UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

ATTITUDE AWARE SMARTPHONES FOR TELE-OPERATED ROBOT CONTROL

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

AMBER M. WALKER
Norman, Oklahoma
2013

ATTITUDE AWARE SMARTPHONES FOR TELE-OPERATED ROBOT CONTROL

A DISSERTATION APPROVED FOR THE SCHOOL OF AEROSPACE AND MECHANICAL ENGINEERING

BY

Dr. David Miller, Chair

Dr. M. Cengiz Altan

Dr. Dean Hougen

Dr. Chen Ling

Dr. Zahed Siddique

Acknowledgements

First, I wish to thank my research adviser, Dr. David Miller, who welcomed me to Oklahoma and took on more than just a student. I have been demanding and strong-willed, and *really* tested my limits by having a baby in the middle of an already accelerated degree. Thanks for believing in me! I have learned about much more than robotics. You took the time to develop me as a researcher, mentor, and teacher. My future cadets thank you!

To my committee, I thank you for your advice, service, and accommodation. I have enjoyed working with each of you. I have especially enjoyed the multi-disciplinary nature of my degree, and each of you had a hand in ensuring its success. Computer science and human factors were very new to me three years ago, but not so now thanks to your confidence, time, and patient instruction.

I also wish to thank my military mentor, COL Daisie Boettner, who has been inspiring me since I took her Fluid Dynamics class at West Point 11 years ago. She will soon be my boss, and I could not ask to work for a more dedicated professional.

Finally, I must thank my family...all of them. Especially my husband, Michael, who has been through all of life's crazy rides with me. He is a truly equal partner, making everything I've accomplished possible. To my son, Tyson, I hope that one day you'll at least read to this page of my dissertation and be inspired! The world is your oyster.

Table of Contents

List of Tables

List of Figures

Abstract

The U.S. military has increasingly turned to unmanned ground vehicles (UGVs) to assist in the most dull, dirty, and dangerous missions. Their presence on the battlefield is redefining how war is waged, expanding opportunities for reconnaissance and surveillance while minimizing soldier mortality. Robotic systems have gotten ever smaller, many now being man-packable. Soldiers may now carry, deploy, and control their own robotic assistant, many with limited formal training. Unfortunately, UGVs add significantly to a soldier's standard load of water, ammunition, armor, and supplies, making weight and portability top concerns. One way to ease the soldier burden is to adopt smartphones for use as robot operator control units (OCUs). Their small, lightweight frame combined with processing power and adaptable software backbone may enable intuitive controls on a device well-suited for other military missions.

Field operations are often conducted when users are gloved and/or dirty, making common smartphone touch interfaces problematic. By using proprioceptive device inputs related to *attitude*, smartphones can be used for control in ways that minimize that touch interface. To test this, an attitude aware smartphone controller (using the device's accelerometers and gyroscope) for a small, tele-operated, ground robot was developed and assessed via a multi-phase usability experiment. The controller's motion algorithm made use of quaternion mathematics to simplify motion handling and the user interface.

Twenty-five users were recruited to assess usability of attitude aware controls, testing their suitability for driving and camera manipulation tasks. Participants operated a small tracked robot on an indoor course with controllers using either

virtual joystick or tilt-based controls while metrics regarding performance, mental workload, and user satisfaction were collected. They were also exposed to customizable controls, identifying modes and settings which contributed most heavily to the controller's usability. Results indicate that attitude-based controls are suitable for tele-operated reconnaissance and surveillance, as 64% of users preferred tilt-based driving controls while performing equally as well as the alternative. Customized configurations showed 60% of users preferred tilt for driving tasks when throttle sensitivity and controller responsiveness could be manipulated. The inherent usability of attitude aware controls optimistically exhibit how smartphones can be leveraged for robotic control, even in harsh environments by gloved users.

Chapter 1

Introduction

1.1 Background

Robots are often said to excel at the "dull, dirty, and dangerous," making them uniquely suited to military missions [109, 119, 48]. As tools on the battlefield, unmanned systems are becoming indispensable; the number of robots in combat is consistently growing. 2011 estimates indicate a 1:50 robot to soldier ratio in Afghanistan, expected to increase to 1:30 by the end of 2013 [131]. As specialized *field* robots, military unmanned systems have been designed to reliably complete a variety of tasks in unstructured and unpredictable operating environments while remaining immune to fatigue, disease, emotion, and environmental discomfort [119, 15].

While unmanned aerial systems were some of the first to be put into military operation, **unmanned ground vehicles** (UGVs) are growing more crucial to the Army and Marine Corps' missions. They are generally cheaper, easier to deploy and maintain, and more mobile than their aerial counterparts. Additionally, very little "specialized" training is needed to operate UGVs; a vast difference from the trained pilots currently flying unmanned aerial vehicles (UAVs). Carlson and Murphy define a UGV as "a ground-based mechanical device that can sense and

interact with its environment" [21]. To date, the most commonly deployed UGVs are found in explosive ordnance disposal, primarily due to their success at saving lives by keeping soldiers out of harm's way. Commanders are taking notice, asking for robots with a greater range of capabilities and redefining how war is waged at the lowest levels [2]. UGVs are now being used (or developed) in support of reconnaissance and surveillance, logistics and support, communications, and combat missions [4].

To support this evolving mission set, robots are being miniaturized. Portable, man-packable robots are highly deployable, making them uniquely suited to small-unit reconnaissance and search and rescue. In these missions, UGVs are almost entirely employed out of line of sight, referred to as *tele-operation*. Users must rely on video feedback from one or more onboard cameras to navigate and surveil. Since even sophisticated UGVs cannot adequately discriminate between friend and foe, or make decisions regarding complex situations (on the battlefield and beyond) [4], the user interface is a critical tool allowing the *operator* to make these decisions. Interestingly, while robots have gotten increasingly smaller, efforts to shrink the operator interface have not proceeded apace.

1.2 Motivation

Historically, most large-scale technologies and weapons systems are researched, developed, and procured as very structured military "programs of record." Battlefield robots, however, have been purchased primarily as commercial systems developed by independent companies anticipating the demand. As such, the robots currently deployed by the military are a hodgepodge of different platforms, operator interfaces (controllers), batteries, and communications that did not

undergo extensive field testing prior to their use in Iraq and Afghanistan. Interestingly, the absence of comprehensive field testing not only affects the reliability and success of the robot platform, it also effects its *controller.* Controllers may be called by many names–operator interface, user interface, or operator control unit (OCU) are all common. As such, the term *controller* should be taken to mean **hardware,** making no assumptions about the user or operator with whom it interfaces.

Researchers like Adams and Nguyen and Bott have examined the development of controller interfaces as they relate to user populations. All have noted systemic issues with users' involvement in the design process coming too late to impact change [1, 84]. This is termed a lack of "user-centric" design, and can significantly affect the quality and usability of products that rely so heavily on user involvement. In the case of military robots over the past ten years, the operational need to ship robots overseas and the limited access to military robot operators have resulted in a product which soldiers had no ability to shape.

1.2.1 Current Operator Control Units (OCUs)

For many years robot control has been achieved via laptop-based systems originally designed for desktop use then crudely converted for field operations. As robots and their users have become consistently more mobile, however, the laptop has become a cumbersome control option. Various versions exist, whether they be worn around the user's neck or carried in a backpack with a tethered handheld component (see Figure 1.1a). Aside from their size, many of these systems are inherently complicated for potentially under-trained operators desiring a capable, intuitive system.

On the whole there has been a lack of significant progress on portable, handheld operator control unit technologies which eschew the standard single-mode, tactile joystick approach. Several companies have recently unveiled updated controllers, but all have made concerted efforts to avoid touchscreens without conducting research into whether or not they could be beneficial. Recon Robotics currently deploys their Throwbot® XT with a simple handheld device where video and joystick are housed on a device weighing 1.6 pounds [105] (Figure 1.1b); iRobot has upgraded to a small, 2 pound handheld controller with 5" LED screen for their FirstLook® robot [59] (Figure 1.1c); and Applied Research Associates (ARA) uses something similar for their Pointman® robot [10] (Figure 1.1d).

1.2.2 Smartphones in the Military

Smartphones are of interest to the military for multiple reasons–communications, adaptability, and "disposability." In fact, the U.S. Army is in the midst of arming soldiers with these devices, following the success of recent tests at White Sands and Fort Bliss training areas [77]. While there are still unanswered questions regarding screen glare, ruggedization, and battery life, the Army, at least, is committed to continuing development towards a standardized military smartphone and operating system.

Initially, these smartphones will connect to the secure military network via Rifleman Radio and provide GPS data, map overlays, and other situational awareness tools [49]; however, industry developers are already in line to expand those capabilities to include applications for logistics, maintenance, and more. The Department of Defense's Robotic Systems Joint Project Office is hopeful

(a) Laptop-based Controller

(b) Recon Robotic's Throwbot Controller

(c) iRobot FirstLook Controller

(d) ARA Pointman Controller

Figure 1.1: Operator Control Unit (OCU) Examples

that one of these applications will enable smartphones for use as future unmanned system OCUs [109].

Smartphones play host to one of the most promising new controller platforms since the advent of modern video games. Their unique combination of processor power, size, and high-resolution displays makes them inherently adaptable–one of the military's doctrinal requirements for robotics systems [37]! While smartphones with tactile keyboards do exist (e.g. Blackberry), the majority are touchscreen devices. These offer the most flexibility for robotics applications, as the device's large screen is available for video display and operator feedback. Unfortunately, touchscreen technology has been slow to be accepted by military personnel due to its disadvantages when compared to traditional tactile controls:

- Touchscreen operation is difficult when hands are dirty or gloved.
- The devices are viewed as flimsy, breakable, and fragile. While this belief may not be factually supported, users continually claim fragility as a reason to avoid touchscreen use. Given the availability of rugged, waterproof cases and the fact that many soldiers successfully carry personal devices of similar make and model (iPods, digital cameras), it is an unfounded, albeit common, complaint [67].
- Small screens limit the resolution for remote viewing and are easily obscured by fingers operating the controls.
- Touchscreens lack physical feedback.

While many users feel strongly about the deficiencies in this list, none are insurmountable. Some can be addressed through small changes to the hardware,

others may be managed via chosen control mode, and perceived fragility will likely only be eliminated by successful, widespread use.

1.3 Problem Definition

By limiting the touches necessary to operate in field environments, *attitude-based control* offers a less demanding, intuitive interface which addresses nearly all of the disadvantages of current OCUs. The intent of this body of work is to provide an option for smartphone-based robot control that meets the needs of dismounted troops using small, portable unmanned ground vehicles (UGVs) for reconnaissance and surveillance. A usability experiment is presented which tests the suitability of an *attitude aware* smartphone controller while identifying heuristics governing its design. By exploiting onboard micro-electromechanical sensors, such as accelerometers and gyroscopes, proprioceptive device inputs (tilt and rotation) can be leveraged to overcome common user complaints regarding touchscreens in field environments.

Defining *Attitude*

Attitude aware refers to an object's three-dimensional pose with respect to its inertial frame (gravity). Smartphones are attitude aware thanks to their accelerometers and gyroscopes, measuring linear motion and rotation respectively. By monitoring device pose, and changes to it, applications can respond to custom gestures, recognize orientation changes, and track dynamic motion. This can improve the user experience by providing a more physical mode of feedback, removing clutter from the display, and mapping to mental models more intuitive to operators. Throughout this paper, **attitude** will be

used to reference this **control modality** and should not be confused with user attitudes or psychological applications of the word.

1.4 Objectives & Scope

Robot use is increasing, the missions they support are becoming more diverse, and more soldiers are therefore becoming operators. Ease of use, size, weight, and portability must therefore be emphasized in all design efforts. To date, development has outpaced military doctrine, resulting in a lack of well-defined requirements and numerous training gaps. Currently the U.S. military boasts an array of different UGVs by different manufacturers, all with their own unique controller. Some of these are handheld devices and some are still laptop sized; however, very few of them share features, limiting interoperability between robot platforms.

Nguyen and Bott, in a survey of law enforcement robot operators, identified that the

> most important criterion for a successful program is producing an end product that the user will use and appreciate. Closing the loop with the user should therefore be the number one priority throughout the design and development process [84].

This project addresses the specific need for a smartphone-based OCU that overcomes the previously stated issues regarding touchscreens in military applications. Operator control unit methodologies and modes that promote simple, efficient control of small reconnaissance robots in use by dismounted ground forces during contingency operations will be evaluated.

1.5 Dissertation Outline

This dissertation is organized into eight chapters; an Introduction (Chapter 1) and Conclusion (Chapter 8) plus six "body" chapters. Chapter 2 provides background knowledge necessary for readers to understand the subjects present in this research, specifically tele-operation, the tilt-based control modality, human factors and usability, and control customization and the user experience. Chapter 3 presents the attitude aware controller prototype. It defines a controller use case, identifies design considerations, presents the motion algorithm for tilt-based controls, and (for the programmers in the audience) goes into substantial detail about the application's functions, prototypes, and structure.

Chapter 4 introduces the three-phase experiment designed to test usability. Procedures and equipment common to all three phases are presented, while details regarding individual phase characteristics are withheld until presentation of results, each in their own chapter, as outlined in the following section.

1.6 Research Questions & Hypotheses

Chapters 5-7 provide results individual to each phase, while Chapter 8 summarizes the series and looks at the broader impact of these results. Guided by two underlying questions, three hypotheses were developed to inform the multi-phase experiment design. The hierarchical presentation to follow nests hypotheses with research questions and identifies the experiment phase and chapter in which tests of each hypothesis are presented.

1.6.1 Research Question #1

(R.1) *Can the operator control unit for a tele-operated UGV be implemented on an attitude aware smartphone such that the advantages of that device platform (performance, usability, size/weight) overcome suspected deficiencies, including negative user perception?*

While users are expected to be skeptical of new, touchscreen technologies, pairing attitude aware controls with a large handheld screen available for remote video viewing, eliminates the need for a touch interface. Therefore, smartphones could prove *usable* when employing attitude aware controls, especially considering their other advantages, namely size and adaptability.

Chapter 5: Phase 1

To test this assumption of usability, the first hypothesis formed the basis for experiment Phase 1. It built off of related work by the Army Research Laboratory (ARL), who had previously designed a smartphone-hosted virtual joystick controller [96]. Phase 1 compared that controller to attitude-based controls when tele-operating (driving) on an indoor course. Given that all current OCUs use one or multiple tactile joysticks, the touchscreen joystick was expected to be a familiar, if less physical, control modality. Experiment trials tested users' driving skills, focusing on control suitability with regards to an operator's primary task.

