evaluation. Future work could include continuous behaviors that covary with confidence value.

The Overwatch application was capable of supporting an arbitrary number of confidence levels at which a robot could operate. Like the confidence model parameters, behavior parameters were specified at runtime via JSON formatted configuration files. The specific configurations used for this research are covered in the next section.

5.4 Methods

5.4.1 Search Task Performance and Efficiency

Two measures of search performance and one measure of search efficiency were defined:

- Normalized search targets detected $d \in [0,1]$
- Normalized search targets located $s \in [0,1]$
- Search task efficiency $\eta \in [0,1]$

For a total number of discoverable search targets $n \in \mathbb{N}_{>0}$, the count of search targets detected was defined as $n_{\mathrm{d}} \in [0..n]$ and the count of search targets located was defined as $n_{\mathrm{s}} \in [0..n_{\mathrm{d}}]$. Both n_{d} and n_{s} provided measures of performance relative to n. To produce more generalized results, Equation 5-6 and Equation 5-7 were used to calculate normalized performance metrics.

$$d = \frac{n_{\mathrm{d}}}{n} \in [0,1] \tag{5-6}$$

$$s = \frac{n_{\mathrm{s}}}{n} \in [0,1] \tag{5-7}$$

Equation 5-8 was used to calculate search task efficiency, where $m_{\max} > 0$ was the maximum motor speed of the robots. This equation rewarded the localization of more targets through the inclusion of s, but penalized the amount of energy expended to locate them by

using average motor speed $m \in [0, m_{\max}]$ as a proxy for total energy used. Since s was already normalized, only the speed component had to be normalized such that $\eta \in [0,1]$, which was accomplished via the $(m_{\max} - m)/m_{\max}$ term.

$$\eta = s \cdot \frac{m_{\max} - m}{m_{\max}} \in [0,1] \tag{5-8}$$

5.4.2 User Study Design

Three user studies were conducted to evaluate search task performance and efficiency with various robot behaviors determined by the confidence model:

- Study 2: Visual Alert
- Study 3: Robot Velocity
- Study 4: Time of Day

Recall that Study 1: Levels of Autonomy was described in Chapter 3. All studies were conducted in accordance with the HIC approval found in Appendix B.

Individual spatial orientation ability of ground robot operators has been correlated with improved target-mapping performance and decreased workload [129]. Spatial orientation describes the ability to visualize and mentally manipulate objects in two or three dimensions. A significant relationship has also been observed between sense-of-direction and target search task performance. Perhaps more importantly, there are indications that spatial ability has a non-uniform association with performance under varying interface modalities [130], [131].

Like the Levels of Autonomy study discussed in Chapter 3, a within-subjects design was selected to mitigate participant variations such as spatial ability by collecting repeated measures with the same participant at all levels of the experimental condition. A number of protections against order effects were implemented:

1. The conditions were tested in counterbalanced order.

2. Participants were presented with on-screen instructional material, received hands-on training, and completed self-paced practice exercises to develop proficiency with the controls and search task.
3. Participants were encouraged to take breaks between trials.
4. Bottled water was offered to prevent dehydration and provide break opportunities.
5. Participants were informed about the study session duration in advance, and sufficient time was allocated to avoid rushing.
6. The study administrator monitored participants for signs of fatigue, frustration, or lack of engagement and offered breaks, training reminders, or encouragement as needed.

5.4.3 Procedure

The test platform described in Chapter 4 was used to collect data. For each study, participants reviewed all study related materials, receiving training, and conducted all practice and study trials during a single session which took about an hour and half depending on the number of trials completed, breaks between trials, and feedback after the trials. Each participant first reviewed the study information sheet and a short self-paced slide presentation providing an overview of the test platform. The study administrator aligned the eye tracker for an optimal view of the participant's and eyes, then initiated the automated calibration procedure. During the calibration procedure, the participant watched as a circle moved to 12 locations on the screen to calibrate the eye tracker. Next, an on-screen tutorial delivered interactive training on the robot input methods using the live system with the robots. The participant than completed a number of practice trials to familiarize with the test environment and search task. Finally, the study trials were conducted to collect data.

The test platform Session batch script was used to automate various aspects of user study

sessions to ensure collection of relevant data and consistent application of experimental controls. The script displayed instructions for both the study administrator and participants. For each session, the script executed the following protocol:

1. Display study information sheet and introduction for participant to review
2. Launch the EyeTribe application to access the eye tracker device
3. Launch the EyeTribe UI application to facilitate device alignment and calibration
4. Copy eye tracker device configuration files to the session log directory
5. Launch the Overwatch application with training configuration settings
6. Launch Overwatch with practice trial configuration settings; Repeat for the configured number of practice trials
7. Launch Overwatch with study trial configuration settings; Repeat for the configured number of study trials

5.4.4 Search Task

Participants were asked to locate as many hidden targets as possible during each 5-minute trial. The study trials employed 3 sets of 11 virtual targets. The number of targets was selected based on preliminary testing to determine how many could be located by an expert operator, with the assumption that novice operators would be unlikely to locate as many as the expert. The targets were randomly selected from a larger set of target locations such that the overall task difficulty of each set would be similar. A smaller number of targets that were easier to find were used for the training trials to ensure participants would quickly discover them and gain experience completing the target localization task. In all cases, the number of targets discoverable was not revealed to participants during the studies.

Participants operated the robots in the test environment to search for the targets. When a robot came within the configured distance and angles of approach, a green line was drawn on

the screen from the center of the robot and through the target. This line was a fixed length so as to not reveal the exact location of the target. To localize the target, participants were instructed to use two (or more) robots to detect the target, then click the intersection of the green detection lines to identify its location. The software did not allow the participant to locate the target by guessing. Thus, the task required multiple robots to complete. Credit for locating a target did not occur until the participant clicked the intersection of the lines and a green and gray circle appeared to mark its location.

5.5 Study 2: Visual Alert

5.5.1 Robot Behavior

This study examined the potential use of a confidence-based visual alert or adaptive robot velocity to mitigate inefficient use of automation. Three behaviors in response to robot confidence were tested:

1. A visual alert on the robot while at the low confidence level.
2. Elevated velocity while operating autonomous at the high confidence level, and reduced robot velocity while operating autonomous at the low confidence level.
3. Control – No alerts or change in robot velocity with respect to confidence.

Ideally, the average velocity of the second behavior would be equal to that of the other behaviors in order to control for velocity-related effects. To do so, the robots would need to spend an equal amount of time at either high or low confidence. Preliminary testing was conducted to determine confidence model parameters with a reasonable likelihood of achieving equal time by confidence level.

Each participant completed 9 study trials, 3 for each behavior. To mitigate learning effects, 3 target sets were used. Each target set was used once per behavior. The orders of presentation for both behaviors and target sets were counterbalanced.

5.5.2 Participants

This study was designed to collected data with 12 participants due to the counterbalancing scheme, which accounted for a factor of interest with 3 levels and one categorical control variable (target set) with levels 3. Data were collected with 3 healthy volunteers having normal or corrected-to-normal vision. The low data collection was due to feedback from the participants, followed by an initial examination of the data.

5.6 Study 3: Robot Velocity

5.6.1 Robot Behavior

This study further examined the use of adaptive robot velocity. Again, three behaviors were tested, but for this study the conditions focused on velocity:

1. Elevated robot velocity while operating autonomously at the high confidence level.
2. Reduced robot velocity while operating autonomously at the low confidence level.
3. Control – No change in robot velocity with respect to confidence.

Whereas the Visual Alert study sought the experimental control of equal average velocities, the Robot Velocity study sacrificed this in favor of comparing the “carrot” of elevated velocity at high confidence with the “stick” of reduced velocity at low confidence.

Like the Visual Alert study, each participant completed 9 study trials, 3 per behavior, using a different target set each time a behavior was repeated. The orders of presentation for both behaviors and target sets were again counterbalanced.

5.6.2 Participants

Data were collected with 12 healthy volunteers who had normal or corrected-to-normal vision (3 females, 9 males; mean age = 28.9, SD = 4.4). Each volunteer participated during a single session up to approximately 2 hours in length. Sessions were scheduled to accommodate the participants. All study trials were conducted between 11:13 a.m. and 7:50 p.m. from start to

finish, a range of 8 hours and 37 minutes. During each session, study trials took an average of 55 minutes total, or about 6 minutes per trial. Approximately 1 hour was spent inducting the participant and conducting practice trials before the study trials, and receiving feedback from the participant after all trials were completed.

5.7 Study 4: Time of Day

5.7.1 Robot Behavior

An initial review of the Robot Velocity data showed indications time of day may have been a factor influencing task performance. Study 4 was conducted to collect more data to evaluate potential time of day effects, and to further examine adaptive robot velocity. To control for time of day, all study trials were conducted between 2:20 p.m. and 4:12 p.m. from start to finish, a range of 1 hour and 52 minutes. Like Study 2: Visual Alert, the goal of equal average velocities was pursued as an experimental control. This time, however, only two behaviors were tested:

1. Elevated velocity while operating autonomously at the high confidence level, and reduced robot velocity while operating autonomously at the low confidence level.
2. Control – No change in robot velocity with respect to confidence.

Each participant completed 6 study trials, 3 per behavior, using a different target set each time a behavior was repeated. The orders of presentation for both behaviors and target sets were counterbalanced.

5.7.2 Participants

In order to study potential time of day effects and to reduce the amount of training required, participants from Study 3: Robot Velocity were invited to return for this study. Six returning participants volunteered, again all healthy with normal or corrected-to-normal vision (1 female, 5 males; mean age = 32.2, SD = 5.4).

5.8 Data Analysis

The hypothesis was the implemented adaptive robot behaviors improve search task performance and efficiency. To understand how robot behavior and other factors were related to performance and efficiency, mixed-effects regression models were constructed to explain the observed data by trial using the explanatory variables listed in Table 5-1. Linear mixed-effects models offer a robust statistic method capable of handling a variety of situations such as unbalanced data and missing values, and can be extended via generalized linear mixed-effects models to analyze data with non-normal error distributions [132-136].

Table 5-1: Explanatory variables used to fit mixed-effects models

Variable	Type	Description
behavior	categorical	robot behavior (factor of interest)
target set	categorical	three predefined sets of search targets
confidence	continuous	average robot confidence value by trial
time of day	continuous	start of a given study trial in fractional hours

Initial data quality checks and exploratory analyses were performed using Microsoft Power BI Desktop (version 2.71.5523.941). Regression analyses were conducted using R (version 3.6.1) [137]. Linear mixed-effects models were fit by maximum likelihood using the `lmer` function of the R package `lme4` (version 1.1.21) [138]. The general form of the R formulas used to specify the models was:

```
y ~ behavior + targetset + csconf + cstime + (1 | pid)
```
(5-9)

The explanatory variables were entered into the model as the fixed effects terms `behavior`, `targetset`, `csconf`, and `cstime`. The continuous variables *confidence* and *time of day* were centered and scaled for model fitting, hence the "`cs`" prefix of `csconf` and `cstime` respectively. The random effects term `(1|pid)` specified random intercepts by participant to account for

correlation due to repeated measures.

The PBmodcomp function of the pbkrtest package (version 0.4.7) [139] was used to perform parametric bootstrap model comparisons to test whether each explanatory variable contributed significantly to the model fit. For each comparison, PBmodcomp compared the full model with a reduced model which omitted the variable being tested, and reported the fraction of simulated likelihood ratio test (LRT) values greater than or equal to the observed LRT value. 30,000 simulations were performed per comparison.

5.9 Results

5.9.1 Visual Alert

Data were collected at three levels of robot behavior:

- Visual alert upon low confidence
- Elevated velocity at high confidence and reduced velocity at low confidence
- No alert or change in velocity

The study was halted after collecting data with 3 participants due to unfavorable feedback about the visual alerts from the participants. The primary concern was that the alerts were viewed as either moderately helpful or distracting. Instead of invoking a shift in attention, one participant reported adapting to *tune out* the alerts. Based on the feedback, it was determined the alerts were not having the intended effect and invoked inconsistent participant behavior that was counterproductive relative to the research objective. A decision was made to focus subsequent studies on robot velocity behavior.

The Visual Alert study also precipitated improvements to the test platform and data collection methods that were instrumental to the success of subsequent studies. It was conducted prior to implementation of the pivot input method in the test platform software. Although carefully planned paths or teleop input could be used to face a robot in the desired

direction, participants consistently reported difficulty orienting the robots. A suggestion was made to enable double-clicking a robot to make it automatically rotate about its yaw axis. In addition to implementing this pivot input method, teleop input was refined to always allow pivoting, even when forward and reverse motion were disabled by the obstacle collision avoidance algorithm.

A total of 27 trials were completed. During a few trials the motor power circuit of one or more robot shut down. This was quickly resolved by the test administrator cycling the motor power toggle switch on the robot, but the participant was briefly unable to use the affected robot. The suspected root cause was excessive inrush current triggering the battery pack or H-bridge overcurrent protection. After the study was halted, the test platform software was updated to more gradually increase and decrease motor speed, which resolved the issue.

Inconsistent confidence parameters were discovered in the configuration files after the study. Greater care was taken to ensure consistent parameters for the subsequent studies. Although limited in quantity, the data collected were also useful to help refine configuration options and data collection methods.

The above concerns were independent of the qualitative participant feedback and study observations regarding visual alerts. It is unlikely participants would have reacted differently to the alerts absent these concerns, and feedback from the participants did not link alerts to either robot orientation or the very small number of brief power interruptions they experienced.

5.9.2 Robot Velocity

The Robot Velocity (RV) study collected data at three behavior levels, which were labeled:

- *steady* – no change in velocity with respect to confidence
- *boost* – elevated velocity at high confidence
- *drop* – reduced velocity at low confidence

Scatter plots of targets detected, targets located, and search efficiency against each of the explanatory variables were produced to visualize the data during exploratory analysis. Reference lines were plotted between the means at each level of the categorical variables, and by simple linear regression for the continuous variables. Figure 5-4 contains a scatter plot matrix with all of these plots together for comparison. The plots showed limited signs of behavior and target set effects, but patterns indicating possible confidence and time of day effects.

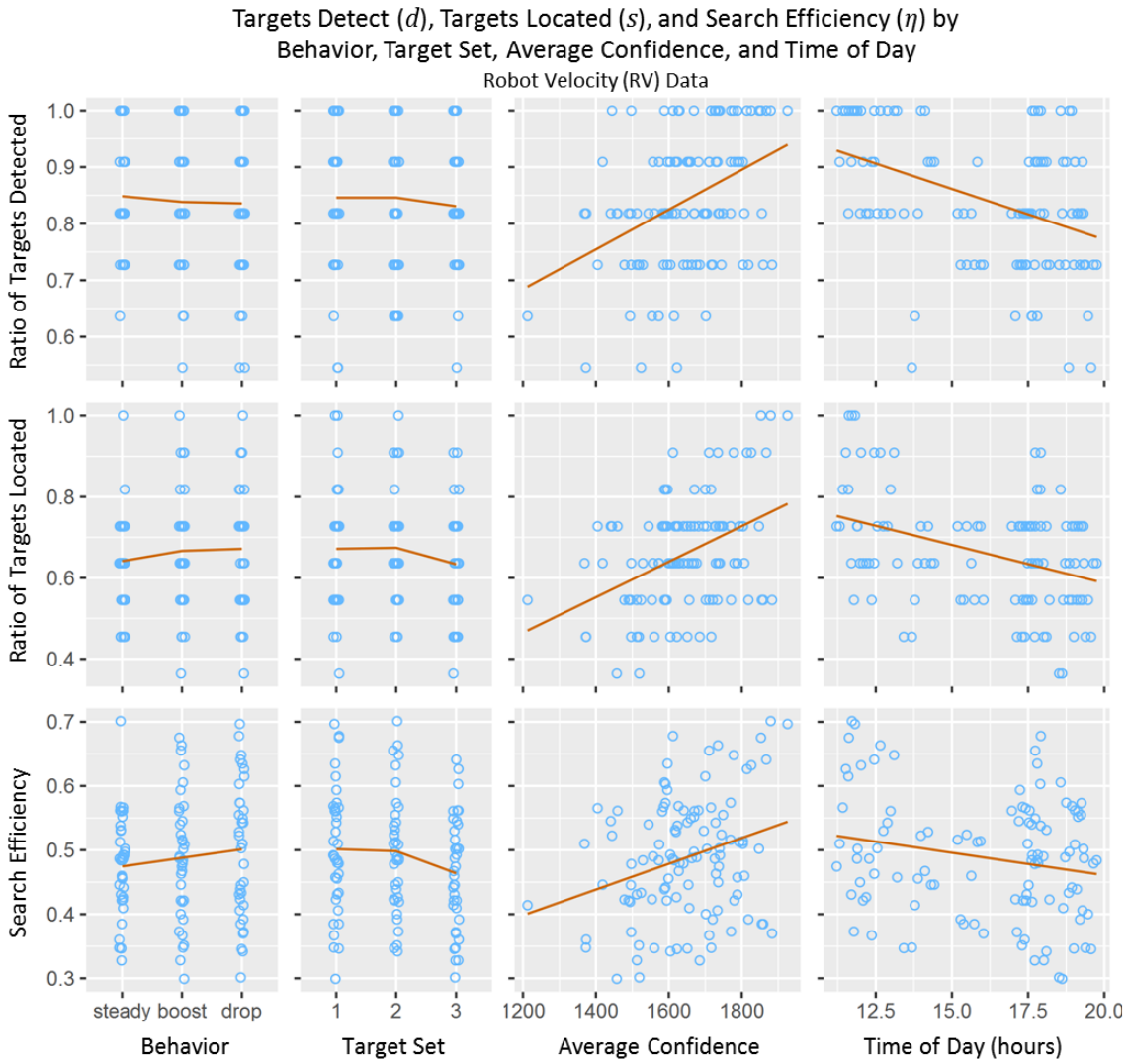

Figure 5-4: RV scatter plots of task performance and efficiency (higher y-axes values are better). Categorical variables: Lines were plotted between mean values, and random variation was used to jitter points horizontally in order to prevent overplotting (behavior and target set). Continuous variables: Simple regression lines were plotted (confidence and time).

Linear mixed-effects models were fitted to explain the observed performance and efficiency using all of the variables, while also accounting for correlation due to repeated measures with each participant. Table 5-2 summarizes the results of parametric bootstrap model comparisons used to test whether each explanatory variable contributed significantly to the model fit, along with reference likelihood ratio test (LRT) χ^2 statistics used by the bootstrap procedure. *BPtest* is the ratio of simulated LRT-values that were greater than or equal to the observed LRT value.

Robot behavior was expected to significantly contribute to fitted models of targets detected, targets located, and efficiency. In contradiction to this hypothesis but consistent with the plots in Figure 5-4, the model comparisons found no main effects of behavior ($p > 0.32$). On the other hand, time of day was not significant ($p > 0.18$) even though the plots showed signs of potential relationships. Target set was also not significant for the mixed-effects models of targets detected ($p = 0.68$), targets located ($p = 0.18$), and search efficiency ($p = 0.074$).

Although behavior was not significant, a significant main effect of confidence was observed for targets detected and located (both $p < 0.01$), as well as search efficiency ($p < 0.05$). In other words, removing confidence from the models significantly decreased the goodness of fit.

Table 5-2: RV parametric bootstrap model comparison results with LRT for reference

Term	Test	df	detected	located	efficiency
behavior	BPtest		0.25 $p = 0.89$	2.26 $p = 0.35$	2.38 $p = 0.32$
	LRT χ^2	2	0.25 $p = 0.88$	2.26 $p = 0.32$	2.38 $p = 0.30$
target set	BPtest		0.83 $p = 0.68$	3.63 $p = 0.18$	5.48 $p = 0.074$
	LRT χ^2	2	0.83 $p = 0.66$	3.63 $p = 0.16$	5.48 $p = 0.065$
confidence	BPtest		10.53 $p = 0.0033$**	12.66 $p = 0.0014$**	7.32 $p = 0.013$*
	LRT χ^2	1	10.53 $p = 0.0012$**	12.66 $p = 0.00037$***	7.32 $p = 0.0068$**
time	BPtest		2.34 $p = 0.18$	1.53 $p = 0.27$	0.060 $p = 0.83$
	LRT χ^2	1	2.34 $p = 0.13$	1.53 $p = 0.22$	0.060 $p = 0.81$

* p < 0.05 ** p < 0.01 *** p < 0.001

The results were interesting for a few reasons. First, the original hypothesis that robot behavior would affect search performance and efficiency was contradicted. Second, although

time of day was not found significant, the relatively low *p*-value was cause for concern. Finally, robot confidence was not expected to exhibit a strong relationship with performance or efficiency, but was the only observed significant effect.

5.9.3 Time of Day

To further examine robot velocity behavior and confidence, the Time of Day (TD) study collected data at two levels of behavior:

- *steady* – no change in velocity with respect to confidence
- *adaptive* – elevated velocity at high confidence, reduced velocity at low confidence

The TD study was conducted to collect additional data to compare with and extend the RV study data. Due to the limited number of observations, the TD data on its own did not support mixed-effects regression modeling. Instead, after visualizing the data from both studies to confirm sufficient similarity, the combined RVTD data were used to fit models and compare the results with those from the RV data alone. Analyses of the combined data took advantage of the robustness offered by linear mixed-effects models [132-136].

Figure 5-5 contains scatter plots of the RVTD data. The RV and TD studies included the same *steady* behavior, which served as a control with which to compare the other behaviors. Initial inspection of the data collected at the steady condition did not support combining observations from RV and TD, so these were differentiated as *steady1* and *steady2* respectively. Overall, the RVTD scatter plots (Figure 5-5) were very similar to RV alone (Figure 5-4). Aside from more levels of behavior, one notable difference is average performance and efficiency by target set appeared to differ by a larger amount for RVTD, although the third target set was consistently associated with the lowest average values.