(H.1) *Over time, and after reasonable training, users will be able to perform surveillance and reconnaissance tasks to a reasonable standard and equally as well with tilt inputs as with a virtual joystick.*

Chapter 6: Phase 2

Phase 2 expanded the controls from Phase 1 and enabled additional robot "degrees of freedom." While Phase 1 examined suitability of attitude-based controls for driving, Phase 2 tests their usability for camera control as well. Many tele-operated UGVs have a pan/tilt camera mount to improve situational awareness and increase the robot's field of view. Therefore, to fully answer the research question posed, controls must be tested for both driving *and* surveilling–crucial tasks in common military missions.

(H.2) *Tilt-based controls are intuitive enough a control modality to be used for a number of robotics applications with multiple degrees of freedom, without significant dedgradation of performance.*

1.6.2 Research Question #2

(R.2) *Can controller customization options entice users to spend more time with the device, become more personally invested in its use, and perform better while using it for tele-operated surveillance and reconnaissance tasks with a ground robot?*

This research question was motivated by several related factors. Given user reluctance to adopt touchscreens for military applications, customizable control was posited to draw the user in, provide a sense of ownership, and hold the

user's attention. Users are often quick to dismiss technologies that are new or unfamiliar; by providing customizable control, users might not give up so soon. In addition to enticing the user to spend time with the device, customizable controls are also likely to improve user satisfaction. Who wouldn't prefer something they could customize?! That said, are preferences selected by novice operators likely to improve performance, or are well-engineered defaults superior?

Chapter 7: Phase 3

Phase 3 capitalizes on the operators' training in Phases 1 and 2 and presents them with a customizable controller interface for both driving and camera channels. Users may choose from either tilt or joystick control modes and can customize sensitivity and responsiveness through an easily accessed settings menu.

(H.3) *Permitting customization of controller layout and functionality will lead to a more satisfied user with better performance metrics.*

All readers may benefit from the background topics addressed in Chapter 2, while software developers will want to then focus on Chapters 3 and 7. Industrial engineers, or those interested in usability, will find Chapters 4-7 relevant. Military officers and procurement specialists should start with a summary of the findings, as well as recommendations for employment of the attitude aware prototype, in Chapter 8.

Chapter 2

Literature Review

This chapter will provide an introduction to the subjects necessary to appreciate the multi-disciplinary foundation of this research. Tele-operation, the primary control mode for military ground robots, is defined and discussed, as well as a look at the progress in tilt-based control modalities. Human factors and usability is likewise important and described alongside customizable control as a component of the user experience.

2.1 Tele-operation

Currently, the military is primarily invested in robots that are tele-operated, or "manually controlled by an operator at a distance that is too great for the operator to see what the robot is doing" [21, 81]. A human user is crucial to operations in unknown environments, where their flexibility and expertise are needed to interpret and make judgments based on remote feedback [128, 123]. Historically, military robots have used *direct* tele-operation interfaces supporting real-time decision making. These are often hand controls, like 3-axis joysticks, paired with video feedback available from a robot mounted camera. Fong and Thorpe describe this as "inside-out" driving, because the operator feels

as if he/she is inside the vehicle looking out [42]. Tele-operation tends to be challenging because operator performance is "limited by the operator's motor skills and his ability to maintain situation awareness...difficulty building mental models of remote environments...[and] distance estimation and obstacle detection" [41, 75].

Winfield describes the control interface as one of three key components of any tele-operated system. The other two are the communications link and the robot. Multiple communication methods are used based on robot system and mission; however, they must be full duplex (two-way) to accommodate commands from the interface while providing vision and sensor data from the robot [128]. Military ground robots are primarily controlled over radio frequency links, while aerial vehicles and space rovers are almost always connected via higher latency satellite links.

van Erp conducted a thorough survey of tele-operation and operator tasks with a specific interest in improving the user interface without taxing the communications link. He discusses at length the human operator's superiority at obstacle avoidance at high speeds and expertise regarding camera control and scanning (i.e. knowing where to look); however, he notes the difficulty for operators in the sensory deprived state inherent to tele-operation [123]. To combat this sensory deprivation, van Erp suggests several heuristics for "task-critical information in UGV control" specific to the image system:

1. The field size of normal drivers is 140°. Smaller fields of view limit peripheral vision and may cause drivers to initiate their control actions earlier than optimal [124, 23]. van Erp recommends a minimum field size of 50°, with 100° preferred.

2. While fields of view may be artificially expanded by applying a magnification factor less than 1.0, van Erp has proven this disastrous for driving performance as it alters the user's perception of speed and distance and transfers less object motion to the display.

3. van Erp quotes his own experimental results to show that the required camera update rate, while task dependent, ranges between 3 to 10 Hz.

4. Color images are preferred in rough terrain, especially when the interface lacks stereoscopic depth cues and/or when image quality is degraded.

5. van Erp acknowledges the advantages of a variable view camera (e.g. pan and tilt camera implementation), providing the operator a larger periodic field of view. Unfortunately, many operators have difficulty determining the viewing direction of the camera compared to the vehicle's heading. He recommends providing adequate vehicle references within the camera field of view, much like his requirement for some portion of the vehicle to be visible in the camera frame for driving.

6. Murphy introduced another consideration for camera placement, presenting evidence of unnatural viewing angles for small vehicles low to the ground [82]. Where vehicle height is not an issue, cameras can be placed on masts (often in stereo pairs) to improve this visual mismatch while increasing field of view. Several authors have examined optimal settings for mobile robots, specifically mast height, elevation of the camera(s),and the focal length of the lens [56, 63].

Chen et. al. examined issues similar to van Erp, surveying more than 150 papers covering human tele-operation performance issues and mitigation

solutions. She states that most human performance issues fall into two categories, either remote perception (already covered in some detail in [123]) or remote manipulation [23]. Her analysis of time lag fills in on a component not present in van Erp's work, noting that system latencies exceeding one second force users to adopt a "move and wait" strategy, with variable lags being more detrimental than fixed ones [68]. Extensive research has been done to characterize these delays, as well as suggest mitigation strategies [33, 20, 24, 78]. Some of the earliest work came out of MIT with Thomas B. Sheridan beginning in 1965 [40], and was followed by his work looking more specifically at supervisory control [114, 113].

Chen's summary also emphasizes the importance of, in addition to visual feedback, providing cues to aid in the operator's sense of orientation. "Track-up maps" (vehicle point of view) have proven superior for local navigation, while north-up maps enhance global awareness [23, 127]. Where available, robot attitude should also be fed back to the user, as operators are not always aware of a robot's precipitous position (i.e. on a steep grade or uneven ground) until it is too late and the robot cannot self-right [75]. Visual display elements are often integrated into a comprehensive representation of robot status using overlays; however, research indicates this often leads to higher perceived mental workload.

Chen concludes her work with a brief survey of available input and output modes in support of multimodal control. Speech inputs and auditory feedback are considered natural and effective for most users and provide hands-free control options. Similarly, gesture-based control using body and arm pose, hand gestures, or even facial expressions, allow for a wide variety of control and are also often hands-free.

Using this knowledge, the attitude aware controller developed for testing ascribed to many of the design guidelines introduced. The vehicle chassis is

visible in the camera's frame while driving, using a color camera with 60° field of view. The camera was mounted approximately 4.5" above the top of the robot's body to bring the camera's reference frame further off the ground. Network latency, while varied, was generally under one second, well within Chen's recommended range [23]. Feedback, while not robust, was addressed primarily through subtle visual indicators in combination with physical device pose/translation. Every effort was made for the robot and controller to abide by well-documented heuristics while identifying new ones more specific to the tilt-based control modality.

2.2 Tilt-based Research

Gestural inputs, as in Chen's survey, usually describe a vision system interface that tracks human gestures [71]; however, they can also be used in conjunction with handheld devices such as Apple's iPhone or Nintendo's Wii [14, 129, 99, 54]. So as not to confuse the reader, gestures specific to handheld devices using accelerometers and gyroscopes will be referred to instead as **attitude aware** or **tilt** controls throughout this paper. A full definition will be provided in Chapter 3.

Rekimoto did some of the earliest work with tilt controls on small screen handheld devices, using a Personal Digital Assistant (PDA) [107]. His work was motivated by the need to refine desktop applications for use on smaller systems, leading to development of tilt-based inputs to control menus, scroll bars, and maps. Experiments indicated that users could manipulate on-screen menus fairly accurately with just 2° of tilt, while normal operations saw users tilting the device between 10-15° [107]. These are some of the earliest reported tilt-based

interaction metrics, which have since been further refined by researchers like Hinckley et. al. [54] and Rahman et. al. [103].

Ken Hinckley, a researcher for Microsoft, is one of the scientists credited with prototyping and testing the mobile interaction technologies that would gain stunning popularity in the touchscreen devices of the early 2000s (e.g. Apple's iPhone) [53]. He started by adding a suite of sensors to a PDA in 2000 and writing software to control device behaviors based on those sensor inputs. He presented experiments where the device was outfitted with proximity sensors to detect when a person brought it towards his/her face, automatically detected device orientation to update the screen configuration (landscape vs. portrait mode), and used pressure sensitive sensors to determine when the device was being held [54]. His goal was to create interactions that were minimally disruptive while minimizing cognitive demands and/or visual attention. He emphasized the need for future work on the subject, stating that:

> While interactive sensing techniques seem to provide many benefits, they also increase opportunities for poor design because the strengths and weaknesses in the design space are not as well understood as traditional GUI design. We need experiments to quantify user performance with these techniques, and we need longitudinal studies to determine if users may find sensing techniques "cool" at first, but later become annoyed by false positives and negatives, for example [54].

Hinckley has spent the years since that article (published in 2000) designing, experimenting, and writing textbooks [55], cementing his expertise in human-computer interaction technologies and input techniques. He encourages developers to embrace multimodal controls, stating, "We shouldn't try to do everything with any one [mode]. We should instead seek to understand how input modalities and techniques can complement one another, such that the advantages

of one make up for the shortcomings of others." He cautions, however, that "*excellence* in user interface design requires tailoring the interface to the input method" [55].

Rahman et. al. took a more physiological approach to designing for multimodal interaction, examining tilt in terms of human anatomy and focusing on the general level of control possible with the wrist [103]. He quantifies three physical axes of rotation: flexion/extension (60/45°); pronation/supination (65/60°); and ulnar/radial deviation (15/30°). The motion algorithm used relied on discretization of the tilt-space, recommending a quadratic function where "the extremities of the range of motion are given the largest amount of space and less angular tilt is allocated to the middle of the range" [103]. The authors also introduced a *click-tilt-release* interaction, where tilt controls were activated by first pressing a button and de-activated when that button was released. This was found to provide a sense of recalibration reference the point of origin, helping to reduce confounding effects.

Several other researchers have presented work using accelerometers and gyroscopes for tilt-based interactions [61, 89]. Jang and Park looked at using PDA-mounted accelerometers to recognize gestures, comparing the sensor's signal pattern to a "known" gesture pattern. Their work provided methods for recognizing and filtering both static and dynamic accelerations, introducing the idea of a "threshold" value and programmatic deadzones to eliminate inadvertent user inputs [61]. Pitman and Cummings, Muller, Piskorski [99, 80, 97] and others present tilt-based interfaces for robotics controls and/or gaming. A full survey of their implementations, motion algorithms, and interface strengths and weaknesses is presented in Table 3.1 in Chapter 3.

Notable lessons learned from tilt-based research include the need for a deadman switch, following Rahman's *click-tilt-release* interaction methodology [103]. This ensures that tilt controls are not always active and helps limit inadvertent control inputs. Likewise, it aids in recalibration around the point of origin, allowing users more precise control over the device's behavior. Rekimoto's work showing that users can manipulate controls with just 2° of tilt provides a strong basis for the level of precision available to robotic control implementations, reinforcing its feasibility [107]. Jang's work with thresholds and deadzones [61] inspired use of programmatic limits around the controller's origin, ensuring that user inputs are distinctly large enough to warrant robot manipulation. Each of these attributes plays towards controller usability, the crucial characteristic of interest in this research.

2.3 Human Factors/Usability

While engineering and application development are crucial to the design of most robotic platforms, an acute understanding of users and *human factors* is needed to successfully develop a robotic *system*. Human factors is the study of designing "the things people use and the environments in which they use [them] to better match the capabilities, limitations, and needs of people" [110]. Mental models, linked closely to affordances, are an individual's perception of the world around them and how things *should* act, or interact [87]. Compatabilities often inform these models and can be either intrinsic or learned. Intrinsic compatabilities are natural mappings rooted in the system design, for instance turning a steering wheel left to turn left. Culturally acquired, or learned, compatabilities may differ between cultures and are often not as obvious; as an example, light switches are

commonly flipped up to turn on in the United States, but are pushed down in several other countries [110].

In order to design for these compatabilities and mental models, they must first be understood. Donald Norman has written several best-selling books discussing "human-centered design," including *The Design of Everyday Things* [86] and *The Design of Future Things* [87]. He provides easy-to-follow guidelines for the design of virtually any system [87]:

1. Provide rich, complex, and natural signals.
2. Be predictable.
3. Provide a good conceptual model.
4. Make the output understandable.
5. Provide continual awareness, without annoyance.
6. Exploit natural mappings to make interaction understandable and effective.

While generic, these heuristics propose a path to making human factors a systemic part of the design process. Norman states that "the ability of a person to discover and make use of affordances is one of the important ways that people function so well, even in novel situations when encountering novel objects" [87]. Just how well this has been done is often assessed via *usability*.

Goodrich and Olsen produced a series of work attempting to quantify human-robot interactions in a methodical manner. The result was seven principles to make human-robot interactions, specifically tele-operation, more efficient, resulting in more *usable* systems [46, 91, 47, 38]. They recommend implicitly switching interfaces or modes without cognitive effort or attention, such

as switching to manual control automatically when a joystick is grasped. They also suggest that robots should use natural human cues, supporting efforts in speech control, sketch-based maps, or point-to-move interfaces [108, 19]. Allowing users to manipulate the information presented is also deemed prudent, as is designing to help manage their attention. Donmez attempted to further work on defining metrics applicable to human-robot interaction by looking beyond the performance of a robot system to measure *human performance* explicitly. He identifies three metrics important to human performance: situation awareness, workload, and mental models of device operations [38, 116].

"Usability comprises ease of use and usefulness, and these drive user satisfaction. User satisfaction, in turn, results in usage" [70, 60]. ISO 9241, *Ergonomics of Human-System Interaction* [58, 62] focuses on effectiveness, efficiency, and satisfaction. Jokela et. al. examines the definition of ISO 9241 and its implementation in designing for usability [62]. Shneiderman summarizes it into five **usability measures** [115]:

1. *Time to learn.* How long does it take for a typical user to learn the system, specific to a set of tasks?

2. *Speed of performance.* How quickly can users carry out those tasks?

3. *Rate of errors by users.* How often do users make errors, and of what type?

4. *Retention over time.* How well do users retain knowledge gained about system operation? Hours? Days?

5. *Subjective satisfaction.* This is a measure of how well users like various aspects of an interface, often collected via interview or survey.

By examining these metrics and prioritizing them, as all systems may demand a different balance of, for instance, error handling and speed, systems can be designed for the ultimate goal: addressing the needs of all users [115]. Shneiderman proposes that designing for differing situations actually results in a better product for all users. While his examples include sidewalk access for disabled users also benefiting parents with strollers, cyclists, etc., the benefits of designing broadly are also relevant in the military, where soldiers attempt to use the same equipment for a myriad of missions in a variety of environments.

Designing for usability inspired both the work done during controller development and the experiment procedures adopted to assess the final attitude aware controller. Inspired by common mental models, the controller prototype was designed to mimic driving a vehicle with a steering wheel. Interfaces remained simplistic, with only one screen/menu for all controls (no switching required). Following conventions of research into human performance, which commonly includes mental workload, the NASA TLX Task Load Index [38, 51] was administered to all users conducting trials as part of the attitude aware usability experiment. Additionally, best practices indicate that effectiveness, efficiency, and satisfaction be quantified [58, 115] in usability assessments; these were collected in each experiment phase as a combination of dependent variables: practice time, trial time, number of driving errors, and user satisfaction (measured via survey).

2.4 Designing for the Masses–Customizable Control

A 2007 study by Synovision, commissioned by the Department of Defense (DoD), was performed to provide guidance to DoD regarding how best to acquire a

common robotic controller based on the survey of technologies available at that time. The controllers used in their study are reported in the report's appendix [117], but the authors felt strongly that "the human factor in robotic control is soldier specific and more qualitative than quantitative in nature." There was no common "intuition" among soldiers regarding number of buttons, location of those buttons, etc., seemingly implying the need for not only more study regarding the users and uses of future common controllers, but also the inherent adaptability of any controller designed to suit this purpose [117].

Several researchers have conducted work to look at customization of desktop software systems, like word processors, to gain insight into user demands and preferences [73, 74, 93]. The theory exists that customization makes users more productive because they can tailor their software to their work. In order to confirm (or deny) this theory, studies were executed to identify the frequency with which users customized options, and which options were manipulated. In a 1996 study by Page et. al., 92% of participants customized their software in some way over the course of 28 days, with the heaviest users making the most changes [93]. High incidences of button removal (44% of users) on one of the shortcut bars implied that defaults were poorly chosen, but also supported the premise that features simple to adapt (like the button bar) would experience higher incidences of customization [93]. His recommendations were to design for casual users less likely to manipulate settings and more likely to "satisfice" rather than optimize [74, 93], while expecting users to to make at least some modifications.

Mackay, in 1991, identified four categories of events which serve as triggers to customization:

1. *External events.* Changes of job or office. Things likely to make users re-evaluate how they manage their time and software.

2. *Social pressure.*

3. *Software changes.* Upgrades, crashes, etc. Interestingly, most users that customize do so to make new software feel more like old software, often discarding or ignoring the newest features [73].

4. *Internal factors.* These are generally time related, where people are bored or have time to spare to try something new and/or repair a recurring problem which had previously gone ignored.

Social pressure can take the form of peer pressure, which is proven to play a large role in user customization. Every user in Mackay's study borrowed some or all of their customization files from other people [72]! Informal experts also tend to emerge, helping co-workers customize and allowing novice users to become proficient at settings manipulation [93].

Customization is inherently user-driven as individuals strive to "align device capabilities and appearance with their needs, desires and inherent behaviors" [16, 50]. It has become a bigger consideration for designers as technology use has expanded beyond the most expert users. Given the now worldwide availability of smartphones, customizing for differing priorities, user needs, capabilities, and cultures is critical [50, 95]. Park et. al. reported that individual mental models are affected by social factors, confirming substantial differences in modular user interfaces developed for varying cultures, ages, and genders [95].

Most recently, in 2012, Haberman looked at how users customized smartphones over time, specifically interaction modalities, interaction styles,

available content, and content presentation [50]. She reported high rates of customization, with all study participants making modifications to their device at the time of acquisition. Users primarily focused on the content and arrangement of their home screens, giving special treatment to prioritized applications. Participants did indicate a desire to move the location of the highest priority icons, as one-handed (primarily thumb-based) interaction was not well suited to the out-of-the-box configuration [50]. The author identified five types of connections between motivation and customization; the most important being that participants desired customizations motivated by their *abilities*, and they wanted customizations that closely aligned with current market offerings (familiarity).

Research into how and why users customize, while dating back to early personal computers, has progressed as technology has become a larger part of people's lives and user expertise improves. Haberman's insight that *all* users customize their smartphones out of the box implies a strong tendency for users to look for, and use, customizable options [50]. It also reaffirms the high rates of customization observed by Mackay in 1990 [73]. Mackay and Page [74, 93] suggested designing for satisfaction, rather than optimization, hinting at an interesting dynamic between performance and satisfaction. To test this relationship and enhance usability of the attitude aware controller, customizable control options were offered to users in the final phase of the usability experiment. Specifically, the custom options included settings for sensitivity and controller responsiveness, indicated important by users in related work by Pettitt [96].

Chapter 3

Application (Software) Development

3.1 Approach

The research questions and hypotheses introduced in Chapter 1 motivated development of a robot platform, custom controller, and human user experiment to assess usability of the proposed control system for tele-operation of ground robots. A significant amount of time was devoted to application development, as the attitude-based control envisioned required incremental development on a system and programming language with which the author had no previous experience. The goal was not simply to produce a prototype application, but to identify the design heuristics which had the greatest impact on usability. This increased the number of control iterations undertaken, as multiple pilot "experiments" were conducted to inform each phase of design.

Attitude aware controls use a device's accelerometers and gyroscopes, measuring linear motion and rotation, to monitor device pose and changes to it. The attitude-based application designed for robot control accepts attitude inputs resulting when the user moves the device like a steering wheel; these then map to robot behavior (throttle and heading). This chapter describes the details of that approach.

3.1.1 Controller Use Case

To best understand the application and its behavior, which comprises the bulk of this chapter, it is important to obtain a frame of reference regarding what the controller was designed to do. What follows is a casual use case, describing the goals, primary actors, and conditions for use, followed by a common scenario describing how the application might be employed [27, 28].

Goal

The goal of users engaged with this controller may vary slightly based on mission, depending on the type of reconnaissance and surveillance required. On the whole, users hope to guide a small, mobile robot equipped with a moveable camera into dangerous or hard to reach places in order to gain information about the environment. Successful operations would provide the user with the intelligence he/she requires while operating from a safe location some distance away.

Actor

The primary actor (user) is anticipated to be a young, male dismounted soldier between 18-35 years old. He may have some college education and grew up with technology–computers, video games, smartphones. His primary specialty in the Army is something besides robot operation, affording him limited time for formal hands-on tele-operation training (likely not exceeding 40 hours) [67, 79]. Some of these users might relish the opportunity to use "cool" technology, while others will feel it is an undue burden "not in their job description." Appendix A provides the text of two interviews conducted with Company Commanders deployed to Afghanistan using ground robots as a regular part of their tactical mission [67, 79],

both of which were used to help define **actor** and **conditions.**

Conditions

Realistic use conditions (uncertain field environments) are hard to describe; however, users are expected to be almost exclusively outdoors during robot operation. At best, the operator might be behind a wall or in a doorway. Lighting conditions may cause issues with screen glare (bright sunlight) or eye strain/fatigue (bright screen in the dark). The user will likely be stationary–standing, squatting, or lying down–with a team of fellow soldiers pulling security while he devotes his attention to robot operation. Security considerations may limit the duration of operations, so speed is crucial.

All soldiers (and therefore all users) wear military utility gloves and eye protection (both tinted and clear) for tactical operations. The robot may be one of many ruggedized, tracked vehicles under 35 pounds and small enough to fit in the soldier's rucksack. The smartphone controller is protected by a waterproof case and can be carried in one of the soldier's cargo pockets.

A note on gloved control. *Soft* buttons are touchscreen buttons, reacting only to conductive inputs like those provided by a user's fingertip flesh. To use them, gloved users must either remove their gloves or use touchscreen-compatible gloves which interweave special fibers into the fingertips. *Hard* buttons are tactile push buttons and can generally be felt even with gloves on. The iPhone has four tactile buttons: the power button, the home button, and the volume up and down buttons. In order to accommodate gloved users, attitude aware control implementations may use the volume up and down buttons, located on the top edge of the phone when positioned for control in landscape mode.

Sample Scenario

The user powers on both robot and controller, ensuring they are connected to the same WiFi network. The user then launches the *Custom Control v3* application (steering wheel icon) on the smartphone, which initiates the connection between controller and robot (see Figure 3.1). After several seconds, a live video feed loads to the device's screen, displaying what the robot can see from its onboard camera (Figure 3.2). A connection status indicator and battery icon also appear onscreen. To this point, the robot and camera have remained unmoving at their neutral position.

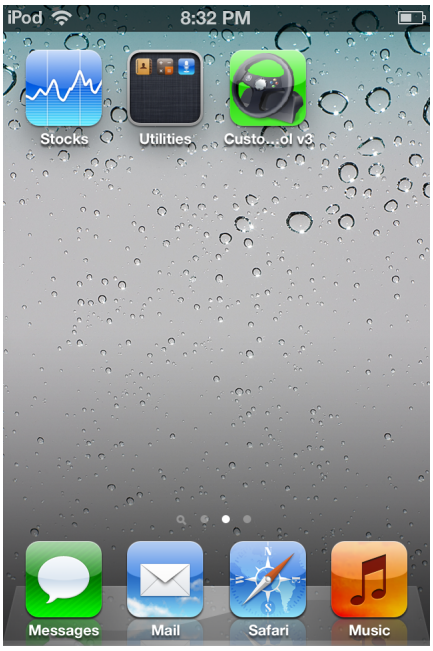


Figure 3.1: Application Icon (Green Steering Wheel) on Device Home Screen

To drive the robot, the user presses and holds a deadman switch (touchscreen or volume button) to activate the controls, then moves the device like a steering wheel. Tilting forward and back control the robot's speed, while rotation left and right control heading. When the user wants (or needs) to stop the robot, he

Figure 3.2: View from Camera Onboard Robot
Convention dictates that some portion of the robot chassis be visible in the camera's frame of view. Here the WiFi dongle and USB cord pictured are attached to the top front of the robot, similar in location to a vehicle's headlights.

simply removes his finger from the deadman switch and the robot stops almost immediately (unlike braking).

If the user then wants to scan the robot's surroundings, he would press and hold the camera deadman switch and again move the device like a steering wheel, this time to pan the camera left and right and tilt it up and down. To stop the camera at any position, the deadman switch is simply released. To capture a photograph from the robot's camera, the user taps a snapshot button, visible only when camera controls are active. When the picture saves to the device, an alert view pops up indicating success. The user hits OK to exit and return to robot control. Following completion of their operations, the robot is recovered and users exit the application by pressing the smartphone's home button. This terminates the connection, making it safe to power down both controller and robot.

3.1.2 Design Considerations

While this particular controller application was designed and developed for Apple smartphones, the *heuristics* identified during development may be applied to any handheld device using a similar set of attitude aware sensors. Some of these design guidelines are very specific to the motion algorithm, which will be discussed in Section 3.3.1. Others are more generic, regarding basic controller behavior, enumerated below.

1. *All device motion should be measured relative to the point where the deadman switch was activated.* This ensures that users do not have to expend mental effort finding the device "origin;" instead the device resets itself at each control activation.

2. Given that the bulk of control is done via gestural manipulation of the device body, and all robot manipulation is done via tele-presence, *the controller screen should be almost wholly devoted to video feedback, and therefore in landscape mode.*

3. *Driving and camera controls should not be activated simultaneously.* Users must choose whether to drive the robot or manipulate its camera at any given time. Enforcing this limitation means to simplify the control task for novice users and avoid disorientation in individuals new to tele-operation. This restriction could be customized, allowing users to enable simultaneous control in something like "expert" mode.

4. All feedback is provided to the user via video from the robot's camera, meaning *the camera should remain in a fixed (trimmed) position while driving.* This camera neutral position is the same for all users. Each

individual sees the same frame with respect to the robot to ensure best practices are maintained, e.g. keeping some portion of the robot chassis in view. This requirement could be relaxed for experienced users but is an important way to limit confounding effects during experimentation phases.

5. While originally designed to work in the same way that the driving controls do, camera controls also use a tilt/rotate steering motion to pan and tilt the camera head. However, enabled by improvements to the motion handling algorithm, *tilt-based camera inputs are actually best implemented when the user imagines that the robot's camera is attached to the controller itself.* Like taking a panoramic picture with the smartphone, users can sweep the controller broadly from left to right to pan the robot's camera and scan its surroundings. It's almost as if operators are using the control device to select a viewing frame of a broader window.

6. *The camera, when stopped, should remain in its current position.* This gives the user an opportunity to fix the camera in a given location for reconnaissance, closer examination of an object, or capturing a photograph.

7. When the camera is at a position other than neutral, and the user depresses the driving deadman switch, *the camera shall immediately snap back to its neutral position and remain there until once again manipulated.* This prevents issues with users not being able to resolve the camera's position from the robot's heading.

8. Given the limitations of a wireless network, periodic interruptions to the connection may occur. *If the controller disconnects from the robot, the robot should automatically stop.* A pop up then notifies the user of the

disconnection and prompts them to reconnect by hitting a button. Doing so re-establishes the connection with the robot and should permit the user to continue operation after only minimal disruption.

3.1.3 Apple and Objective-C

The Apple iPhone and iPod Touch® were selected as controller hardware for this project due to their well-supported development environment, XCode, and their standardized device characteristics: processors, operating system, and micro-electromechanical systems (MEMs). These devices run Apple's iOS, which is coded in Objective-C, an object-oriented programming language [31, 52]. The custom control application defined in this chapter uses this language, which may not be familiar to all readers. Below is a brief introduction to some of the Objective-C syntax used throughout the remainder of this chapter.