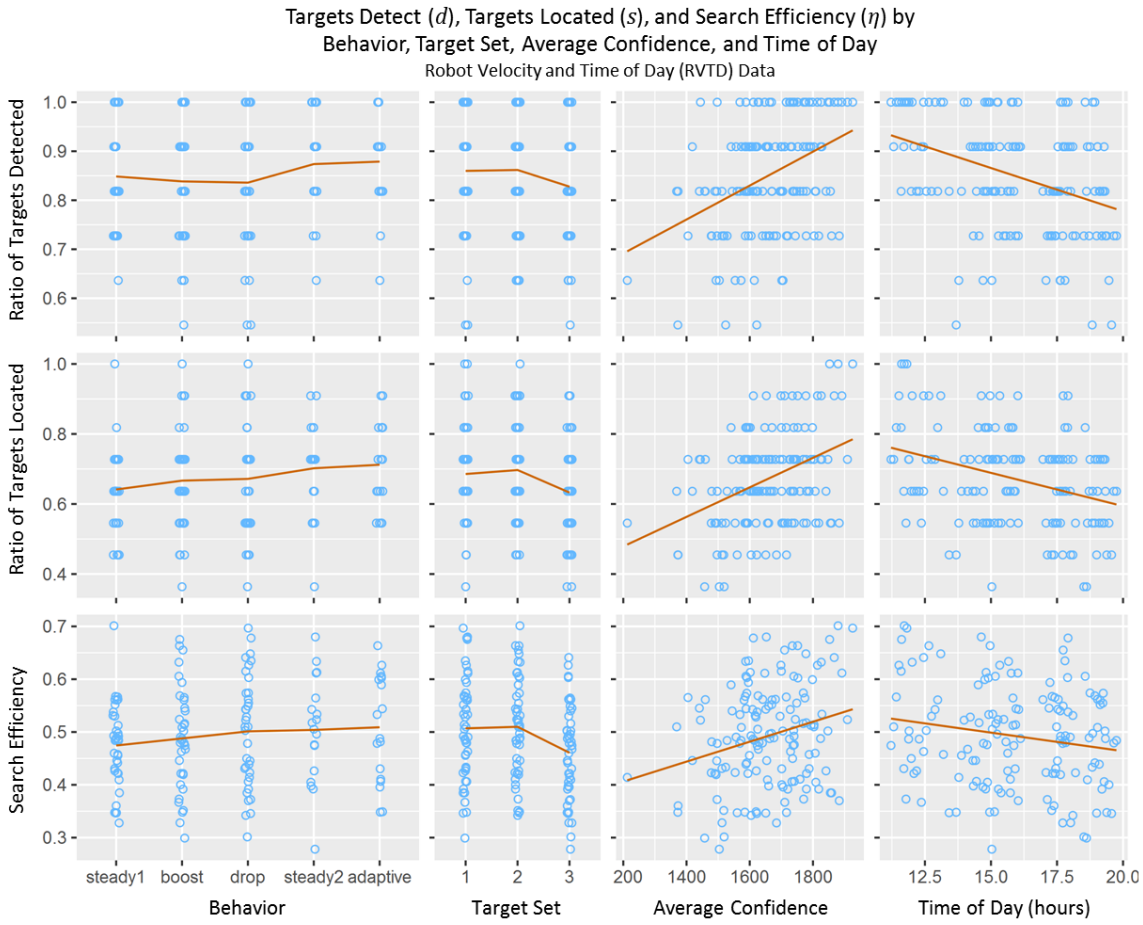

Figure 5-5: RVTD scatter plots of task performance and efficiency (higher values are better). Categorical variables: Lines were plotted between mean values, and random variation was used to jitter points horizontally in order to prevent overplotting (behavior and target set). Continuous variables: Simple regression lines were plotted (confidence and time).

Linear mixed-effects models were again fitted to explain targets detected, targets located, and search efficiency. The results of parametric bootstrap model comparisons are summarized in Table 5-3 with LRT statistics for reference. In general, the RVTD results reinforce the RV findings. As before, behavior was not significant for all three models ($p < 0.44$), nor was time of day ($p < 0.27$). Consistent with RV, the RVTD models found confidence was significant for targets detected and located ($p < 0.001$) and efficiency ($p < 0.05$). Target set was again not significant for targets detected ($p = 0.18$). Unlike RV, however, an effect of target set was observed for targets located ($p < 0.05$) and efficiency ($p < 0.01$).

Table 5-3: RVTD parametric bootstrap model comparisons with LRT for reference

Term	Test	df	detected	located	efficiency
behavior	BPtest		1.10 $p = 0.91$	3.98 $p = 0.44$	2.83 $p = 0.62$
	LRT χ^2	4	1.10 $p = 0.89$	3.98 $p = 0.41$	2.83 $p = 0.59$
target set	BPtest		3.64 $p = 0.18$	8.55 $p = 0.017$*	10.23 $p = 0.0092$**
	LRT χ^2	2	3.64 $p = 0.16$	8.55 $p = 0.014$*	10.23 $p = 0.0060$**
confidence	BPtest		16.07 $p = 0.00023$***	15.88 $p = 0.0003$***	7.95 $p = 0.010$*
	LRT χ^2	1	16.07 $p = 6.1e-05$***	15.88 $p = 6.74e-05$***	7.95 $p = 0.0048$**
time	BPtest		1.43 $p = 0.27$	0.96 $p = 0.37$	0.12 $p = 0.75$
	LRT χ^2	1	1.43 $p = 0.23$	0.96 $p = 0.33$	0.12 $p = 0.73$

* $p < 0.05$ ** $p < 0.01$ *** $p < 0.001$

5.9.4 Summary

All three studies presented in this chapter applied the same fundamental design, counterbalancing scheme, search task, and general procedure. The key differences were robot behaviors and the added Time of Day control. The qualitative results of the Visual Alert study are reviewed in the Discussion.

Table 5-4 summarizes the quantitative results from the RV and RVTD data analyses. These studies used the same target sets, confidence model parameters, and other configuration settings. This makes comparisons between target set and confidence results relatively straightforward.

Table 5-4: Summary of performance and efficiency mixed-effects model comparisons

Term	Data	detected	located	efficiency
behavior	RV	0.25 $p = 0.89$	2.26 $p = 0.35$	2.38 $p = 0.32$
	RVTD	1.10 $p = 0.91$	3.98 $p = 0.44$	2.83 $p = 0.62$
target set	RV	0.83 $p = 0.68$	3.63 $p = 0.18$	5.48 $p = 0.074$
	RVTD	3.64 $p = 0.18$	8.55 $p = 0.017$*	10.23 $p = 0.0092$**
confidence	RV	10.53 $p = 0.0033$**	12.66 $p = 0.0014$**	7.32 $p = 0.013$*
	RVTD	16.07 $p = 0.00023$***	15.88 $p = 0.0003$***	7.95 $p = 0.010$*
time	RV	2.34 $p = 0.18$	1.53 $p = 0.27$	0.06 $p = 0.83$
	RVTD	1.43 $p = 0.27$	0.96 $p = 0.37$	0.12 $p = 0.75$

* $p < 0.05$ ** $p < 0.01$ *** $p < 0.001$

Although the target maps were carefully designed equitable conditions, differences by target set were not unexpected. The significance of confidence, however, was not anticipated. Results with both the RV and extended RVTD data reinforce the significance of confidence to models of observed performance and efficiency. See Section 5.10.3 for further discussion.

Conditional scatter plots were created of targets detected, targets located, and search efficiency versus confidence (see Figure 5-6, Figure 5-7, and Figure 5-8). These showed limited evidence of interaction between confidence and the other explanatory variables given the inherent variability of data collected with human participants performing a challenging search task using physical robots in a real test environment. Simple linear regression was used to fit lines at each level of the categorical variables study, behavior, and target set. For the continuous variable, time of day was divided into quintile intervals.

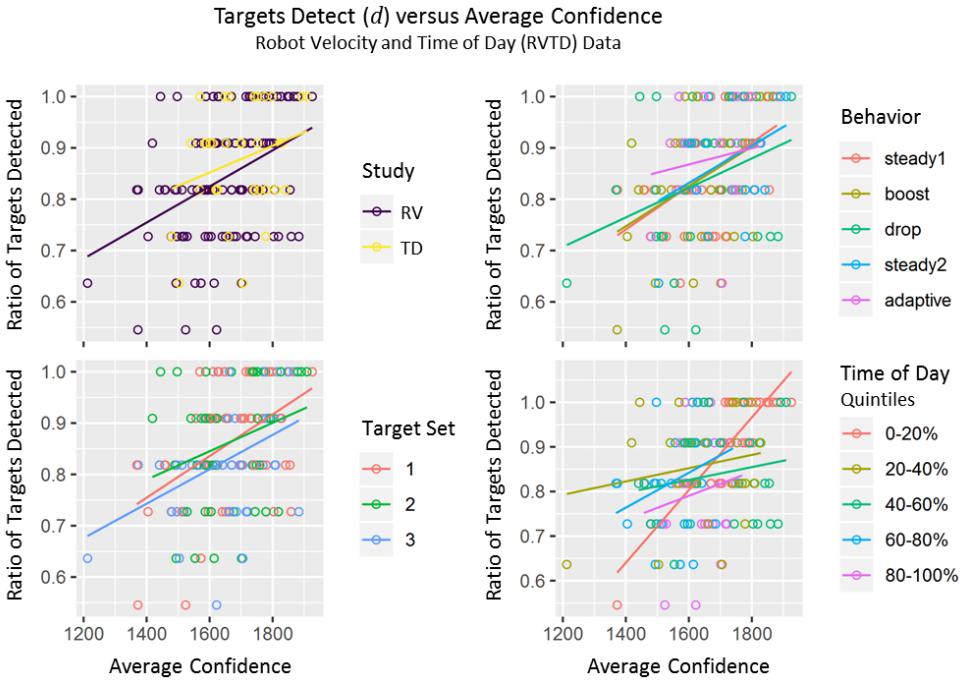

Figure 5-6: Targets detected versus confidence was relatively consistent by study, behavior, target set, and time of day (RVTD plots shown; RV plots were similar).

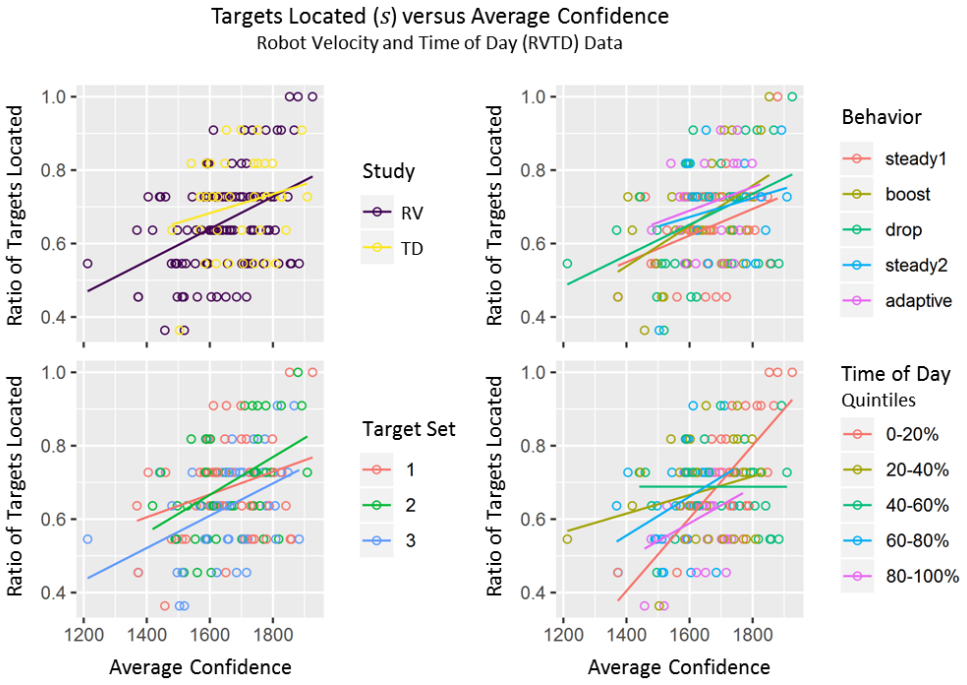

Figure 5-7: Targets located versus confidence was relatively consistent by study, behavior, target set, and time of day (RVTD plots shown; RV plots were similar).