A *class* is a blueprint for objects, defining how instances of itself (created at runtime) should be treated [5, 7]. In object-oriented programming, a class usually represents a noun, such as a person, place, or thing, and is always defined with a capitalized name, e.g. `Class`. Classes' behaviors are defined by their *methods*, or subroutines. Two types of methods exist in Objective-C: class methods, denoted by `+(void)methodName`; and instance methods, denoted by `-(void)methodName`. Instance methods are the more common of the two, but require that an instance of their class be allocated before being called. Methods, and all other Objective-C *objects*, are defined with titles whose first letter is lower-case, followed by capital letters denoting each new word in the statement. Some methods require *parameters* be passed to them, e.g. variables required to calculate a function. This is done by placing a

colon after the method name, e.g. `-(float)methodName:(int)firstParameter second:(int)secondParameter`. The declaration in parentheses before each parameter and method identifies its type, such as integer or float; methods that do not return an object are void. `-(IBAction)methodName` is a special type of method which is accessed by a user action i.e. a touch or multi-touch event on an `IBOutlet` object, which are interfaces to the user (text boxes, buttons, sliders).

Comments in code are denoted by `//` or `/*`, depending on their length. In XCode, a special form of commenting using pragma marks is used to insert "bookmarks" which help better organize large programs. Titles preceded by `#` define such sections, e.g. `#Section Title`. Full, compilable versions of the controller code, including comments, can be accessed in Appendix D.

3.1.4 Model-View-Controller

Objective-C is an object-oriented programming language in which the model-view-controller (MVC) construct is commonly utilized. It emphasizes reusability of code and separation of tasks [5, 7]. As the goal of this project was to provide not only a prototype controller, but also a set of design guidelines for future use, designing for interoperability, expansion, and reuse was paramount. Figure 3.3 depicts the model-view-controller construct used in the attitude aware controller application. References in the graphic depict how the remainder of this chapter is organized. **View:** Section 3.2; **Controller:** Section 3.3; and **Model:** Section 3.4.

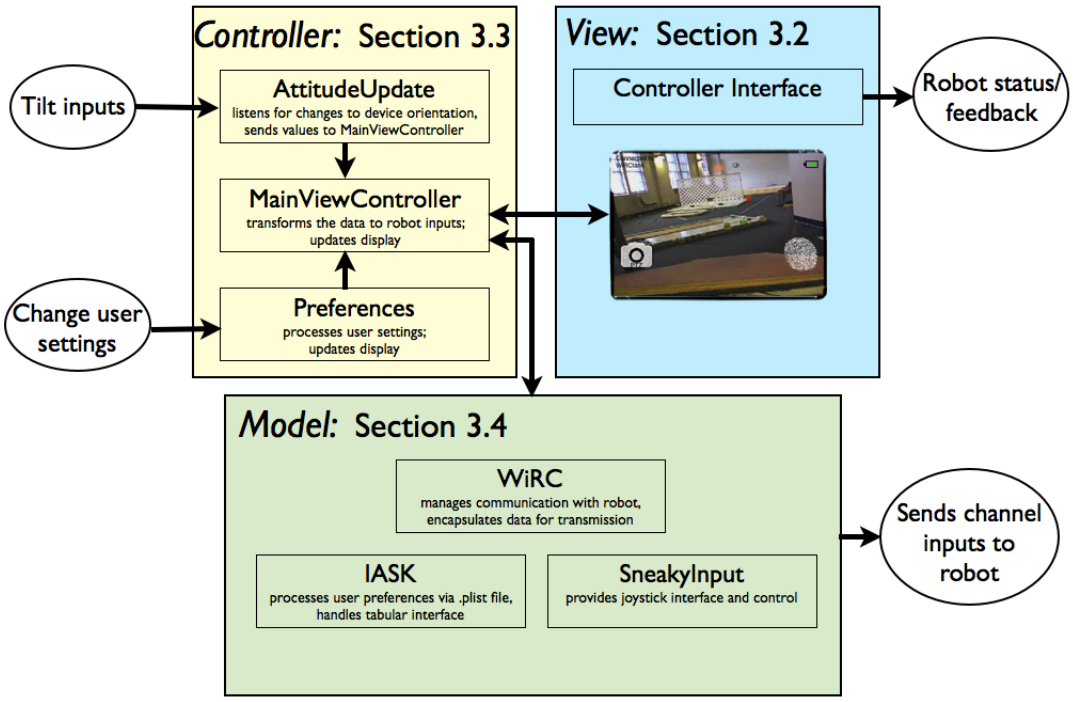
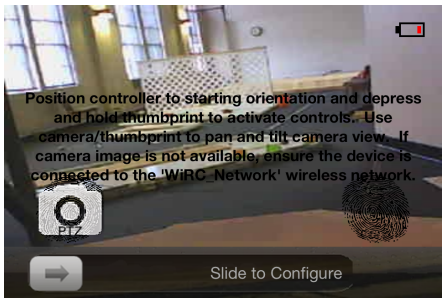


Figure 3.3: Custom Control Application Architecture

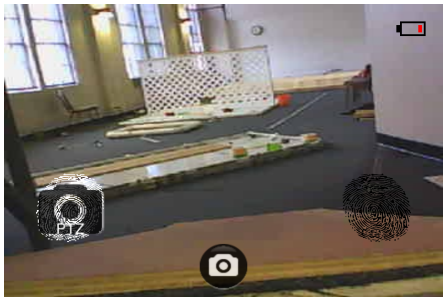
3.2 MVC: *View* (Controller Interface)

The graphical interface presented to the user was designed to be intuitive and accessible, regardless of technological proficiency or previous system experience. Low-level sensor inputs and controls were mostly hidden to the user and presented in forms easy to understand e.g. battery icon versus remaining voltage. The motion input portion of the controller required users to hold and move the device like a steering wheel, using a mental model most would find familiar. This control interface prioritized display of the robot's camera feed and limited the clutter of other forms of feedback, like connection, battery, and driving status.



(a) Controller Home Screen

(b) Driving with Controller



(c) Controlling Camera

Figure 3.4: Default Controller Application

Figure 3.4a shows the "home screen" of the attitude aware controller used in most phases of this study. The video feedback takes up the whole of the screen in landscape mode (320 x 480 pixels), with soft-button thumbprints, i.e. deadman

switches, overlaid on the left and right indicating where the user should press and hold to activate the attitude sensors for either the camera (left) or the vehicle motors (right). Along the top of the display (from left to right) is the controller connection status, the driving status, and the battery status. Driving status was comprised of three features: 1) a driving light which displayed green for driving, red for stopped, and blinking red for reverse; 2) driving text which read either "Driving" or "Reversing"; and 3) a visual representation of controller rotation, provided by keeping the video frame level to the horizon, similar to many aircraft attitude indicators (see Figure 3.4b).

A brief set of instructions displays on screen until controls are activated for the first time. The slider along the bottom of the screen allows access to the settings menu–a feature only made available in later phases of this research. Finally, the camera snapshot button in the bottom center of the screen allows users to capture a still photograph via the vehicle's onboard camera (see Figure 3.4c); these save automatically to the phone's camera roll. The application pictured is the *default*, used as the basis for all development. Details of modifications resulting in other versions of the controller will be presented in Chapter 4.

3.3 MVC: *Controller*

The **controller** portion of this application's MVC architecture is made up of three primary classes. `AttitudeUpdate` and `MainViewController` work together to manage the tilt-based aspects of the application (Section 3.3.1) and control mapping (Section 3.3.2). The third, `Preferences`, is summarized in Section 3.3.3, which describes the construct of user-customizable settings. Specific features of `MainViewController` not covered in the discussion of the motion algorithm or

user preferences are presented later in Section 3.5, which further details the class's architecture and flow.

3.3.1 Motion Algorithm

As arguably the most important part of this application's development, substantial time and effort was devoted to writing a motion algorithm capable of reliably handling complex three-dimensional motion. A survey of available tilt-based games and applications for the iPhone was conducted to collect and examine ideas and implementations which could inform this design (see Table 3.1). The majority of these games, built primarily for entertainment purposes, did not stand up to rigorous testing under use case conditions. Generally controls dictated a single user starting position, where a prescribed neutral point had to be found and held in 3D space. Almost none accounted for users who would be operating in unconventional positions e.g. lying down, and those that did accounted only for "inversion," a specific scenario where the device's screen is pointed down, towards gravity, and accelerometer values are therefore reversed. Apple provides no specific guidance for how to utilize accelerometer and gyroscope data, beyond documenting its properties in their definition of the Core Motion framework. Even fellow engineers have not completely solved the problem of how to orient and calibrate a system given the variance in user's mental models defining how they *expect* the device to behave. Pitman and Cummings designed a tilt-based controller for micro-air vehicles at the Massachusetts Institute of Technology; in recent experiments they observed users inadvertently flying the vehicle backwards, given their instinct to angle the device for comfortable viewing. This differed from the designed neutral

angle (level to the ground), causing the observed behavior [99, 98]. This problem was likewise noted in early versions of the attitude aware controller, motivating research to conquer this obstacle to intuitive control.

Attitude

The iDevices used in this research are equipped with 3-axis accelerometers and gyroscopes, which detect and measure motion and rotation about three axes, *x,* *y,* and *z*. Given their ability to identify motion, these devices are called **attitude aware**, meaning able to identify their orientation in three dimensional space at any given time, usually with respect to gravity. Primarily they contribute to the user experience by detecting orientation changes, but can be tailored for use in almost any instance where motion-based inputs are recorded e.g. pedometers, panoramic picture applications, and tilt-based games. Apple makes this possible through their Core Motion framework, encapsulating accelerometer, gyroscope, and magnetometer data in multiple forms. Developers can choose to access any of this sensor data independently, or as *attitude*, where sensor data are fused into a representative output. This output can be represented mathematically as Euler angles, a rotation matrix, or as a quaternion, which are properties of the `attitude` object [8]. Using a device motion manager, developers define the device reference frame, the type of attitude output requested, and the frequency with which such data should be captured.

Table 3.1: Survey of Tilt-Based Games and Products

Product	Developer	Key Features	Reference
AR. Drone	Parrot	A unique hobbyist quadcopter controlled via iPhone, and a development kit for more advanced controls and gaming is open source. The packaged controller recently updated to v2.0, which now supports both relative and absolute control modes. In the relative mode, the aircraft is the point of reference, and all tilts are translated into movement with respect to it. In absolute control mode, the pilot is the point of reference, and all tilts are translated into movement with respect to him/her. It is meant to provide a more intuitive interface for first-time pilots, or those who lack experience with remote control platforms. Absolute mode almost assures that users are co-located with the quadcopter, as true tele-operation would work best in the older, relative control mode. This controller does use attitude in two dimensions; however, does not account for controller inversion. It uses separate accelerometer and gyro streams rather than fused sensor data, leading to gimbal lock.	[130, 97]
WiFi RC	Dension	This application was developed to work out of the box with the WiRC hardware described in Section 4.3 for operating remote control vehicles. Its tilt mode requires the user to pick which gyro will control which channel (non-obvious in landscape mode). It then provides feedback regarding angle of input via small tick marks that move along the perimeter of the screen. By pressing the "play" button, users can re-calibrate the tilt control's center point; however, calibration does not correct for inverted use. There is no deadman switch, meaning controls are always active; presumably a user could stop by pressing the play button and re-setting the controller to zero. Gimbal lock occurs when the device is level to the ground.	[34]
Tilt to Live	Alex Okafor	In this game, the objective is for the user to maneuver his/her arrow around obstacles on the screen. The user interface provides options for orientation (choose from regular (45°), top-down (level with ground), or custom) as well as tilt sensitivity, adjusted separately for each axis. The custom orientation allows a user to set his/her own neutral point and does account for inverted reference frames. Users are prompted to choose an orientation before the start of each new game, and the interface is intuitive enough for even new users to understand, using graphics to demonstrate the choices.	[90]
Mad Bomber	Charles Scalesse	The author provides details of his motion algorithm in the reference provided. This game requires the user to tilt to move their character onscreen in an effort to catch dropping bombs. Uniquely, this game provides accelerometer calibration which supports any device play orientation, including inversion. This is accessed via an options menu which prompts the user to orient the device at the desired angle and then press a button to calibrate. Inversion is handled separately. Controls do not seem to be properly filtered to account for sensor drift, and only one-axis motion is measured, making the algorithm less complex.	[111]
Sphero	Orbotix	This application is used to control a small, robotic ball. The tilt-based controls (available in addition to joystick controls) are fairly elementary. They do not account for user-defined origins, meaning users must start with the device parallel to the ground. It likewise does not operate correctly inverted. There is no deadman switch operation, making it much too easy to lose control.	[92]
Cube Runner	Andy Qua	This game, similar to Tilt to Live, has users represented as small flying arrows through a three-dimensional landscape filled with cubes they must navigate around. It is presented in portrait or landscape mode (the only one surveyed) and tilting OR rotating left and right steers the aircraft. Tilt does not control animation speed. In the settings, users can calibrate the center point from which movements are measured along the single axis utilized. In testing, calibration successfully overcame starting positions where the device was purposefully off-center by up to 30°. Additionally, it automatically detects cases of inversion, even during active game play, and makes behind-the-scenes adjustments to ensure controls are not reversed; however, it does not account for users who wish to leave the device parallel to the ground i.e. resting on a table, as gimbal lock prevents measurable *rotation* at this attitude.	[102]

Reference Frame

In the Core Motion framework, the motion manager requires a reference frame be set when it is activated. Apple's SDK offers four reference frames. z is vertical in each, and developers can choose between an x-axis that is arbitrary, arbitrary corrected, magnetic north, or true north; the magnetometer is required for all but the first [8]. Since z always points to gravity, it is positive coming up from the device screen and negative through the back of the device (see Figure 3.5) [9].

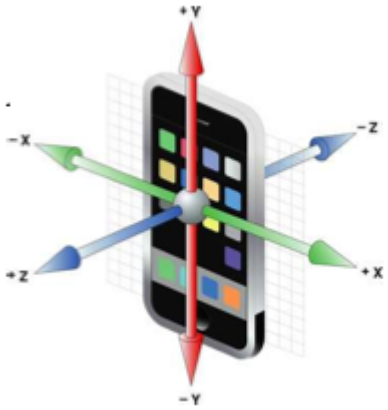


Figure 3.5: iPhone Coordinate System

The reference frame is important to attitude aware controls given that orientation is measured in that coordinate system. A fixed, global reference frame would mean that the origin exists at a very specific point in space, a difficult concept on a device which can move freely in three dimensions. Therefore, attitude aware controls are best implemented with a relative reference frame which defines an origin based on user input and device pose and measures all further device motion with respect to it. In other words, this approach provides for self-zeroing and instantaneous reference frames when attitude controls are activated.

In the attitude aware controller, this is achieved by declaring a new reference frame each time the deadman switch, which activates the motion manager, is touched. The following is executed with each `-(IBAction)` on the deadman switch button:

```
CMAttitude *referenceAttitude =  motionManager.deviceMotion.attitude;
self.referenceFrame = referenceAttitude;
```

Users can be holding the device in any position when setting the reference frame, and at the instant that the deadman switch is activated, the phone is at its origin (0, 0, 0). Not only does this improve usability, it also, in conjunction with quaternion attitude measurements, allows users to initiate the motion manager irrespective of gravity. Previous implementations found it difficult for users operating the device in unconventional positions, such as lying down, where the z-axis is reversed compared to Figure 3.5. One game, Tilt to Live, attempts to control for this by offering the user a choice of starting positions [90]; this takes that a step further by identifying the starting position and accounting for its orientation *without* explicit user input.

Quaternion Based Approach

As alluded to in the discussion of reference frames, quaternions play a large part in the adaptability and precision of the motion handling within the attitude aware application. Apple provides brief documentation on their use as a `CMAttitude` representation, but they are governed by complex mathematics that many developers choose not to explore. Two of the better know "robotics" implementations–*AR.Drone* and *Sphero*–use mostly Euler representations and/or separate accelerometer and gyroscope data in their publicly available SDKs

[97, 92]. Several applications use quaternion representation from a publicly available math class (Quaternion.h) [90], but none have documented the use of quaternions as direct output of Apple's Core Motion framework. It appears, based on the limited literature available, that quaternions in iOS programming are used more for 3D animation e.g. spherical linear interpolation, than capturing device motion.

Quaternions provide computational benefits, storing four numbers instead of the nine needed for rotation matrices. Quaternions also avoid gimbal lock–a common downfall of Euler angles–which occurs when one degree of freedom is lost due to two axes being driven into parallel configurations [112]. On the iPhone, specifically, this occurs most often when the the pitch value (rotation about the axis running along the iPhone's width) is lost in landscape orientations. Overall, quaternions can greatly simplify motion algorithms while addressing common issues regarding reference frames and gimbal lock. Their mathematical properties make them both a valuable and under-utilized tool for attitude aware control.

Developed by W.R. Hamilton in the eighteenth century, quaternions were originally devised as a fourth-dimensional extension of complex numbers (Eqn. 3.1) [17]. Most commonly, quaternions are normalized to a magnitude = 1, as Apple's Core Motion framework does, yielding what is known as the *unit quaternion* [112]. In this representation, the four dimensions can be presented as a rotation of θ radians (scalar) about a unit vector $\{x, y, z\}$ [8], decomposed for better understanding in Eqn. 3.2.

$$w + xi + yj + zk \tag{3.1}$$

where $i^2 = j^2 = k^2 = ijk = -1$ with real w, x, y, z

represented as $\{w, \vec{v}\}$

where $\vec{v} = \{x, y, z\}$

$\{q.x, q.y, q.z, q.w\}$ where q is the unit quaternion

$$\begin{aligned} q.x &= x \times \sin\left(\frac{\theta}{2}\right) \\ q.y &= y \times \sin\left(\frac{\theta}{2}\right) \\ q.z &= z \times \sin\left(\frac{\theta}{2}\right) \\ q.w &= \cos\left(\frac{\theta}{2}\right) \end{aligned} \tag{3.2}$$

For the purposes of attitude aware control mapping, quaternion motion, when decomposed to its individual *x, y,* and *z* components, must be done so as not to lose the fourth dimension. This is accomplished quite easily by calculating **relative** motion, multiplying the inverse of the reference quaternion by the quaternion describing current attitude.

```
//Capture quaternion of reference frame
    neutralQuat = referenceFrame.quaternion;
```

Take the inverse of `neutralQuat`:

$\texttt{neutralQuat}^{-1} = newNeutral\ (q_n) = \{-w, x, y, z\}$

```
//Capture quaternion of current orientation (at each time step)
    currentQuat = currentAttitude.quaternion;
```

Save `currentQuat`:

`currentQuat` = currentAttitude (q_c) = $\{w, x, y, z\}$

Quaternion multiplication is a special operation where matrices are associative but not commutative i.e. order matters [39]. When $q_n \times q_c$, Eqn. 3.3 results.

$$\begin{aligned} w &= (q_n.w \times q_c.w - q_n.x \times q_c.x - q_n.y \times q_c.y - q_n.z \times q_c.z) \\ x &= (q_n.w \times q_c.x + q_n.x \times q_c.w + q_n.y \times q_c.z - q_n.z \times q_c.y) \\ y &= (q_n.w \times q_c.y - q_n.x \times q_c.z + q_n.y \times q_c.w + q_n.z \times q_c.x) \\ z &= (q_n.w \times q_c.z + q_n.x \times q_c.y - q_n.y \times q_c.x + q_n.z \times q_c.w) \end{aligned} \tag{3.3}$$

After returning the values (between 0 and 1) for *x, y,* and *z*, sensor data is filtered to account for gyroscopic drift and a transform function is applied to convert those values to appropriate control inputs (pulse width modulation, in μs).

Pulse Width Modulation: Pulse width modulation (PWM) is a common technique in robotics, where microprocessors digitally encode analog signals. Their measure, in μs, is the duration of the digital signal's *on* time. By modulating (changing) this "on time," the analog output value is varied [13]. The microprocessor used in this study accepted pulse width modulation values between 800 - 2200 μs.

Complementary Filter

Before sensor data is ready for transform to pulse width modulation, it must be filtered to account for (primarily) gyroscopic drift. A study by Nymoen et. al. found that the iPod Touch used in their testing

> performs quite well when it comes to *roll* and *pitch*, with superior drift performance, and equal noise level, but the *yaw* measurements from the iPod are less accurate and less precise. The average yaw drift of a still recording is 70.6×10^{-5} °/s which is equivalent to a drift of 2.5°/h. An additional effort to force the device to give inaccurate yaw data by shaking the device violently for 23 s, resulted in a yaw drift of 11.5°.

In the attitude aware controller, the yaw axis of rotation is mapped to steering; therefore, the implication of drift from this research cannot be ignored. Pilot testing indicated that drift did indeed worsen as a user turned the device or maintained dynamic motion. This error manifests itself by making it appear as if the user is inputting a gradual turn, when in fact the device is stationary. Obviously this is not ideal given that inputs are tied directly to robot motion. Since the deadman switch is designed to re-level the device each time it is activated, it also serves to "re-calibrate" sensors that have drifted off; that said, users may not understand that phenomenon, nor should they have to. Therefore, a complementary filter was applied to ensure sensor data was filtered to prevent large-scale drifts in the z-direction [30]. Done partially through trial and error, appropriate constants were determined and tested to eliminate most of the sensor drift on both development devices when in landscape mode (Eqn. 3.4).

$$filterZ = 0.65 \times z + 0.35 \times oldZ \qquad (3.4)$$

where z=current sensor reading and $oldZ$=previous sensor reading.

This filter does a satisfactory job of managing drift that is intrinsic to the device sensors, as well as that due to shaking, because it provides a weighted balance of both current and previous readings. It is unknown whether this filter would be directly applicable, in its current form, to smartphones by other manufacturers; however, it did prove adequate on both the iPhone 4 and iPod Touch used in this study.

3.3.2 Transform Functions

Following motion processing, which is handled primarily by `AttitudeUpdate`, $filterZ$ is pushed to a method in `MainViewController` that converts the raw sensor data (encapsulated as a quaternion) into pulse width modulation, measured in microseconds. Using raw y data and filtered z data, the function `-(float)convertQuattoPWM:(int)vehicleType` first determines the *centerDistConstant*, which is the value added to each channel's "trim." It is common in robotics and remote control applications to refer to a vehicle's *trim*. This is the input at which a motor is not moving, also known as neutral. Every speed controller is slightly different, so this is not a fixed number, rather a point from which other measurements/changes are generally taken.

The value c, in Eqn. 3.5, defaults to 700, derived from a default channel trim value of 1500 μs and the robot accepted input range of 800-2200 μs. *centerDistConstant* is related to the user-selected *responsivenessConstant* by Eqn. 3.5. This equation is evaluated twice, once for the y-direction, and once for the z-direction.

$$centerDistConstant = \frac{c}{responsivenessConstant} \tag{3.5}$$

responsivenessConstant is a user setting which determines the device range of motion. If a user prefers large movements to yield smaller results, this value would be closer to 0.60 (the programmatically established upper limit), equal to approximately 60% of the device's range of motion along a given axis. If a user prefers small movements to yield larger results, the value would be closer to 0.10 (the programmatically established lower limit), equal to approximately 10% of the device's range of motion along a given axis. The transform function includes this value as a scaling factor (Eqn. 3.5), and also as a limit.

```
if (filtZ > responsivenessTurn) {
    filtZ = responsivenessTurn;
    }
if (filtZ < -responsivenessTurn) {
    filtZ = -responsivenessTurn;
    }
```

The next set of transforms is dependent on the input parameter `vehicleType`. Built to permit cross-platform control, the attitude aware controller was tested on two vehicle platforms–a tracked Kyosho Blizzard described in Section 4.3 and an HPI Racing 4WD Rock Crawler. For vehicles, like automobiles, who use Ackerman steering to avoid tire slip [66], the transform consists of calculating a scaled "value from center" that is then added to the trim values for each channel (see Eqn. 3.6). Here $yDistPWM$ becomes the value from center. The constant previously derived, based on the responsiveness constant, is multiplied by the current y value, which is negative due to the particular behavior of the 4WD

vehicle used during testing. Based on directionality of the speed controller, this value may remain positive. Finally, that value is multiplied by the *accelScale*, another user-based preference that scales the throttle by between 10% and 80%. The default acceleration is scaled by 40%, which in pilot testing performed well for indoor use. The values $centerPWM$ and $carSteerTrim$ are global constants dependent on the specific vehicle platform. The 4WD Rock Crawler, for instance, had a throttle trim = 1500 μs, but a steering trim = 1570 μs.

$$\begin{aligned} yDistPWM &= centerDistConstantY \times -(y) \times accelScale; \\ zDistPWM &= centerDistConstantZ \times -(filtZ) \\ yPWM &= centerPWM + yDistPWM \\ zPWM &= carSteerTrim + zDistPWM \end{aligned} \tag{3.6}$$

Alternatively, if the vehicle is tank-like where right and left tracks are channels, and steering and throttle are controlled by a combination of their inputs, the transform functions become more complex. Here, a broader understanding of control mapping is necessary. In general, a tracked vehicle is capable of zero-point turns, where the left and right tracks rotate opposite one another at the same speed. Additionally, gradual turns are executed by scaling the outer track to a higher speed than the inner track. Driving straight is handled by applying the same throttle command to each of the tracks, either forward or reverse. The transform functions seen in Eqn. 3.7 demonstrate these scenarios; however, a zero-point turn does not exist. Given the speed controller's limitations with reverse, which will be described in more detail later in this chapter, stationary turns were done by applying throttle to only one track, while keeping

the other stationary. Deadspace was programmed near the controller's neutral point such that movement would only register outside of that zone; for throttle (forward/backward) it was set to ± 20 μs, and for steering (rotation) ± 60 μs.

$$yDistPWM = centerDistConstantY \times -(y) \times accelScale$$
$$zDistPWM = centerDistConstantZ \times -(filtZ) \times accelScale$$

Turning left gradually:

$$rightTrack = tankTrimR + yDistPWM$$
$$leftTrack = tankTrimL - (zDistPWM \times 2)$$

Turning left in place:

$$rightTrack = tankTrimR \tag{3.7}$$
$$leftTrack = tankTrimL - (zDistPWM \times 2)$$

Equations for left and right tracks are reversed for right turns.

Driving forward:

$$rightTrack = tankTrimR + yDistPWM$$
$$leftTrack = tankTrimL + yDistPWM$$

$rightTrack$ and $leftTrack$ are variables saved and later sent to the robot, while $tankTrimL$ and $tankTrimR$ are the global constants for the Kyosho Blizzard (both equal to 1550 μs).

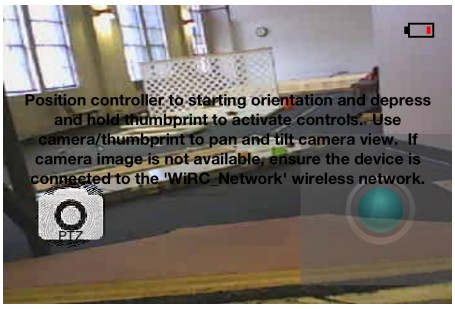
Finally, the last channels to consider are the two camera servos controlling pan and tilt. Their transform functions are equivalent to Eqn. 3.6, except with a positive $filtZ$ variable. Notably, the trim positions on the camera servos were not near 1500 μs, rather tilt = 1200 μs and pan = 1300 μs. All trim values were determined experimentally and hard-coded as global constants. The program was written to be generic enough to account for any trim value, although there is no interface currently available for the user to set these values him/herself.

3.3.3 User Settings/Preferences

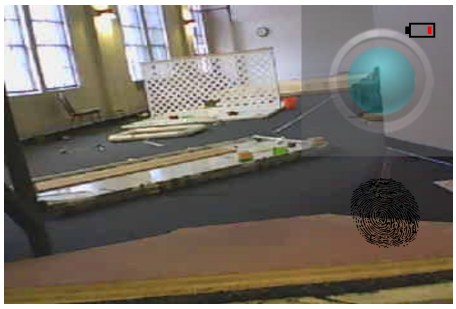
Another key component of the application's **controller** architecture was the `Preferences` class, written to process all user settings. The application was designed to accommodate a number of customizable options, in answer to one of the research questions inspiring this project: "to what effect would customization impact controller usability and user satisfaction?" First, users were able to choose their control **mode** for both driving and camera channels. If they chose tilt-based inputs, a soft button (thumbprint) appeared in the bottom right of the screen; however, by altering **control location**, this button could be moved to any one of six positions around the perimeter of the device. A hard button deadman switch (volume button) also existed in the prototype, intended primarily for gloved operation.

When the joystick control mode was selected, users again had a choice of control location, in addition to controlling the **size of that joystick** by scaling from 75% to 150% of the default joystick size. Figure 3.6 shows several variations of the controller customized with various layouts and control modes. Note that when both channels are joystick operated, only one joystick is presented onscreen;

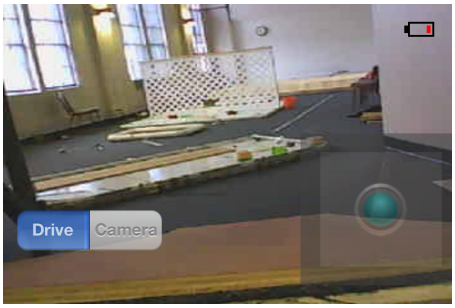
this eliminates the option of simultaneous control while managing processor requirements given the difficulty of animation rendering alongside streaming video.