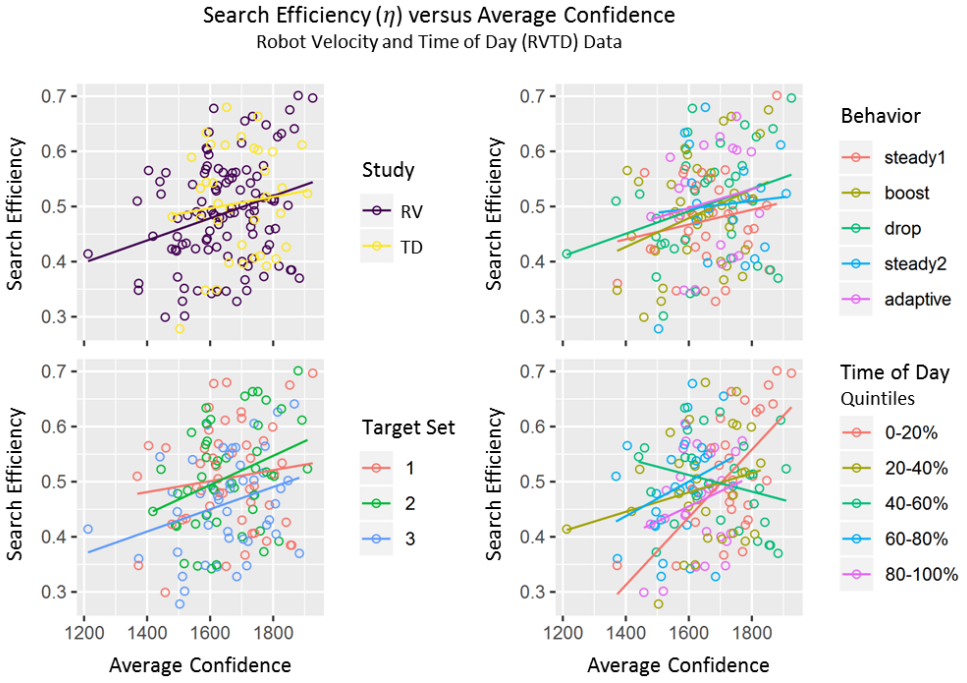

Figure 5-8: Search efficiency versus confidence was relatively consistent by study, behavior, target set, and time of day (RVTD plots shown; RV plots were similar).

The RV and TD studies shared the same baseline *steady* behavior, which did not change robot velocity conditionally in response to confidence, but evaluated different adaptive velocity behaviors. The RV *boost* and *drop* behaviors implemented relatively moderate velocity increases and decreases respectively, while the TD *adaptive* behavior used the highest and lowest velocities practical to maximize effects. The non-significant results for all models using either RV or RVTD data provided strong evidence these behaviors did not contribute to observed performance and efficiency.

Exploratory analysis of the RV data led to concerns about time-of-day effects such as fatigue. The TD study controlled for time of day by limiting trials to within the same 2-hour window. The TD data fell between the minimum and maximum times of the RV trials, and added data points in what had been a more sparsely populated interval of time in the RV data. As with behavior, the result of no significant time-of-day effects for the combined data reinforced that time did not contribute to observed performance and efficiency despite what initially appeared to be relationships shown by the scatter plots.

5.10 Discussion

5.10.1 General Feedback

Participants were highly enthusiastic, with many reporting they enjoyed participating in the study and felt the session went by fast. Positive feedback was received about the search task, which participants found challenging but rewarding. The physical robots enhanced engagement by arousing participant's curiosity and sparking discussion about the test platform and the presented research. Interest continued even after the study sessions, and six out of twelve participants from the RV study eagerly volunteered to participate again for the TD study. These results provided validation for the platform approach using physical robots and evidence of interest in this research within the broader community.

Although the literature contains examples of difficulty getting eye tracking equipment to work with certain individual human subjects, no participants were excluded due to unsuccessful calibration or as a result of excessive tracking errors during the studies conducted for this work.

5.10.2 Visual Alert Discussion

The Visual Alert study was stopped early but still yielded results germane to the research objective. Participants provided valuable qualitative feedback concerning the use of visual alerts triggered by changes in robot confidence. This information helped set the conditions for the RV and TD studies. Participants generally held an unfavorable view of the implemented alert behavior, and reported their perception that it was modestly helpful at best or distracting. Although the alerts were intended to shift the operator's attention to where it is needed, one participant adapted during the study to *tune out* the alerts in direct opposition to this goal. Based on the feedback and study observations, it was determined the alerts were not having the intended effect and invoked inconsistent participant behavior that was counterproductive relative to the research objective. A decision was made to focus on robot velocity in the subsequent studies.

5.10.3 Robot Confidence and Behavior Discussion

The confidence model described in section 5.3 was implemented as a means of adapting robot behavior in response to operator attention. The user studies described in this chapter were conducted to compare several adaptive behaviors using a within-subjects design. Data for targets detected, targets located, and search efficiency were collected via repeated measures of each participant performing a common search task with all of the robot behaviors evaluated by a specific study.

The expected outcome of the adaptive behaviors was an improvement in search task performance and efficiency, but behavior was found to be not significant. Instead, parametric

bootstrap comparisons of mixed-effects models found the by-trial average *confidence value* itself contributed significantly to models, evidence of positive relationships with both search task performance measures and search efficiency.

5.11 Conclusion

The robot confidence model described in this chapter was developed in order to improve telerobotic performance and efficiency for multirobot systems by adapting robot behavior based on operator attention. The user studies presented in this chapter were conducted to evaluate various robot behaviors to this end. These studies contributed to an understanding of which behaviors are likely to support this goal. Specifically, subjective feedback from Visual Alert participants suggested visual alerts may invoke a variety of potentially counterproductive operator responses, including maladaptive behaviors such as learning to *tune out* the alerts in order to ignore them. Subsequent studies focused on adapting robot velocity according to confidence. Instead of finding the expected outcome that robot behavior improved performance and efficiency, the results provided evidence that the confidence model itself has utility as a *predictor* of telerobotic performance and efficiency.

CHAPTER 6: DISCUSSION AND FUTURE WORK

6.1 Summary

In Chapter 3, Study 1: Levels of Autonomy examined multirobot teleoperation at three levels of robot autonomy and identified the problem of overreliance on autonomy. This led to the hypothesis that operator attention can be assessed in real time and used to mitigate inefficient use of automation by the operator and result in improved telerobotic task performance and efficiency.

The multirobot test platform described in Chapter 4 was designed and implemented to evaluate techniques of assessing operator attention in real time and adapting robot behavior accordingly. The resulting system integrated eye tracking to measure physiological properties associated with selective attention.

In Chapter 5, a generalized robot confidence model was introduced which transforms multiple indicators of operator attention to a single confidence value which can be used to adapt robot behaviors. The model produces a confidence value for each robot using a weighted-maximum to aggregate any number of inputs which may exhibit a high degree of variability, such as eye gaze fixations near a point of interest, along with a decremented previous value as feedback and a minimum confidence limit. The model was implemented using eye gaze fixation and user input as indicators of attention, along with adaptive behaviors which were automatically selected by configurable confidence threshold values. The resulting implementation assessed operator attention in real time to determine the confidence value of each robot and altered robot behavior accordingly.

Also detailed in Chapter 5, a series of three controlled user studies were conducted to examine potential effects of the implemented adaptive behaviors with respect to search task performance and efficiency. The Visual Alert study found the implemented flashing alerts

triggered by low confidence to be counterproductive. The Robot Velocity and Time of Day studies evaluated behaviors that varied robot velocity based on confidence. Regression analyses were performed by fitting linear mixed-effects models of observed performance and efficiency, then using parametric bootstrap comparison to assess the significance of each explanatory variable to the model fit. Although the adaptive behaviors were not significant, average by-trial robot confidence was found to have contributed significantly to the models.

6.2 Specific Contributions

This research developed novel techniques applicable to user interface designs for the remote operation of multiple unmanned vehicles. First, a generalized robot confidence model was introduced which transforms an arbitrary number of operator attention indicators to a confidence value for each robot in order to enable adaptive robot behaviors. Second, the model was implemented and successfully evaluated to reveal evidence linking average robot confidence to multirobot search task performance and efficiency. These contributions provide important steps toward effective human teleoperation of multiple mobile robots.

Other research related to robot confidence has been aimed at influencing human trust in autonomy and, ultimately, optimizing the allocation of control between autonomy and manual operation. Instead of directly affecting trust or improving allocation of control, the presented work addressed challenges related to human cognition that limit teleoperation of multiple mobile robots. Specifically, this work employed robot confidence as a metaphor relating indicators of operator attention and robot behaviors which response to these indicators, and observed correlations between average confidence and three measures of multirobot search performance and efficiency.

6.3 Future Work

6.3.1 Expanding the Scope

The scope of this research focused on a single human using a fixed control station to teleoperate a small number of mobile robots in order to complete specific search tasks designed to evaluate user interface techniques in a controlled environment. However, the techniques developed in this research could apply more broadly.

Real-world applications may involve a team of multiple robots and multiple humans. The presented techniques were implemented for one operator but could be extended to multiple operators. For example, attention could be estimated based on eye gaze fixations and input from multiple users, or the confidence model might be expanded to include attention from multiple sources in addition to aggregate attention from all users. The model might also be a useful mechanism for deconflicting and prioritizing input from multiple sources.

The presented work integrated a remote eye tracker to measure physiological properties associated with selective attention, and used these data as input to derive robot confidence. Real-world applications with multiple mobile robots in the field may involve first responders, dismounted soldiers, or other telerobot operators who require freedom of movement in environments not conducive to display-mounted eye tracking equipment. There is significant interest in head-mounted eye tracking, especially in the defense sector where many potential end users (e.g., dismounted soldiers) already wear helmets and helmet-mounted equipment. This work could be extended to head- or helmet-mounted eye tracking to address specific related integration challenges and identify new research opportunities.

The presented system separated the operator from the robot environment for a variety of reasons, both practical and experimental. Placing the operator and robots in a shared environment would afford opportunities to extend this work. For example, direct observation of

the operator by the robots would add new information for estimating operator attention such as body language.

This work produced four small mobile robots which were used to develop and evaluate the presented techniques. These were deliberately equipped with limited autonomy in order to place higher demand on the operator. Future work could enable higher levels of autonomy and scale the number of robots up. A swarm of robots, for example, may implement individual or collective behaviors influenced by individual or collective confidence, or perhaps a confidence gradient over the volume in which the robots are operating.

On the other end of the spectrum, future work could continue to focus on a small number of robots, but having more advanced capabilities or platform specialization. For example, any number of UGVs and UAVs with different capabilities could work together as a heterogeneous multirobot team. The presented techniques might be extended to account for the unique role or capabilities of a particular robot.

This research used specific search tasks designed to evaluate the presented techniques in a controlled environment. Instead of searching for hidden targets, future work could examine other tasks involving multiple robots such as coordinated material handling and construction.

6.3.2 Extending the Techniques

There are a number of potential opportunities to improve and build upon the techniques developed in this research. The presented robot confidence model focused on eye gaze fixation to estimate operator attention, but can be extended to assess other physiological properties and aspects of human cognition. In addition to eye gaze coordinates, the eye tracker integrated for this work measured and outputted streaming pupillometry data. These data might be used to assess real-time workload using task-evoked pupillary response.

The confidence model could also be extended to incorporate other potentially relevant

information such as recent collisions, target detections, and distance traveled. The model could differentiate between types of user input and consider the frequency of input. Study participants were observed on multiple occasions inputting a pivot maneuver to orient a robot, then failing to pay attention as the robot rotated past the desired angle. Confidence might be decreased if a pattern of inputs suggests the operator is not sufficient engaged. A model of robot confidence might also estimate the confidence of the operator and adapt robot behavior accordingly.

Specific visual alert and robot velocity behaviors were developed and evaluated for this work. Other robot behaviors could be implemented such as stopping upon target detection during low confidence, or increasing autonomy during low confidence to shift allocation of control toward the robots while the operator is busy attending other tasks. This work used threshold confidence values to trigger behaviors. Future work could implement continuous responses to confidence. Robot behavior could also be expanded to include collective team behaviors such as directional icons on or near each robot to direct operator attention toward a specific robot or area in the environment.

This work implemented confidence and behavior independently for each robot. Confidence could be extended to include information about other robots such as relative location, velocity, target detections, collisions, fixations by the operator, and user input. This information can be managed centrally, as with the presented test platform, or exchanged among robots using high speed communication. In addition to accounting for the input, conditions, and actions of other robots, confidence could be jointly negotiated among robots to avoid under- or overconfidence due to a robot having incomplete knowledge.

Finally, this work introduced and implemented a relatively simple model of robot confidence to avoid unnecessary complexity. Additional parameters could be incorporated and discrete

inputs, states, and outputs could be made continuous, such as enabling gradual increases and decrease in confidence over time rather than single discrete changes in response to inputs. The presented techniques were applied consistently for every robot and operator, but might be further developed to achieve the desired aim. For example, machine learning could be employed to train a specific robot based on operational conditions, or to tailor the model for individual operators. The presented techniques have the potential to enable more effective and efficient teleoperation of multiple mobile robots to perform spatially distributed and hazardous tasks in complex environments, and equip human-multirobot interfaces that adapt to the unique needs of individual operators.

APPENDIX A: TRACKED ROBOT DESIGN DETAILS

Table A-1: Tracked robot hardware bill of materials

Component	Description	Qty
battery pack, digital power	Anker Astro E1 5200 mAh, A1211012	1
battery pack, motor power	Anker Astro E1 6700 mAh, A1211015	1
cable, power	USB A plug, USB Micro-B plug	2
chassis	Dagu Rover 5	1
connector, driver power, contact	18-22AWG, TE 1123721-2	2
connector, driver power, header	2 circuit, TE 1744048-2	1
connector, driver power, housing	2 circuit, TE 1744036-2	1
connector, motor, header	4 circuit, TE 1744048-4	1
connector, motor, housing	housing, 4 circuit, TE 1744036-4	1
controller board	Raspberry Pi 2 Model B	1
controller memory	16GB microSDHC	1
controller module, motor driver	DRV8835 Dual Motor Driver	1
controller module, wireless	Edimax EW-7811Un USB Wi-Fi	1
fiducial marker clamp	steel binder clip, 19 mm (3/4 in), silver finish	4
fiducial marker tag	AprilTag, black on white paper	1
hex hut, controller mount	N2.5-0.45, 2.1 mm thick, nylon	4
hex nut, controller mount	M2.5-0.45, steel	4
LED lamp board	100 mA, motor power circuit	1
LED lamp receptacle	USB A, motor power circuit	1
motor power receptacle	USB Micro-B breakout board	1
motor power switch, rocker, SPST	AC 250V 3A 2 pin on/off I/O SPST snap-in	1
motor power wire	2-conductor, 20 AWG, black-red	AR
mounting plate, controller	polycarbonate sheet, 0.093 × 8 × 5 in	1
mounting plate, fiducial	polycarbonate sheet, 0.093 × 10 × 8 in	1
screw, controller mount	M2.5-0.45 × 5 mm, pan head, nylon 6/6	4
screw, mounting plate	#6-32 × 3/8-in, flat head, zinc plated	4
standoff, controller mount	M2.5-0.45 × 6 mm Female × 6 mm Male, Nylon	4
standoff, mounting plate	#6-32 × 1.5-in, male/female, aluminum	4

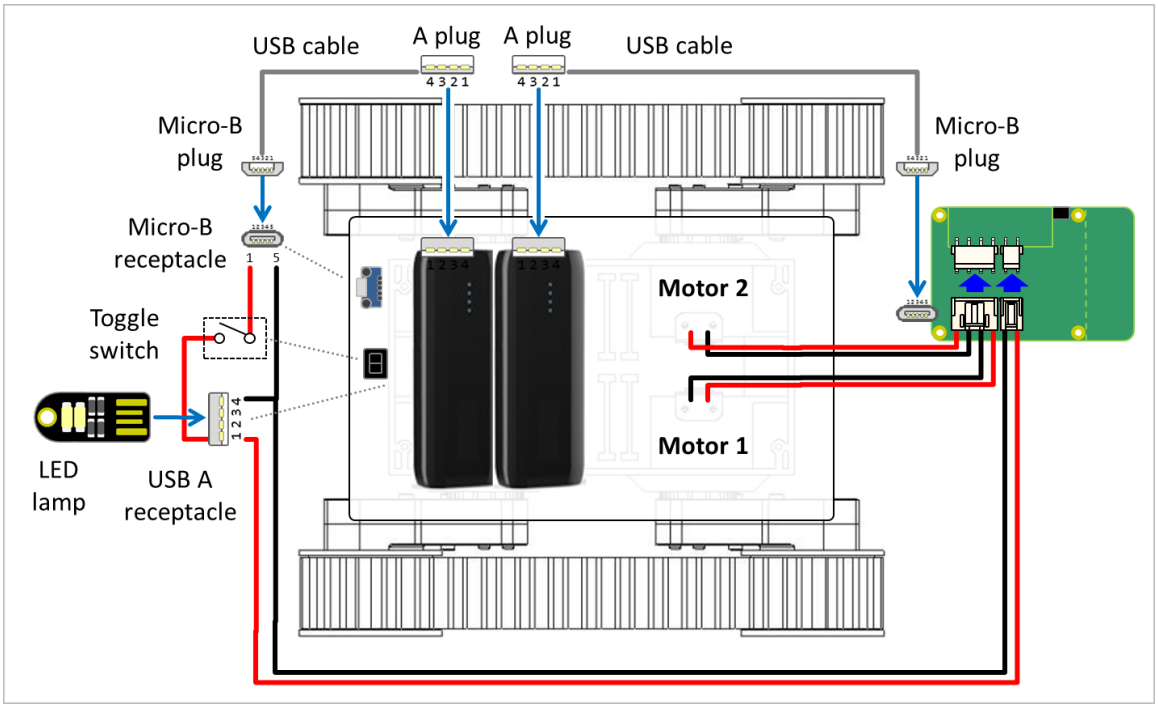

Figure A-1: Tracked robot electrical power circuit.

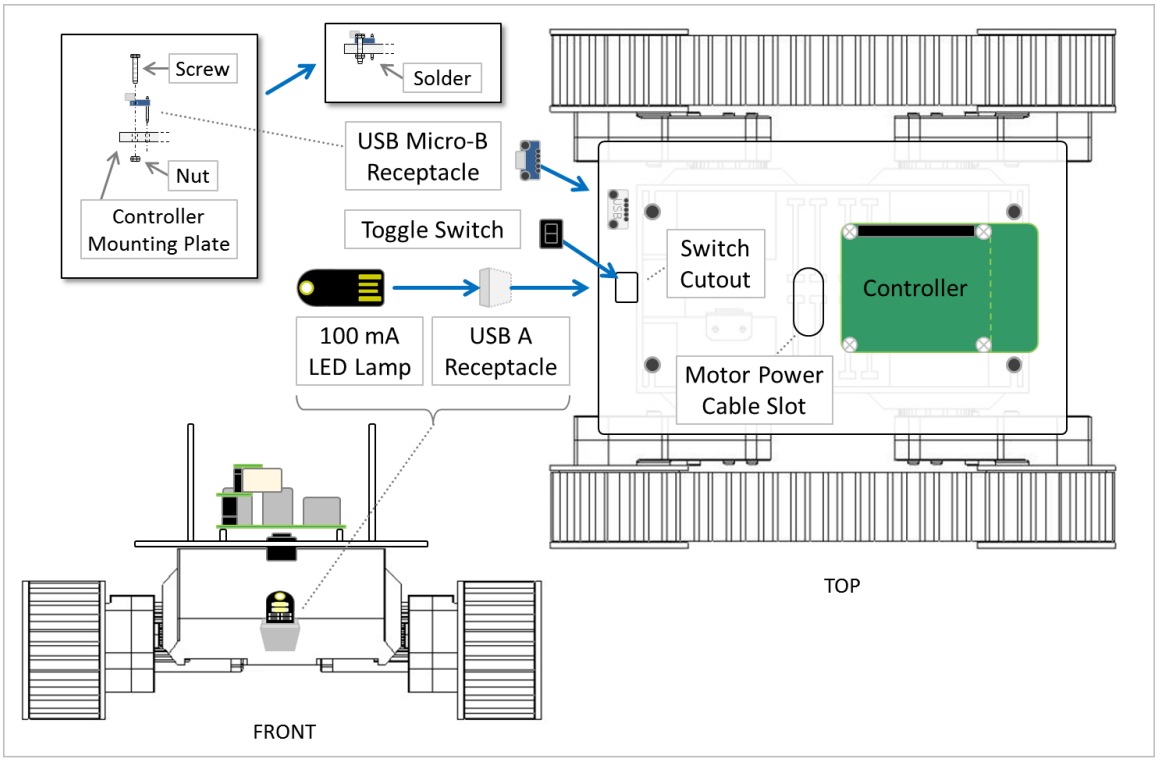

Figure A-2: Tracked robot power component integration.

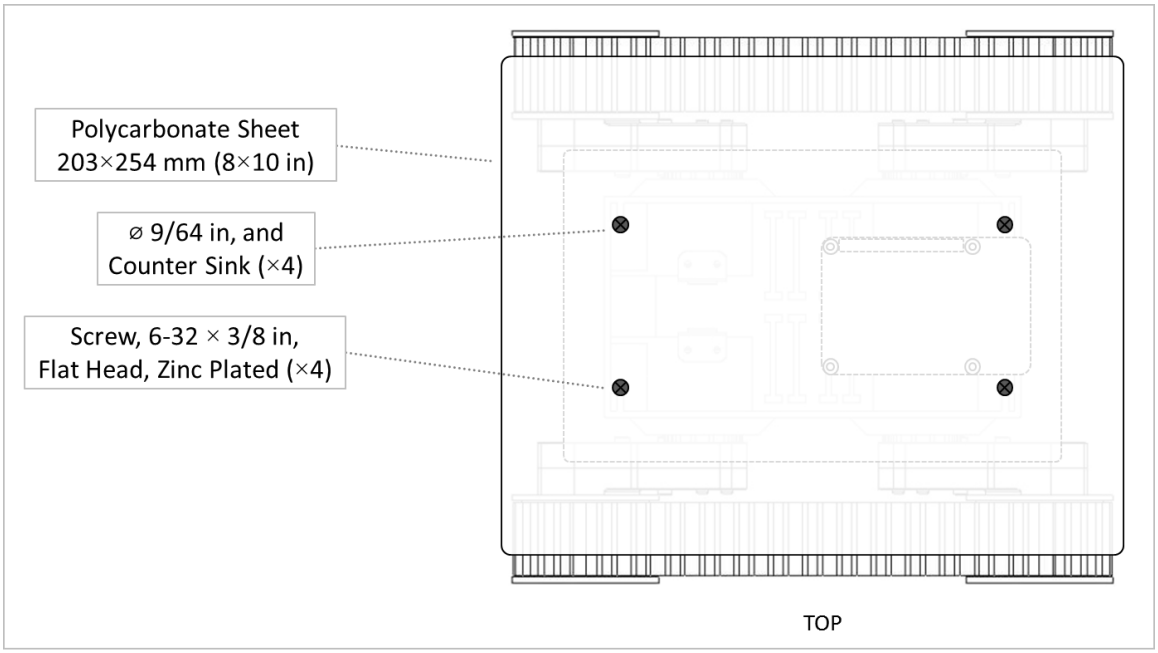

Figure A-3: Tracked robot fiducial mounting plate.