(a) Small Joystick with Camera Tilt



(b) Large Joystick, Modified Position



(c) Dual Joystick Mode
(One Joystick+Channel Selector)

(d) Hard Button Option
(White Light Indicates Button Press)

Figure 3.6: Custom Controller Application Examples

The final three options related to control scaling; the first being **acceleration sensitivity**. This value, between 0.1 and 0.8, acts as a direct, linear scale factor for the vehicle's throttle. On the tank robot, especially, indoor driving was nearly impossible at the highest speeds attainable. Pilot experiments indicated 0.4 as a suitable default, allowing the vehicle to traverse inclines without sacrificing maneuverability in tight spaces. The other two options, **degree of tilt**, are a representative measure of controller responsiveness. Some users felt that the controller had to be moved too much in order to initiate robot motion; others felt that they achieved full throttle too quickly, with not enough control over

intermediate values. Degree of tilt addresses those problems by providing a responsiveness factor. Between 0.1 and 0.6 with a default of 0.45, these values represent roughly the degree of tilt/rotation available when either driving or turning. By lowering this number, users are limiting the overall range of motion of the controller and therefore making it more responsive. Likewise, when at its maximum, range of motion is nearly 90°, making the controller less responsive but perhaps more precise, as users feel better in control of all inputs between neutral and maximum. These settable values and their defaults are summarized in Table 3.2.

Table 3.2: User-Settable Values

	Default	**1**	**2**	**3**	**4**	**5**	**6**
Channel Mode	1	Tilt	Joystick	-	-	-	-
Button Type	1	Soft Button	Hard Button	-	-	-	-
Control Location	3	Bottom left	Bottom center	Bottom right	Top right	Top center	Top left
	Default	**Min**	**Max**				
Acceleration Sensitivity	0.4	0.1	0.8				
° of Tilt: Straight	0.45	0.1	0.6				
° of Tilt: Turning	0.45	0.1	0.6				
Joystick Size/Sensitivity	1.0	0.75	1.5				

Using an open source project called In App Settings Kit (IASK), customizable options were created programmatically in a .plist file which uses an ordered structure and "specifier keys" to identify cells (in a tabular view) and their attributes. The values of these cells were updated at program launch and anytime the settings view was dismissed, indicating possible changes had been made. Two custom buttons in the settings view permitted expanded functions like **resetting controller defaults** and **saving options to file**. This, in particular, existed to support experiment data collection, archiving each user's final configuration file for analysis. The entire settings view is shown in Figure 3.7. Using `saveContext` in the `AppDelegate`, all settings options were saved and preserved regardless of

how/when the controller program was killed. This means that once a user has established a controller to his/her liking, it will remain that way between uses, when the device's power is cycled, and during application crashes, until manually restored to defaults.

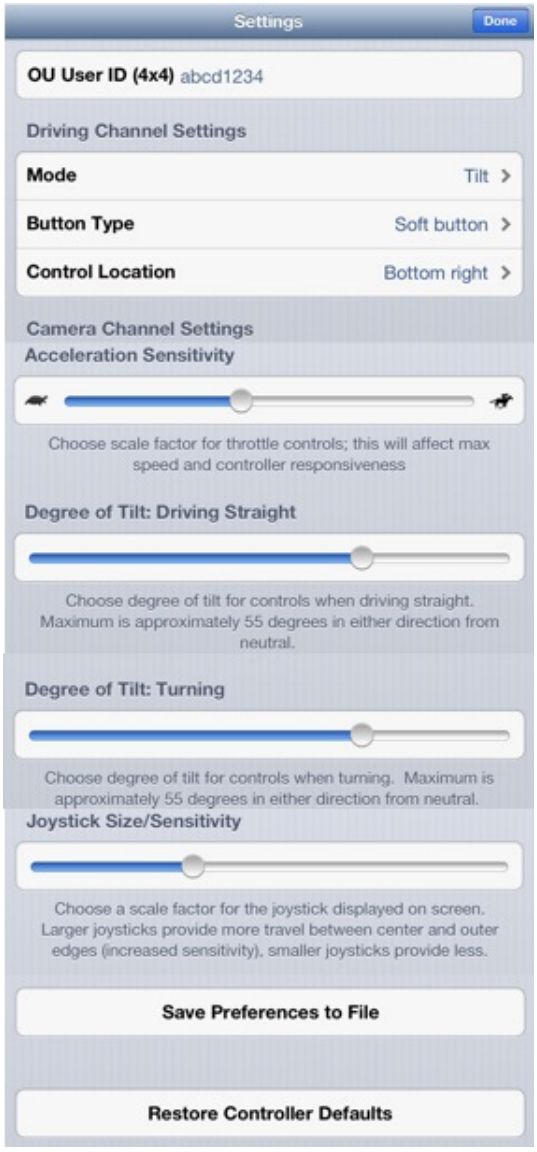


Figure 3.7: Settings View

3.4 MVC: *Model*

The model is generally the most complex portion of any MVC application, as it houses the bulk of the program's brains. A brief description of each model class is provided in the list below. For those with little or no programming background, this concludes the introduction and description of the application's execution and structure. Programmers and engineers with a specific interest in program syntax and organization may find Sections 3.4.1 and 3.5 useful.

1. *WiRC:* This SDK was provided by the manufacturer of the onboard processor, a Dension WiRC, used to connect and control the robot via iPhone; their research and development engineer, Balint Viragh, provided it under limited distribution [126]. It contains all of the classes comprising the WiRC communication protocols, which are also outlined in their user guides [35, 36]. This comprises the whole of the controller's back-end, and encapsulates data for transfer to the on-board WiRC processor which then distributes it to the appropriate robot channels. Several interface functions make this possible and will be discussed in Section 3.4.1.

2. *IASK:* IASK stands for In App Settings Kit; it is an open source project which uses Apple-supported .plist files to alter application settings within the application, rather than through the global settings menu on the iPhone [125]. It relies primarily on the standard Apple settings cells e.g. sliders, switches, and text fields; however, it is expandable to include custom cells and buttons. This program was used to manage all of the user-controlled customization, introduced in Section 3.3.3.

3. *SneakyInput:* SneakyInput is a modified version of the open source package SneakyJoystick, a gaming interface which includes joysticks, d-pads, and buttons [94]. It is an elegant solution using cocos2d libraries to support animation. It was employed in this project to facilitate joystick modes in combination with the tilt controls already discussed.

3.4.1 Communication Protocol & WiRC SDK

While the WiRC SDK is made up of more than 15 classes, the primary interface between `MainViewController` and the WiRC device is defined in WiRC.m. WiRC.m opens the connection between the two devices and provides the send/receive pipe necessary to handle all of the WiRC's major functions. The other classes focus on the construct of specific messages, as well as error handling. That said, only a few interface functions are necessary to operate the WiRC from within the `MainViewController`, which is declared as a delegate of the `WiRC` class. The first of these is a method named `startConnectInNewThread`. It does exactly what its title implies, by broadcasting the `WiRCDiscover:` command with parameters for versioning and timeouts. This polls for available WiRC devices on the network and returns them by index number and IP address; then, the function connects to index 0 by calling WiRC's `connectAndLogin`. Finally, `startConnectInNewThread` completes communication setup by establishing the camera feed, using the interface function `startCameraWithID:`.

Upon connection and setup, the WiRC is then looking for periodic channel data (PCD) messages, handled via the `setChannel:` method. This method requires a parameter for `channelValue` and `withValue`, which provides a channel number (1-8) along with the PWM value of that channel (800-2200 μs). Aside from their call in `MainViewController`, these WiRC methods are only accessed

from one other class, `AppDelegate`. There they are called to handle application termination and moves to/from the background. This is done by calling the WiRC delegate, `wirc`, on an instance of the `mainViewController`. The only new method presented here is `disconnect`, which terminates the connection thread.

```
//When application resumes in foreground
if (mainViewController.wirc.isConnected == NO) {
   [self.mainViewController startConnectInNewThread];
   }
//When application will terminate
   [self.mainViewController.wirc disconnect];
```

3.5 MainViewController

As the heart of the application, and the class from which the main program thread is managed, `MainViewController` is the most complex custom class in this application. Aside from interfacing with the WiRC communication functions, it also loads all of the interface views and handles the control mapping. The mathematics and motion algorithm have already been discussed, so this section will focus primarily on items of interest to fellow Objective-C programmers: unique method definitions and program flow. Given that the application is built to support joystick modes, which are facilitated via `SneakyInput`, the program uses a combination of UIViews and CCLayers (cocos2d); UIViews are controlled by ViewControllers, while CCLayers fall under a CCDirector object. In the `AppDelegate`, this is handled by creating an instance of the CCDirector and adding the `mainViewController` object to the window as the `rootViewController`. This ensures that, upon loading, the program displays the `MainViewController` class, which then takes over managing the application.

Upon loading the view, `MainViewController` adds instances of variables needed

to manage the controller, as well as loading and hiding UIButtons and UIViews where appropriate. Many local functions are called first in `viewDidLoad` and again at appropriate points during runtime, when specific changes are made. To aid in debugging and reuse, the `MainViewController` is separated into several pragma marked categories, each described in turn. Appendix D should be referenced as a supplement to this section.

#Setup This section consists of the method `viewDidLoad` and `setupAnimationView`, as well as a function that initiates the `appSettingsViewController` and ties it to the appropriate nib file. `viewDidLoad` primarily sets up instances of objects and classes, including `AttitudeUpdate` and `Preferences`. When `AttitudeUpdate` is allocated, the `startMotion:` method is called to begin the Core Motion motion manager to make available updates from the accelerometers and gyroscopes. In addition, it starts the WiRC connection in a new processor thread, by calling `startConnectInNewThread` on `detachNewThreadSelector:`. `setupAnimationView` loads a CCGLView named `cocosView` and attaches the `director` object to it. This is the view which handles the joystick sprites, called by `runWithScene:[HelloWorldLayer scene:]`.

#Volume Functions One of the user-controlled settings includes an option to use the volume hard buttons as deadman switches in lieu of the soft button thumbprints. To enable this, the application must register for volume updates by calling `establishVolumeListener`:

```
[[NSNotificationCenter defaultCenter]
                        addObserver:self
                        selector:@selector(volumeChanged:)
                        name:@"AVSystemController_
                               SystemVolumeDidChangeNotification"
                        object:nil];
```

When updates are registered, they are compared against the previous volume value to determine whether the volume up or down button was used as the control input. That information then informs which robot channel is affected.

#Camera View & Functions The methods in this section exist to receive the camera feed and display it within the appropriate UIView. Additionally, a UIImageObject called `lastSavedCameraShot` caches each previous frame for use with the camera snapshot feature. The complexities of the image handling algorithm are hidden behind the `didReceivedCameraFrameNum:` method, which processes individual frames of the camera's MJPEG stream. An `-(IBAction)screenCapture:` method was written to save the `lastSavedCameraShot` to the camera roll via `UIImageWriteToSavedPhotosAlbum`. An error handling method was used to either confirm that the picture was saved correctly or identify to the user that it was not. These alerts were pushed as UIAlertView objects, or the semi-transparent blue pop up boxes familiar to most iPhone users (see Figure 3.8).

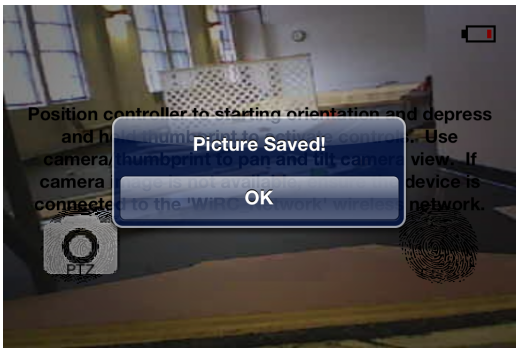


Figure 3.8: Alert View Pop-up

#Connections Aside from the specific connection methods discussed in Section 3.4.1, this section also includes methods to display connection status. Specifically, `onConnect:` provides the user with feedback in two cases: 1) if the WiRC is *not* connected, an alert view informs the user that they have been disconnected and provides a button to re-connect; 2) if the WiRC *is* connected, the camera feed is displayed and information regarding network name and status is pulled from

`fetchSSIDInfo`.

#Motion Fittingly, the motion section is the most robust. This is primarily due to the number of functions needed to handle several types of control inputs from several different sources. This includes methods which receive motion and joystick data, methods which transform that data, and methods that transmit that data. Almost all of those rely on `vehicleType` as an input parameter, as the application currently supports two vehicle profiles, stored as integers (1 = car and 2 = tank). The algorithm remains the same, but variable signs (+/-), speed controller limits, and motor deadspace are very much dependent on the individual robotic platform and could be expected to change even between two vehicles of the same profile type. Most of these differences (e.g. trim) are saved as global constants, which would allow creation of vehicle libraries to catalog these values. The first method called in the motion section is `setDefaultTrims`, which uses these global constants to ensure that vehicles are set to their neutral (trim) position on all channels when operation commences.

When tilt-based controls are used for either driving or camera control, `AttitudeUpdate` runs in the background awaiting instruction from the `MainViewController`. `activateAccelerometers:` provides this instruction, called when either deadman switch is depressed. A series of conditionals determines how to handle the input dependent on the sender. Ultimately, driving inputs cause the driving light and status to appear on the control screen and camera inputs hide the driving lights and status but display the snapshot tool. The end result in both instances is a call to the `startMotion` method. `startMotion` is a local interface to `[AttitudeUpdate startMotionLoop]`. In that method the device reference frame is saved and motionLoop timer is established to pull data via `motionUpdates`. After filtering the data and converting it to relative quaternion motion, the class eventually returns a value for w, x, y, and z. These are read in to `calculateUpdates` (also on a timer loop) and handled according to which switch initiated the action–driving or

camera. For both, data is transformed via `convertQuattoPWM` before being transmitted to the device. The z-input is also used in calculating a `cameraTurn` variable, which rotates the UIView to help the user visualize degree of rotation by keeping the camera frame level to the horizon.

If joystick-based controls are present, either in combination with or exclusive of tilt-based inputs, a `calculateLoop` timer is initiated to poll for updates in the method `readInJoystick`. This method first creates a CCNode to attach to the joystick sprite in the scene running in the `cocosView`. The values returned by this node are dependent on the size of the joystick (in pixels) and are scaled accordingly in the method which transforms them to PWM (`convertJoytoPWM:`). Like with tilt-control, joystick data comes from either a driving or camera input, and that tag, or sender, informs how the data is processed by the remainder of the program.