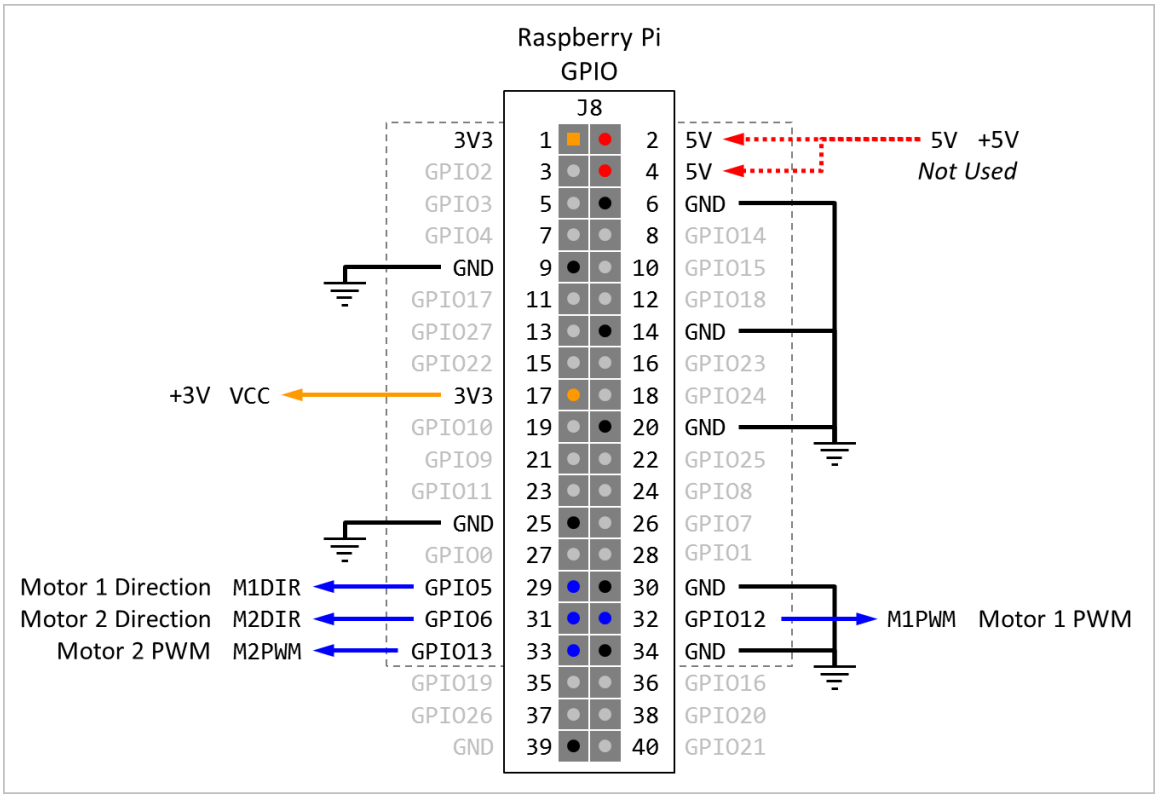

Figure A-4: Tracked robot motor driver interface.

APPENDIX B: HUMAN INVESTIGATION COMMITTEE APPROVAL

WAYNE STATE UNIVERSITY

IRB Administration Office
87 East Canfield, Second Floor
Detroit, Michigan 48201
Phone: (313) 577-1628
FAX: (313) 993-7122
http://irb.wayne.edu

NOTICE OF EXPEDITED CONTINUATION APPROVAL

To: Abhilash Pandya
Electrical & Computer Engineering

From: Dr. Scott Millis or designee
Chairperson, Behavioral Institutional Review Board (B3)

Date: August 12, 2019

RE:
- IRB #: 082105B3E(R)
- Protocol Title: Human Factors Analysis for Robotic Teleoperation
- Funding Source: Award: 017781-001
- Protocol #: 0705004861

Expiration Date: August 11, 2020

Risk Level / Category: Research not involving greater than minimal risk

---

Continuation for the above-referenced protocol and items listed below (if applicable) were APPROVED following Expedited Review by the Chairperson/designee of the Wayne State University Institutional Review Board (B3) for the period of **08/12/2019 through 08/11/2020**. This approval does not replace any departmental or other approvals that may be required.

- Actively accruing participants.
- Waiver of written documentation of consent continued and approved.
- Research Information Sheet (dated 05/14/2019)
- This protocol incurred a period of non-approval from August 1, 2019 to August 11, 2019. The Principal Investigator has confirmed that no research activities have occurred during the non-approval period. Justification for continuing review of minimal risk research: Application for federal funding will be submitted in the future.

- Federal regulations require that all research be reviewed at least annually. You *may* receive a *"Continuation Renewal Reminder"* approximately two months prior to the expiration date; however, it is the Principal Investigator's responsibility to obtain review and continued approval ***before*** the expiration date. Data collected during a period of lapsed approval is unapproved research and can never be reported or published as research data.
- All changes or amendments to the above-referenced protocol require review and approval by the IRB **BEFORE** implementation.
- Adverse Reactions/Unexpected Events (AR/UE) must be submitted on the appropriate form within the timeframe specified in the IRB Administration Office Policy (http://www.irb.wayne.edu//policies-human-research.php).

NOTE:

1. Upon notification of an impending regulatory site visit, hold notification, and/or external audit the IRB Administration Office must be contacted immediately.
2. Forms should be downloaded from the IRB website at each use.

*Based on the Expedited Review List, revised November 1998

Notify the IRB of any changes to the funding status of the above-referenced protocol.

REFERENCES

[1] G. Dudek, M. R. M. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for multi-agent robotics," *Autonomous Robots,* vol. 3, pp. 375-397, 1996.

[2] A. Farinelli, L. Iocchi, and D. Nardi, "Multirobot systems: a classification focused on coordination," *IEEE Transactions on Systems Man and Cybernetics, Part B: Cybernetics,* vol. 34, pp. 2015-2028, 2004.

[3] T. Arai, E. Pagello, and L. E. Parker, "Guest editorial advances in multirobot systems," *Robotics and Automation, IEEE Transactions on,* vol. 18, pp. 655-661, 2002.

[4] L. E. Parker, "Current research in multirobot systems," *Artificial Life and Robotics,* vol. 7, pp. 1-5, 2003.

[5] N. P. Lucas, A. K. Pandya, and R. D. Ellis, "Review of multi-robot taxonomy, trends, and applications for defense and space," in *SPIE Defense, Security, and Sensing*, 2012, pp. 83871N-83871N-10.

[6] M. J. Mataric "Reinforcement learning in the multi-robot domain," *Autonomous Robots,* vol. 4, pp. 73-83, 1997.

[7] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots,* vol. 8, pp. 345-383, 2000.

[8] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE,* vol. 94, pp. 1257-1270, 2006.

[9] L. E. Parker, "ALLIANCE: An architecture for fault tolerant multirobot cooperation," *Robotics and Automation, IEEE Transactions on,* vol. 14, pp. 220-240, 1998.

[10] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots,* vol. 8, pp. 325-344, 2000.

[11] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot

exploration," *IEEE Transactions on Robotics,* vol. 21, pp. 376-386, 2005.

[12] L. E. Parker, "Heterogeneous multi-robot cooperation," Citeseer, 1994.

[13] J. Thangavelautham, K. Law, T. Fu, N. A. El Samid, A. D. Smith, and G. M. D'Eleuterio, "Autonomous multirobot excavation for lunar applications," *Robotica,* vol. 35, pp. 2330-2362, 2017.

[14] A. Stroupe, T. Huntsberger, A. Okon, H. Aghazarian, and M. Robinson, "Behavior-based multi-robot collaboration for autonomous construction tasks," *IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 1495-1500, 2005.

[15] R. Simmons, S. Singh, F. Heger, L. M. Hiatt, S. Koterba, N. Melchior*, et al.*, "Human-robot teams for large-scale assembly," in *Proc. of the NASA Science Technology Conference 2007*, 2007.

[16] S. Dubowsky and P. Boning, "The coordinated control of space robot teams for the on-orbit construction of large flexible space structures," in *Proceedings of the 2007 IEEE International Conference Robotics and Automation, Special Workshop on Space Robotics*, 2007.

[17] H. Ueno, T. Nishimaki, M. Oda, and N. Inaba, "Autonomous cooperative robots for space structure assembly and maintenance," in *Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, NARA, Japan, 2003.

[18] T. Huntsberger, P. Pirjanian, A. Trebi-Ollennu, H. Das, H. Aghazarian, A. Ganino*, et al.*, "Tightly-coupled coordination of multi-robot systems for mars exploration," *IEEE Transactions on Robotics and Automation,* 2001.

[19] J. Leitner, "Multi-robot cooperation in space: A survey," *2009 Advanced Technologies for Enhanced Quality of Life,* pp. 144-151, 2009.

[20] A. Rosenfeld, N. Agmon, O. Maksimov, and S. Kraus, "Intelligent agent supporting human–multi-robot team collaboration," *Artificial Intelligence,* vol. 252, pp. 211-231, 2017.

[21] S.-Y. Chien, M. Lewis, S. Mehrotra, S. Han, N. Brooks, H. Wang*, et al.*, "Task Switching for Supervisory Control of Multi-Robot Teams," *IEEE Transactions on Human-Machine Systems,* 2016.

[22] G.-J. M. Kruijff, M. Janíček, S. Keshavdas, B. Larochelle, H. Zender, N. J. Smets*, et al.*, "Experience in system design for human-robot teaming in urban search and rescue," in *Field and Service Robotics*, 2014, pp. 111-125.

[23] V. Zadorozhny and M. Lewis, "Information fusion based on collective intelligence for multi-robot search and rescue missions," in *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, 2013, pp. 275-278.

[24] M. Lewis, H. Wang, S. Y. Chien, P. Velagapudi, P. Scerri, and K. Sycara, "Choosing autonomy modes for multirobot search," *Human Factors,* vol. 52, pp. 225-233, 2010.

[25] I. Kruijff-Korbayová, F. Colas, M. Gianni, F. Pirri, J. Greeff, K. Hindriks*, et al.*, "TRADR project: Long-term human-robot teaming for robot assisted disaster response," *KI-Künstliche Intelligenz,* vol. 29, pp. 193-201, 2015.

[26] J. Gregory, J. Fink, E. Stump, J. Twigg, J. Rogers, D. Baran*, et al.*, "Application of multi-robot systems to disaster-relief scenarios with limited communication," in *Field and Service Robotics*, 2016, pp. 639-653.

[27] S. Dawson, C. Crawford, E. Dillon, and M. Anderson, "Affecting operator trust in intelligent multirobot surveillance systems," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 3298-3304.

[28] M. Dogar, R. A. Knepper, A. Spielberg, C. Choi, H. I. Christensen, and D. Rus, "Multi-scale

assembly with robot teams," *The International Journal of Robotics Research,* vol. 34, pp. 1645-1659, 2015.

[29] G. Warwick. (13 October 2014). *Unmanned K-Max Operates With Unmanned Ground Vehicle*. Available: http://aviationweek.com/military-government/unmanned-k-max-operates-unmanned-ground-vehicle

[30] L. Y. Wang, A. Syed, G. Yin, A. Pandya, and H. Zhang, "Coordinated vehicle platoon control: Weighted and constrained consensus and communication network topologies," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, 2012, pp. 4057-4062.

[31] S. Lee, N. P. Lucas, R. D. Ellis, and A. Pandya, "Development and human factors analysis of an augmented reality interface for multi-robot tele-operation and control," in *SPIE Defense, Security, and Sensing*, 2012, pp. 83870N-83870N-8.

[32] G. Witus, A. Pandya, S. Hunt, and R. Ellis, "Remote Operators Prefer Goal-based Semi-Autonomous Algorithms (PREPRINT)," ed: ARMY TANK AUTOMOTIVE RESEARCH DEVELOPMENT AND ENGINEERING CENTER WARREN MI, 2011.

[33] S. Y.-S. Lee, "An augmented reality interface for multi-robot tele-operation and control," 2011.

[34] S. Hunt, G. Witus, R. D. Ellis, and A. Pandya, "Remote Operators Prefer Goal-based Semi-Autonomous Algorithms (PREPRINT)," DTIC Document2011.

[35] Y.-S. L. Sam, S. Hunt, A. Cao, and A. Pandya, "Combined Virtual and Real Robotic Test-bed for Single Operator Control of Multiple Robots," *Proc. of SPIE Vol,* vol. 7692, pp. 769208-1.

[36] S. Y.-S. Lee, "An augmented reality interface for multi-robot tele-operation and control," Electrical and Computer Engineering, Wayne State University Dissertations. Paper 381,

2011.

[37] H. A. Yanco, J. L. Drury, and J. Scholtz, "Beyond usability evaluation: Analysis of human-robot interaction at a major robotics competition," *Human–Computer Interaction,* vol. 19, pp. 117-149, 2004.

[38] J. Y. Chen, E. C. Haas, and M. J. Barnes, "Human performance issues and user interface design for teleoperated robots," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 37, pp. 1231-1245, 2007.

[39] J. W. Crandall, M. A. Goodrich, D. R. Olsen Jr, and C. W. Nielsen, "Validating human-robot interaction schemes in multitasking environments," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on,* vol. 35, pp. 438-449, 2005.

[40] M. Cummings, C. Nehme, J. Crandall, and P. Mitchell, "Predicting operator capacity for supervisory control of multiple UAVs," *Innovations in Intelligent Machines-1,* pp. 11-37, 2007.

[41] D. R. Olsen Jr and S. B. Wood, "Fan-out: measuring human control of multiple robots," in *Proc. SIGCHI Conf. on Human Factors in Computer Systems*, 2004, pp. 231-238.

[42] D. R. Olsen and M. A. Goodrich, "Metrics for evaluating human-robot interactions," in *Proceedings of PERMIS*, 2003.

[43] M. A. Goodrich and D. R. Olsen Jr, "Seven principles of efficient human robot interaction," 2003, pp. 3942-3948 vol. 4.

[44] J. W. Crandall and M. L. Cummings, "Identifying predictive metrics for supervisory control of multiple robots," *Robotics, IEEE Transactions on,* vol. 23, pp. 942-951, 2007.

[45] C. D. Wickens, S. E. Gordon, and Y. Liu, *An Introduction to Human Factors Engineering*: Pearson Prentice Hall, 2004.

[46] J. W. Crandall, M. L. Cummings, M. Della Penna, and P. M. de Jong, "Computing the effects of operator attention allocation in human control of multiple robots," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on,* vol. 41, pp. 385-397, 2011.

[47] D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee, "Pheromone robotics," *Autonomous Robots,* vol. 11, pp. 319-324, 2001.

[48] M. Daily, Y. Cho, K. Martin, and D. Payton, "World embedded interfaces for human-robot interaction," in *Proceedings of the 36th Hawaii International Conference on System Sciences*, 2003.

[49] R. T. Azuma, "A survey of augmented reality," in *Presence-Teleoperators and Virtual Environments*, 1997, pp. 355-385.

[50] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality," *Computer Graphics and Applications, IEEE,* vol. 21, pp. 34-47, 2001.

[51] K. Chintamani, A. Cao, R. D. Ellis, and A. K. Pandya, "Improved telemanipulator navigation during display-control misalignments using augmented reality cues," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans,* vol. 40, pp. 29-39, 2010.

[52] A. Pandya, M.-R. Siadat, G. Auner, M. Kalash, and R. D. Ellis, "Development and human factors analysis of neuronavigation vs. augmented reality," *Studies in health technology and informatics,* pp. 291-297, 2004.

[53] A. Steinfeld, T. Fong, D. Kaber, M. Lewis, J. Scholtz, A. Schultz*, et al.*, "Common metrics for human-robot interaction," in *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, 2006, pp. 33-40.

[54] M. Lewis, "Human Interaction With Multiple Remote Robots," *Reviews of Human Factors and Ergonomics,* vol. 9, pp. 131-174, 2013.

[55] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," *Human Factors: The Journal of the Human Factors and Ergonomics Society,* vol. 37, pp. 32-64, 1995.

[56] M. R. Endsley, "Situation awareness global assessment technique (SAGAT)," in *Aerospace and Electronics Conference, 1988. NAECON 1988., Proceedings of the IEEE 1988 National*, 1988, pp. 789-795.

[57] M. R. Endsley, "Measurement of situation awareness in dynamic systems," *Human Factors: The Journal of the Human Factors and Ergonomics Society,* vol. 37, pp. 65-84, 1995.

[58] E. S. Stein, "Air traffic controller workload: An examination of workload probe," U.S. Department of Transportation, Federal Aviation Administration, Atlantic City, NJ DOT/FAA/CT-TN84/24, 1985.

[59] S. Rubio, E. Díaz, J. Martín, and J. M. Puente, "Evaluation of Subjective Mental Workload: A Comparison of SWAT, NASA-TLX, and Workload Profile Methods," *Applied Psychology,* vol. 53, pp. 61-86, 2004.

[60] S. G. Hart and L. E. Staveland, "Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research," *Human mental workload,* vol. 1, pp. 139–183, 1988.

[61] S. G. Hart, "NASA-task load index (NASA-TLX); 20 years later," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 2006, pp. 904-908.

[62] NASA Human Systems Integration Division. (18 Oct 2014). *NASA TLX Publications/Instruction Manual*. Available: http://humansystems.arc.nasa.gov/groups/TLX/tlxpublications.html

[63] J. Veltman and A. Gaillard, "Physiological workload reactions to increasing levels of task difficulty," *Ergonomics,* vol. 41, pp. 656-669, 1998.

[64] A. F. Kramer, "Physiological Metrics of Mental Workload: A Review of Recent Progress," University of Illinois at Urbana-Champaign1990.

[65] A. T. Duchowski, "A breadth-first survey of eye-tracking applications," *Behavior Research Methods, Instruments, & Computers,* vol. 34, pp. 455-470, 2002.

[66] C. H. Morimoto and M. R. Mimica, "Eye gaze tracking techniques for interactive applications," *Computer Vision and Image Understanding,* vol. 98, pp. 4-24, 2005.

[67] D. W. Hansen and Q. Ji, "In the eye of the beholder: A survey of models for eyes and gaze," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 32, pp. 478-500, 2010.

[68] D. D. Salvucci and J. H. Goldberg, "Identifying fixations and saccades in eye-tracking protocols," in *Proceedings of the 2000 symposium on Eye tracking research & applications*, 2000, pp. 71-78.

[69] R. J. Jacob and K. S. Karn, "Eye Tracking in Human-Computer Interaction and Usability Research: Ready to Deliver the Promises," in *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*, J. Hyönä, R. Radach, and H. Deubel, Eds., ed Amsterdam: North-Holland, 2003, pp. 573-605.

[70] R. J. Jacob, "What you look at is what you get: Eye movement-based interaction techniques," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1990, pp. 11-18.

[71] R. J. Jacob, "The use of eye movements in human-computer interaction techniques: what you look at is what you get," *ACM Transactions on Information Systems (TOIS),* vol. 9, pp. 152-169, 1991.

[72] J. H. Goldberg and J. C. Schryver, "Eye-gaze-contingent control of the computer interface: Methodology and example for zoom detection," *Behavior Research Methods, Instruments, & Computers,* vol. 27, pp. 338-350, September 1, 1995 1995.

[73] J. H. Goldberg and J. C. Schryver, "Eye-gaze determination of user intent at the computer interface," *Studies in Visual Information Processing,* vol. 6, pp. 491-502, February 1995 1995.

[74] J. H. Goldberg and J. C. Schryver, "Eye-gaze control of the computer interface: discrimination of zoom intent," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 1993, pp. 1370-1374.

[75] J. H. Goldberg and X. P. Kotval, "Computer interface evaluation using eye movements: methods and constructs," *International Journal of Industrial Ergonomics,* vol. 24, pp. 631-645, 1999.

[76] M. A. Just and P. A. Carpenter, "A theory of reading: from eye fixations to comprehension," *Psychological review,* vol. 87, p. 329, 1980.

[77] A. Hyrskykari, "Utilizing eye movements: Overcoming inaccuracy while tracking the focus of attention during reading," *Computers in human behavior,* vol. 22, pp. 657-671, 2006.

[78] K. Rayner, "Eye movements in reading and information processing: 20 years of research," *Psychological bulletin,* vol. 124, p. 372, 1998.

[79] C. Ware and H. H. Mikaelian, "An evaluation of an eye tracker as a device for computer input," in *ACM SIGCHI Bulletin*, 1987, pp. 183-188.

[80] R. J. Jacob, "Eye movement-based human-computer interaction techniques: Toward non-command interfaces," *Advances in human-computer interaction,* vol. 4, pp. 151-190, 1993.

[81] L. E. Sibert and R. J. Jacob, "Evaluation of eye gaze interaction," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2000, pp. 281-288.

[82] D. Gartenberg, L. A. Breslow, J. Park, J. M. McCurry, and J. G. Trafton, "Adaptive automation and cue invocation: the effect of cue timing on operator error," in *Proceedings of the 2013 ACM annual conference on Human factors in computing systems*, 2013, pp. 3121-3130.

[83] L. A. Breslow, D. Gartenberg, J. Malcolm McCurry, and J. Gregory Trafton, "Dynamic Operator Overload: A Model for Predicting Workload During Supervisory Control," *IEEE Transactions on Human-Machine Systems,* vol. 44, pp. 30-40, 2014.

[84] D. Gartenberg, M. McCurry, and G. Trafton, "Situation awareness reacquisition in a supervisory control task," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 2011, pp. 355-359.

[85] R. M. Ratwani, J. M. McCurry, and J. G. Trafton, "Single operator, multiple robots: an eye movement based theoretic model of operator situation awareness," in *Proceeding of the 5th ACM/IEEE International Conference on Human-Robot Interaction*, Osaka, Japan, 2010, pp. 235-242.

[86] J. Beatty, "Task-evoked pupillary responses, processing load, and the structure of processing resources," *Psychological bulletin,* vol. 91, p. 276, 1982.

[87] S. T. Iqbal, X. S. Zheng, and B. P. Bailey, "Task-evoked pupillary response to mental workload in human-computer interaction," in *CHI'04 extended abstracts on Human factors in computing systems*, 2004, pp. 1477-1480.

[88] J. Klingner, R. Kumar, and P. Hanrahan, "Measuring the task-evoked pupillary response with a remote eye tracker," in *Proceedings of the 2008 symposium on Eye tracking research & applications*, 2008, pp. 69-72.