After transform to PWM, all control inputs, regardless of mode, call the `checkDirection` method. It is used to identify the vehicle's intended direction of travel such that driving lights and statuses can be updated accordingly. A solid green light denotes forward motion (driving or turning), and a red light illustrates that the attitude controls are active but within the neutral zone i.e. the robot is stopped; a blinking red light is used to illustrate reverse. Reverse must be handled as a special driving case due to the limitations of both robot's speed controllers. Each requires a "trigger" to engage reverse; on the radio frequency (RF) transmitters, this is done by either pausing in neutral or inputting full reverse, depending on the vehicle platform. Programmatically this is accomplished using false signals sent to the robot over 50 timer cycles (1 sec). Regardless of user input, a false signal of **trim** is first transmitted to shift the car to reverse. For the tank, the false signal starts at **trim**, then sends **full reverse**. Initial tests proved these programmatic lags were successful, although unreliable. Executing on the tracked vehicle was especially difficult given that both track channels engaged reverse independent of the other. Therefore, during human

user trials, reverse functionality was simplified, permitting users to reverse only straight back at a set speed.

The final step to transmit values to the robot is executed in `sendValuesToDevice:(int)vehicleType withValues:(int)pwm(int)pwm2`. It processes the PWM variables and distributes them to the correct channels. Channel 1 is steering (car) or right track (tank), and channel 2 is throttle or left track. Channels 3 and 4 control camera pan and tilt on both robots.

Just as important as processing control inputs, is stopping those inputs when necessary! The `-(IBAction)stopAccelerometers:` is triggered when the deadman switch is released, and in turn calls `stopAttitudeMotion` which sends the trim commands to the robot and resets the controller display. Stopping the joystick inputs is handled separately, as it is controlled on its own timer; however, it serves the same purpose. Multiple lines exist within these methods to ensure all variables, commands, and timers are properly reset, invalidated, or restarted in an effort to ensure the robot does not move when not commanded to do so.

#Messages This section is comprised primarily of methods packaged with the WiRC SDK. These include error handling and connection initiation. `didReceivedPSD:` is used to process periodic status updates from the robot. Currently, this only includes battery charge, measured in millivolts. To simplify its presentation to the user, these readings were displayed graphically as a battery icon; red indicates insufficient power, less than 5500 mV. The other message method is called `provideInstructions`, which pushes the appropriate text to the UIView upon application launch. For distribution these instructions would likely be replaced with one or a series of splash screens to more dynamically illustrate the intended controller functions.

#IASK and Settings Slider These sections are closely related, as one handles the display of the slider and one handles the actions which occur when the slider reaches its maximum value. The slider was designed to mimic the iPhone's lock screen, where

sliding to unlock prevents accidental access to certain features. This settings slider is displayed for five seconds at a time, either at application launch or when the center of the display is double-tapped. This ensures that the slider does not remain on the screen for longer than necessary, interfering with the buttons along the bottom of the display. When the slider is slid fully to the right, `showSettings` is called to load the `appSettingsViewController`, introduced in Section 3.3.3.

 # Check Settings The final section was written to interface with the custom class, `Preferences`, which contains a class method for each of the user-controlled custom variables. This facilitates use of these variables within the `MainViewController`, where they are used to tag the mode of operation and provide scaling factors.

Chapter 4

Experiment Design

4.1 Research Questions

Looking back at research question R.1, the core of this research is to identify and assess operator control unit methodologies and modes that promote simple, efficient control of small reconnaissance robots using smartphones, which provide a powerful, lightweight, expandable platform for control. If specific usability issues are addressed, these devices have the potential to replace any number of different controllers currently in use. The best way to test this assertion, and the controller application prototype, is via formal usability assessments with human users; ideally these trials are conducted in environments similar to those in which the controller will be deployed.

The multi-phase experiment defined in this chapter was designed with this aim in mind. Twenty-five participants were recruited to help answer the question "Can smartphones be implemented in tele-operated control of ground robots such that their advantages (performance, usability, size/weight) overcome suspected deficiencies, including negative user perception?" Several hypotheses were were developed and tested using the experiment designed herein to collect data to either support or refute them. Details of hypotheses are presented with results in Chapters 5, 6, and 7 and were previously introduced in Chapter 1.

4.2 Related Work

The Army Research Laboratory (ARL) has done significant work with smartphone-based controllers, conducting a series of scalability experiments started in 2008. Redden summarized the results of the complete series [106], while recent work by Pettitt et. al. [96, 43] is most relevant to the usability experiment proposed here. In Pettitt's technical report, the authors described an experiment whereby an Android-based virtual joystick controller (see Figure 4.1) was compared to a traditional controller. Users were asked to drive a PackBot Explorer robot on two courses–indoor and out–using each controller after a brief training period. The majority of the courses were driven while the robot was out of line of sight of the operator, and video feedback was provided on the Android phone in all instances. Observer/controllers following the robot through the course measured user performance via time to complete the course, number of driving errors (collisions), and number of course errors (driving outside the marked areas). Participants, all military service members, also provided feedback via the NASA Task Load Index (TLX) and surveys designed to assess controller feasibility and usability.

Figure 4.1: Army Research Laboratory's Android Operator Control Unit [96]

Results indicated poor performance with the virtual joystick, reaching significance ($p < 0.001$) in mean time to complete the courses, mean number of off course errors, and mean number of driving errors. Additionally, users reported a higher total workload score with the Android controller, specifically on the mental, effort, and frustration scales. Finally, participants rated their own performance with the virtual joystick controller poorly, with 26 of 30 participants preferring the traditional control option [96]. While most users appreciated the Android's light weight, small size, ease of use, and one-handed operation, many complained about the lack of haptic feedback and sensitivity of a rather small virtual joystick. The authors suggested providing larger buttons and/or larger spacing between buttons and incorporating some type of haptic or audio feedback, as many users stated it was difficult to identify when the virtual joystick was "engaged." They also noted previous research using transparent buttons on the screen helping to maximize limited screen real estate, a possible improvement to their own design [96].

The multi-phase experiment presented in this chapter was designed to expand upon this study by comparing a second smartphone-based option built to overcome many of the user-perceived deficiencies of Pettitt's controller. Using attitude aware, tilt-based controls and limiting the user's touch interface, the screen can be devoted to video feedback for tele-operation while providing inherently haptic, physical feedback to the user in the form of the device's attitude in 3D space. By closely mimicking the experiment design of Pettitt et. al., the usability experiment began by comparing ARL's version of virtual joystick control to the newly designed attitude aware controller, with the goal of ascertaining whether the design characteristics of that controller provide a suitable, more satisfying smartphone-based control option. Building upon this, the final experiment phase examined customization of controller options, such as sensitivity and button size, inspired by research question R.2 and common user complaints in the ARL study.

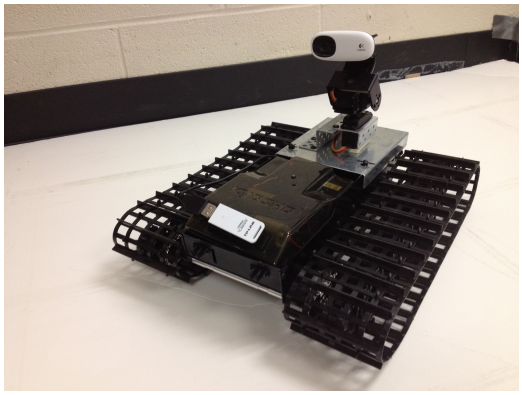
4.3 Hardware

All experiments took place using a modified Kyosho Blizzard SR Ready-to-Run (RTR) remote control tracked vehicle (Figure 4.2a), whose specifications are available in Table 4.1. The factory-provided radio frequency (RF) module was swapped for a Dension WiRC WiFi over RC system, which accommodates eight analog input channels plus two USB video streams and communicates via WiFi dongle over standard 802.11n/g/b wireless in the 2.4GHz range (see Figure 4.2b). Other vehicle modifications include the addition of pan-tilt Lynxmotion camera mounts for the Logitech C110 USB camera, and two standard servos to power their motion (see Figure 4.2c). This setup permitted approximately 120 ° of motion to pan left and right, as well as scan up and down. The fixed camera driving position looked down the centerline of the robot with its front "bumper" in view along the bottom of the frame, coinciding with studies which indicate that video feedback should provide a visual reference to the device chassis [63, 127]. Overall, while smaller and less rugged than the PackBot used in ARL's studies, the Kyosho Blizzard remains a highly maneuverable tracked robot providing a reasonable, if less refined, platform for comparison.

Table 4.1: Kyosho Blizzard SR RTR Specifications

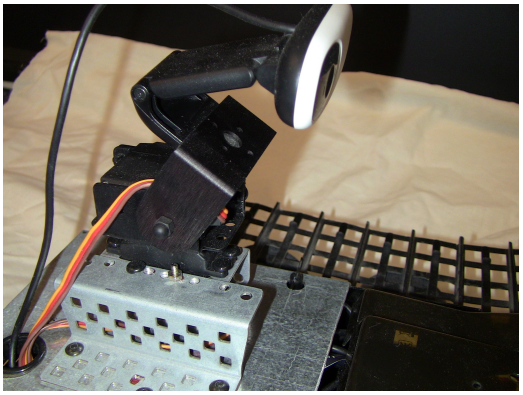
Length:	14.75"
Width:	12"
Height (Chassis):	4"
Height (Camera):	8.5"
Motors:	Twin 370 motors with KA-17W speed controllers
Drive System:	Sprocket and chain
Power:	7.2 V (6-cell) NiMH Battery

Control of the robot was achieved via custom application, described in Chapter 3, developed for both a 4th generation iPod Touch and an iPhone 4, used throughout this experiment. Each was ruggedized by an Otterbox Defender case, making their size, weight, and shape as comparable as possible (see Figure 4.2d).



(a) Kyosho Blizzard SR RTR Modified for Tele-operation

(b) Dension WiRC WiFi over RC [34] (System comes with everything shown here)



(c) Lynxmotion Pan/Tilt Camera Mount with Servos

(d) iPhone 4 and 4th Gen iPod Touch Ruggedized by Otterbox Cases

Figure 4.2: Experiment Hardware

4.4 Course design

An indoor robotic test course was developed, similar to the one used in ARL's experiments [96], to test users in both mobility and reconnaissance tasks. Testing took place in a series of rooms divided further into corridors and doorways to mimic an urban business or residence. Obstacles were placed throughout the rooms in a manner that would force users to maneuver over and around them in order to successfully complete the course. Obstacles varied in type: office furniture, bins, doorjambs, and other miscellaneous items. Doorways and corridors varied from larger 36" openings to narrower 22" openings; furthermore the course included turns in both directions, including some which required maneuvering at more than 90°. See Figure 4.3 for a schematic of the course layout and Figure 4.4 for pictures of the setup.

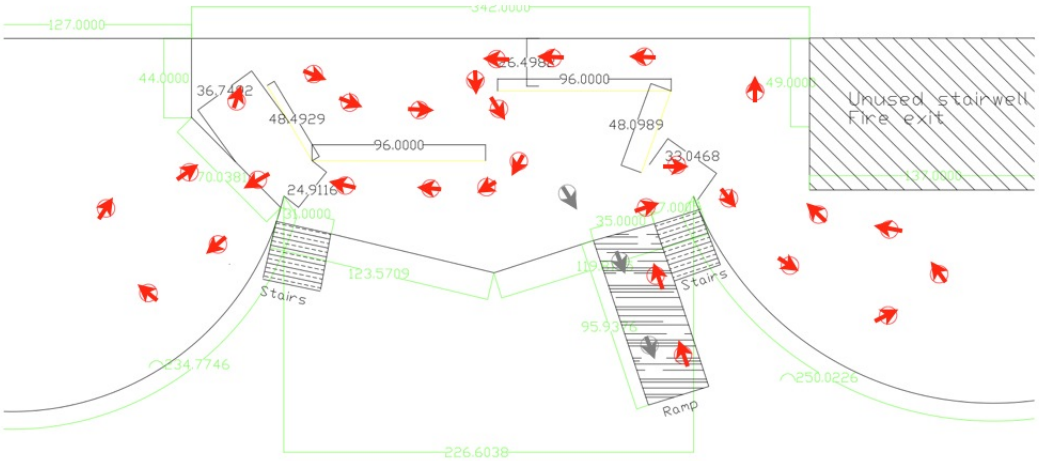


Figure 4.3: Layout of Test Course

A path through the course was marked with red duct tape arrows (depicted in the graphic) spaced 18-24" apart. These provided navigational cues as users followed them counter-clockwise through the course, and served as a measure of driving accuracy that will be further discussed with other dependent variables. Data regarding performance was collected during the course of live trials; however, the entirety of the course was

(a) Picture of Course (Left) (b) Picture of Course (Middle) (c) Picture of Course (Right)

Figure 4.4: Pictures of Course

also overlooked by a series of four closed circuit cameras used to record robot trials as a redundancy measure.

4.5 Method

Each the three experiment phases compared two similar, but different controllers, using the tracked robot described in Section 4.3. Controller type was examined as a within-subject variable, meaning each user tested with both available controller options in each phase. The order in which controllers were presented to users was counterbalanced across the first two phases to counteract both learning and ordering effects (see Table 4.2). In Phase 3, Controller E was the only controller presented.

4.5.1 Independent Variables

The independent variables in this experiment were *controller type* and *task*. Tasks stayed constant within a phase, but changed between them, building upon a user's experience with the system to add robot capabilities while increasing task complexity. Exactly how and why this was done is explained in the definition of each experiment phase. A total of five controllers were presented to each user over the course of three phases. They are summarized in Table 4.3 and later described in detail.