[89] U. Ahlstrom and F. J. Friedman-Berg, "Using eye movement activity as a correlate of cognitive workload," *International Journal of Industrial Ergonomics,* vol. 36, pp. 623-636, 2006.

[90] L. L. Di Stasi, M. B. McCamy, A. Catena, S. L. Macknik, J. J. Cañas, and S. Martinez-Conde, "Microsaccade and drift dynamics reflect mental fatigue," *European Journal of Neuroscience,* 2013.

[91] T. E. Hutchinson, K. P. White Jr, W. N. Martin, K. C. Reichert, and L. A. Frey, "Human-computer interaction using eye-gaze input," *Systems, Man and Cybernetics, IEEE Transactions on,* vol. 19, pp. 1527-1534, 1989.

[92] A. Doshi and M. M. Trivedi, "On the Roles of Eye Gaze and Head Dynamics in Predicting Driver's Intent to Change Lanes," *Intelligent Transportation Systems, IEEE Transactions on,* vol. 10, pp. 453-462, 2009.

[93] J. C. McCall, D. P. Wipf, M. M. Trivedi, and B. D. Rao, "Lane change intent analysis using robust operators and sparse Bayesian learning," *Intelligent Transportation Systems, IEEE Transactions on,* vol. 8, pp. 431-440, 2007.

[94] M. M. Trivedi and S. Y. Cheng, "Holistic sensing and active displays for intelligent driver support systems," *Computer,* vol. 40, pp. 60-68, 2007.

[95] T. Poitschke, F. Laquai, S. Stamboliev, and G. Rigoll, "Gaze-based interaction on multiple displays in an automotive environment," in *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, 2011, pp. 543-548.

[96] A. Pandya, L. A. Reisner, B. King, N. Lucas, A. Composto, M. Klein*, et al.*, "A Review of Camera Viewpoint Automation in Robotic and Laparoscopic Surgery," *Robotics,* vol. 3, pp. 310-329, 2014.

[97] D. W. Hansen, H. H. Skovsgaard, J. P. Hansen, and E. Møllenbach, "Noise Tolerant

Selection by Gaze-Controlled Pan and Zoom in 3D," in *Proceedings of the 2008 Symposium on Eye Tracking Research and Applications*, 2008, pp. 205-212.

[98] H. O. Latif, N. Sherkat, and A. Lotfi, "TeleGaze: Teleoperation through eye gaze," in *7th IEEE International Conference on Cybernetic Intelligent Systems (CIS)*, 2008, pp. 1-6.

[99] H. O. Latif, N. Sherkat, and A. Lotfi, "Teleoperation through eye gaze (TeleGaze): a multimodal approach," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2009, pp. 711-716.

[100] D. Zhu, T. Gedeon, and K. Taylor, "“Moving to the centre”: A gaze-driven remote camera control for teleoperation," *Interacting with Computers,* vol. 23, pp. 85-95, 2011.

[101] J. Kotus, B. Kunka, A. Czyzewski, P. Szczuko, P. Dalka, and R. Rybacki, "Gaze-tracking and Acoustic Vector Sensors Technologies for PTZ Camera Steering and Acoustic Event Detection," in *Database and Expert Systems Applications (DEXA), 2010 Workshop on*, 2010, pp. 276-280.

[102] S. Ali, L. Reisner, B. King, A. Cao, G. Auner, M. Klein*, et al.*, "Eye gaze tracking for endoscopic camera positioning: an application of a hardware/software interface developed to automate Aesop," *Studies in health technology and informatics,* vol. 132, pp. 4-7, 2007.

[103] D. Fono and R. Vertegaal, "EyeWindows: evaluation of eye-controlled zooming windows for focus selection," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2005, pp. 151-160.

[104] J. L. Wright, J. Y. Chen, and S. G. Lakhmani, "Agent transparency and reliability in human–robot interaction: the influence on user confidence and perceived reliability," *IEEE Transactions on Human-Machine Systems,* 2019.

[105] H. Saeidi and Y. Wang, "Incorporating Trust and Self-Confidence Analysis in the

Guidance and Control of (Semi) Autonomous Mobile Robotic Systems," *IEEE Robotics and Automation Letters,* vol. 4, pp. 239-246, 2019.

[106] M. Desai, P. Kaniarasu, M. Medvedev, A. Steinfeld, and H. Yanco, "Impact of robot failures and feedback on real-time trust," in *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, 2013, pp. 251-258.

[107] D. A. Sanders, B. J. Sanders, A. Gegov, and D. Ndzi, "Using confidence factors to share control between a mobile robot tele-operater and ultrasonic sensors," in *2017 Intelligent Systems Conference (IntelliSys)*, 2017, pp. 1026-1033.

[108] S. Chernova and M. Veloso, "Confidence-based multi-robot learning from demonstration," *International Journal of Social Robotics,* vol. 2, pp. 195-215, 2010.

[109] S. Chernova, "Confidence-based robot policy learning from demonstration," CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE2009.

[110] S. Chernova and M. Veloso, "Teaching collaborative multi-robot tasks through demonstration," in *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*, 2008, pp. 385-390.

[111] A. Tran, "Robot confidence modeling and role change in physical human-robot collaboration," 2019.

[112] J. F. Fisac, A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, S. Wang, C. J. Tomlin*, et al.*, "Probabilistically safe robot planning with confidence-based human predictions," *arXiv preprint arXiv:1806.00109,* 2018.

[113] A. Pronobis and B. Caputo, "Confidence-based cue integration for visual place recognition," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 2394-2401.

[114] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based

augmented reality conferencing system," in *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on*, 1999, pp. 85-94.

[115] H. Kato, T. Mita, S. Hiura, A. Nakazawa, T. Pintaric, P. Lamb*, et al.*, "ARToolKit Library," 2.72.1 ed: GNU General Public License version 2.0 (GPLv2), 2007.

[116] HIT Lab. (15 Oct 2014). *ARToolKit*. Available: http://www.hitl.washington.edu/artoolkit/

[117] HIT Lab NZ. (2004-2006). *ARToolKit API*. Available: http://artoolkit.sourceforge.net/apidoc/index.html

[118] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on,* vol. 4, pp. 100-107, 1968.

[119] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 3400-3407.

[120] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, 2016, pp. 4193-4198.

[121] Raspberry Pi Foundation. (19 Oct 2014). *Raspberry Pi*. Available: http://www.raspberrypi.org/

[122] J. San Agustin, H. Skovsgaard, J. P. Hansen, and D. W. Hansen, "Low-cost gaze interaction: ready to deliver the promises," in *CHI'09 Extended Abstracts on Human Factors in Computing Systems*, 2009, pp. 4453-4458.

[123] J. San Agustin, H. Skovsgaard, E. Mollenbach, M. Barret, M. Tall, D. W. Hansen*, et al.*, "Evaluation of a low-cost open-source gaze tracker," in *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, 2010, pp. 77-80.

[124] C. M. Kohlhoff, "Asio C++ Library," ed, 2003-2015.

[125] N. Lohmann, "JSON for Modern C++," *URL: https://github.com/nlohmann/json*, 2016.

[126] B. Wong, "Points of view: Color blindness," ed: Nature Publishing Group, 2011.

[127] G. Bradski, "The OpenCV Library," *Dr Dobb's Journal Software Tools,* vol. 25, pp. 120-125, 2000.

[128] D. v. Heesch, "Doxygen," ed, 1997-2018.

[129] J. Y. C. Chen, "Effects of operator spatial ability on uav-guided ground navigation," in *Proceeding of the 5th ACM/IEEE International Conference on Human-Robot Interaction*, Osaka, Japan, 2010, pp. 139-140.

[130] J. Y. C. Chen and M. J. Barnes, "Robotics operator performance in a military multi-tasking environment," in *Proceedings of the 3rd ACM/IEEE International Conference on Human-Robot Interaction*, Amsterdam, The Netherlands, 2008, pp. 279-286.

[131] J. Y. C. Chen, "Concurrent performance of military tasks and robotics tasks: effects of automation unreliability and individual differences," in *Proceedings of the 4th ACM/IEEE International Conference on Human-Robot Interaction*, La Jolla, California, USA, 2009, pp. 181-188.

[132] B. M. Bolker, M. E. Brooks, C. J. Clark, S. W. Geange, J. R. Poulsen, M. H. H. Stevens*, et al.*, "Generalized linear mixed models: a practical guide for ecology and evolution," *Trends in ecology & evolution,* vol. 24, pp. 127-135, 2009.

[133] D. J. Barr, R. Levy, C. Scheepers, and H. J. Tily, "Random effects structure for confirmatory hypothesis testing: Keep it maximal," *Journal of memory and language,* vol. 68, pp. 255-278, 2013.

[134] R. H. Baayen, D. J. Davidson, and D. M. Bates, "Mixed-effects modeling with crossed random effects for subjects and items," *Journal of memory and language,* vol. 59, pp. 390-412, 2008.

[135] D. Bates, R. Kliegl, S. Vasishth, and H. Baayen, "Parsimonious mixed models," *arXiv preprint arXiv:1506.04967,* 2015.

[136] B. M. Bolker, M. E. Brooks, C. J. Clark, S. W. Geange, J. R. Poulsen, M. H. H. Stevens*, et al.*, "GLMMs in action: gene-by-environment interaction in total fruit production wild populations of Arabidopsis thaliana," vol. Revised version, part 1, pp. 127-135, 2011.

[137] R Core Team. (2019). *R: A Language and Environment for Statistical Computing*. Available: https://www.R-project.org/

[138] D. Bates, M. Mächler, B. Bolker, and S. Walker, "Fitting Linear Mixed-Effects Models Using lme4," *Journal of Statistical Software,* vol. 67, pp. 1-48, 2015.

[139] U. Halekoh and S. Højsgaard, "A kenward-roger approximation and parametric bootstrap methods for tests in linear mixed models–the R package pbkrtest," *Journal of Statistical Software,* vol. 59, pp. 1-30, 2014.

ABSTRACT

MULTIROBOT CONFIDENCE AND BEHAVIOR MODELING: AN EVALUATION OF TELEROBOTIC PERFORMANCE AND EFFICIENCY

by

NATHAN PAUL LUCAS

December 2019

Advisor: Dr. Abhilash Pandya

Major: Computer Engineering

Degree: Doctor of Philosophy

There is considerable interest in multirobot systems capable of performing spatially distributed, hazardous, and complex tasks as a team. There is also growing interest in manned-unmanned teams leveraging the unique abilities of humans and automated machines working alongside each other. The limitations of human perception and cognition affect the ability of operators to integrate information from multiple mobile robots, switch between their spatial frames of reference, and divide attention among many sensory inputs and command outputs. Automation is necessary to help the operator manage increasing demands as the number of robots scales up. However, more automation does not necessarily equate to better performance.

This research developed novel techniques applicable to user interface designs for the remote operation of multiple unmanned vehicles. A generalized robot confidence model was introduced which transforms an arbitrary number of indicators of operator attention to a confidence value for each robot in order to enable adaptive behaviors for an arbitrary number of robots. The model was implemented and successfully evaluated to reveal evidence linking average robot confidence to multirobot search task performance and efficiency. The contributions of this work provide important steps toward effective human teleoperation of

multiple mobile robots to perform spatially distributed and hazardous tasks in complex environments for space exploration, defense, homeland security, search and rescue, and other real-world applications.

AUTOBIOGRAPHICAL STATEMENT

Nathan Lucas is a doctoral candidate in Computer Engineering at Wayne State University. He holds a Bachelor of Science in Electrical Engineering and a Master of Science in Electrical and Computer Engineering from Lawrence Technological University. His research interests include human-computer interaction, user interface design, and multirobot systems.