Table 4.2: Experiment Counterbalance Design

A, B, C, D, and E are levels of the independent variable, *controller* (see Table 4.3)
X, Y, G, and H are *runs*, or specific sets of object locations used in the reconnaissance task

Participant	Test				
	1	2	3	4	5
1	A	B	C-X	D-Y	E-H
2	B	A	D-Y	C-X	E-H
3	A	B	D-X	C-Y	E-G
4	B	A	C-Y	D-X	E-G
5	A	B	C-X	D-Y	E-H
6	B	A	D-Y	C-X	E-H
7	A	B	D-X	C-Y	E-G
8	B	A	C-Y	D-X	E-G
9	A	B	C-X	D-Y	E-H
10	B	A	D-Y	C-X	E-H
11	A	B	D-X	C-Y	E-G
12	B	A	C-Y	D-X	E-G
13	A	B	C-X	D-Y	E-H
14	B	A	D-Y	C-X	E-H
15	A	B	D-X	C-Y	E-G
16	B	A	C-Y	D-X	E-G
17	A	B	C-X	D-Y	E-H
18	B	A	D-Y	C-X	E-H
19	A	B	D-X	C-Y	E-G
20	B	A	C-Y	D-X	E-G
21	A	B	C-X	D-Y	E-H
22	B	A	D-Y	C-X	E-H
23	A	B	D-X	C-Y	E-G
24	B	A	C-Y	D-X	E-G
25	A	B	C-X	D-Y	E-H

Table 4.3: Summary of Controllers as the Independent Variable

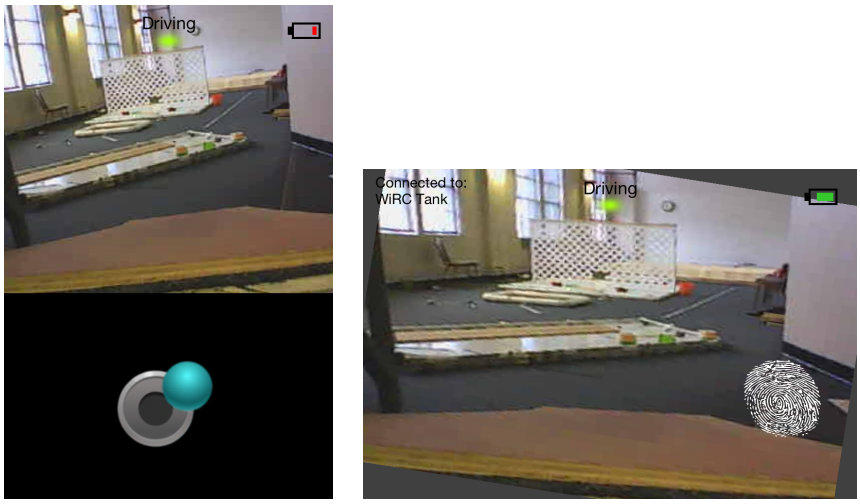
Controller Label	**Phase**	**Application Name**	**Driving Mode**	**Camera Mode**
A	1	Joystick v1	Joystick	Disabled
B	1	Tilt v1	Tilt	Disabled
C	2	Tilt + Joy v2	Tilt	Joystick
D	2	Tilt v2	Tilt	Tilt
E	3	Custom v3	Custom	Custom

4.5.2 Definition of Experiment Phases

Phase 1. Phase 1 was designed as a direct extension of Pettitt's ARL study. As such, an iOS version of their Android OCU was developed, defined as **Controller A** (see Figure 4.5a). Presented in portrait mode, the upper half of the screen was devoted to video feedback while the bottom half of the screen housed the virtual joystick. Control of the robot was achieved by dragging the joystick freely in any direction; when the joystick was released and/or returned to center, all robot motion ceased. **Controller B** consisted of a stripped down version of the custom attitude aware application described in Chapter 3. Users were presented with a control interface in landscape mode with full-screen video feedback. A thumbprint button located in the bottom right of the screen served as the deadman switch and activated tilt controls when pressed and held (see Figure 4.5b). Deadman switch activation also re-leveled the controls, meaning that the attitude at which controls were activated became the new neutral point. Users then tilted the device forward and backward to control the throttle and rotated left and right to control heading. Releasing the deadman switch stopped the robot.

The task included driving the robot around the course as prescribed during training. Users would be unable to see the robot, although were within auditory feedback range. The robot began at the bottom of the ramp, following the arrows first to the right, then through the remainder of the course. Users were instructed to strive for an equal balance of speed, accuracy, and precision, whose applicable performance measures (time, path points hit, and number of collisions) will be defined as dependent variables.

Phase 2. Phase 2 added control of the pan/tilt camera affixed to the robot. In an effort to manage task complexity, camera motion was only accessible when the robot was stopped. Users were directed to drive their robot along the same course used in Phase 1 and stop to visually scan for three objects along the way. In order to mitigate the effects of individual scanning strategy, duct tape boxes were added to the course



(a) Controller A: Joystick v1

(b) Controller B: Tilt v1

Figure 4.5: Phase 1 Controllers

(one in each room), from which users were directed to scan for the object(s) specified. Boxes were positioned to ensure that object(s) would be visible if users properly scanned all 360° of a room from within them. The objects used were colored foam balls 3.5" in diameter (see Figure 4.6a). Users saw the balls in advance of their timed runs and were aware of which color would be present in each room, though objects were not in position during training. Upon finding an object, participants took a picture of it using the camera button at the base of the controller and mapped its approximate location on the worksheet provided (see Figure 4.7). Users were asked to place an "X" where they thought they had found the object. While not imperative to understanding the control task, these results were used as a measure of a participant's ability to localize their robot in space, and plays towards overall spatial ability as it relates to tele-operation. Two *runs* were executed, X and Y. Each run consisted of a set of three object locations, one in each room as shown in Table 4.4. The runs were designed such that difficulty was roughly equal and objects could be found when the room was scanned, but would not be found by accident in the course of driving. Users saw each run once, counterbalanced along with control type and order as depicted in Table 4.2.

Table 4.4: Definition of Runs for Phases 2 and 3 (Object Locations)

	Robot Rm (Right)	**Stage** (Center)	**Server Rm** (Left)
Run X/G	Table	Chair	Circuit
Run Y/H	Ladder	Corner	Floor

*runs were re-named for Phase 3 so that users were unaware that the same object locations were being utilized.

(a) Foam Balls used for Visual Identification Tasks

(b) Objects as Seen through Robot Camera

Figure 4.6: Phase 2 Visual Identification Task

Both controllers in Phase 2 adopted attitude aware driving controls, but maintained different modes for camera manipulation. **Controller C** used a virtual joystick to control the camera pan/tilt by dragging left to look left, up to look up, etc. The joystick was roughly equal in size to the driving deadman switch and located in the lower left hand corner (see Figure 4.8a). **Controller D** presented the user with the driving deadman switch (a plain thumbprint button) on the bottom right of the screen, and a camera deadman switch (a thumbprint with camera overlay) on the bottom left of the screen (see Figure 4.8b). It used attitude aware tilt-based inputs for driving *and* camera manipulation. Both worked in the same way–press and hold to activate tilt controls, release to stop. In both Controller C and D, the camera remained focused on its current target when the joystick or deadman switch were released, allowing users to look more closely at the frame and/or take a picture. To ensure users did not become

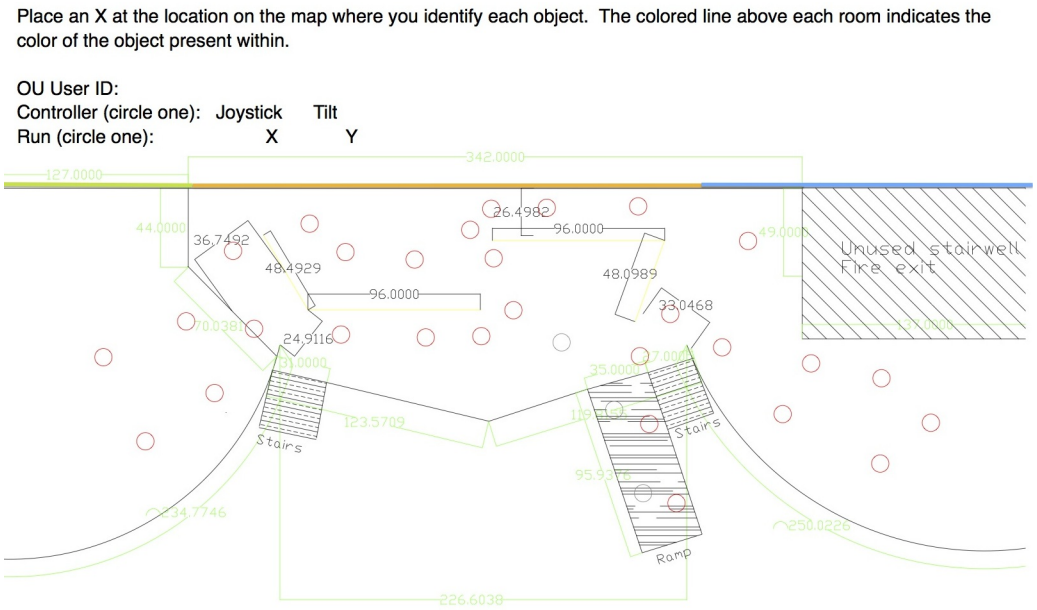


Figure 4.7: User Worksheet for Mapping Object Locations

disoriented when driving, the camera returned to its neutral position (facing forward on the centerline of the robot with the front bumper visible in the bottom of the frame) each time the driving deadman switch was activated.

The intent of adding reconnaissance tasks in Phase 2 was to test the suitability of each type (mode) of control input for the robot's third and fourth degree of freedom (camera pan and tilt), while also examining the effects of interface mode confusion.

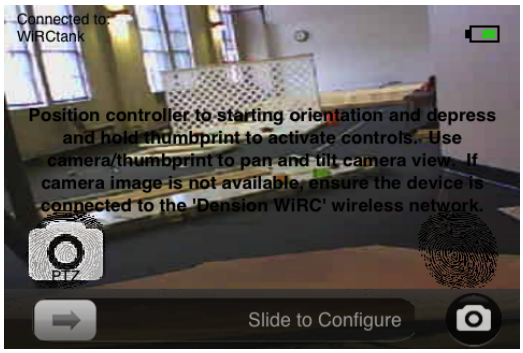
(a) Controller C: Tilt + Joy v2

(b) Controller D: Tilt v2

Figure 4.8: Phase 2 Controllers

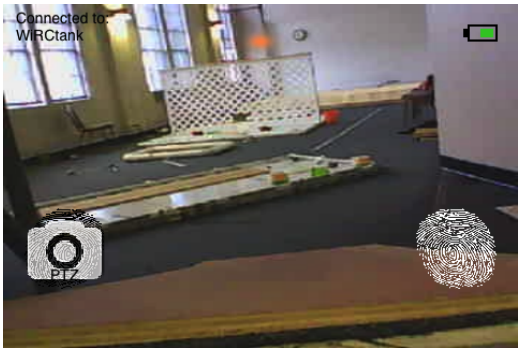
Work by Chong and Lankenau indicates that mode confusion may result when a user's mental models differ from the system's actual model and/or when feedback is insufficient to indicate operation mode [26, 69]. In Controllers C and D, feedback is provided primarily via soft buttons highlighting upon touch, as well as the presence (or absence) of driving lights. Given the similarities of the control inputs, i.e. press, hold, and tilt, it was important to consider whether users experienced mode confusion with either set of controls, especially considering all robot degrees of freedom are accessed from the same interface screen.

Phase 3. Phase 3 culminated in allowing users to *customize* their controls! After exposure to multiple combinations of driving and camera controls, participants were given the option to modify their controller, including choice of control mode(s) and interface layout. The default controller was chosen as Controller D, and the application began in that version for all users. By double-tapping the center of the screen and sliding to unlock (see Figure 4.9a), users entered a settings menu (Figure 4.9b) from which they could choose either joystick or tilt controls for both driving and camera channels. Based on those choices, users could then position the on screen components (thumbprints and/or joysticks) in one of six locations around the screen. Additionally, users could adjust sensitivity and responsiveness via sliding scale, changing the maximum throttle speed and device range of motion. All of these settings were thoroughly explained to each user before decisions were made regarding the final configuration, referred to as **Controller E**, an example of which is shown in Figure 4.9c. Aside from asking users to configure their custom controller, tasks for Phase 3 remained identical to Phase 2. Users were again asked to drive the robot through the course balancing speed, precision, and accuracy, while stopping in each of the three scanning boxes to identify the objects placed throughout the rooms.



(a) Controller E
"Slide to Open Settings" Interface

(b) Controller E Settings Menu
visible at launch, scroll down for more



(c) Controller E:
Final Configuration Example

Figure 4.9: Phase 3 Control Interface

4.5.3 Dependent Variables

Experiment dependent variables consist of both quantitative and qualitative information collected during the course of training and timed trials. Performance measures recorded include practice time, time to complete the course (in seconds), number of major driving errors, number of minor driving errors, and number of path points hit. Driving errors were differentiated based on severity. A major error included instances where the observer had to step in a make a correction e.g. repositioning the robot, putting it back on the course after a major deviation, etc. Minor errors were collisions from which the operator could self-correct, often engaging reverse. The arrows depicting the path through the course were called path points and were considered "hit" if any part of the robot came in physical contact with them while traveling in the direction in which they were pointing. There were 32 total path points, and the individual administering the experiment noted the number hit as well as their location.

In addition to these performance metrics, information was also collected for the tasks specific to Phase 2 and 3. Users, after locating each object, mapped them and took a picture. Photographs were saved to provide confirmation that an object was visually identified, allowing future analysis of the object's position within the frame. The maps, as described previously, provided a secondary measure of users' spatial awareness and were graded on a nine-point scale. For each object marked, users could earn three points, for nine points total. The scale was defined as: zero points for failing to mark, or being completely wrong; one point for identifying the object on the wrong wall, but one adjacent to its actual position; two points for noting the object along the correct wall but in the wrong position; and three points for identifying the object within 10% of its

exact location.

Video feedback was also collected throughout experiment trials. Cameras recorded not only the robot's actions (primarily as a backup), but also the user's reactions, movements, and facial expressions while guiding their robot through the course. The webcam built into the user workstation was used to record while the live feed was hidden from users so as not distract them. These videos were cataloged and archived but are not currently being utilized. They represent hours of tele-operation and could easily produce interesting insights if analyzed as part of a more psychologically focused extension of this research. In addition to these video recordings, .plist files were also saved from the device application (per user) during Phase 3. These preserved the final controller configurations used by each participant in the trial and informed analysis regarding control preferences and defaults.

Following each timed trial, users were asked to complete the NASA TLX workload index and provide usability feedback via surveys hosted on SurveyMonkey.com. The NASA TLX was administered using a desktop application by Playgraph and included both the Likert scale and pairwise comparisons [51, 57]. Surveys were customized for each phase of the experiment and were vaguely inspired by the surveys used in ARL's study. In Phases 1 and 2, users completed post-iteration surveys after each timed trial and a post-experiment survey at the end of the phase. In Phase 3, given there was only one timed trial, only one survey was administered. The questions (and results) comprising each survey can be viewed in their entirety in Appendix C. Survey questions focused on usability and user preference, and the answers helped guide design decisions between phases i.e. which features to customize, how feedback should look. Most questions were answered on a five-point Likert scale, but some

permitted free text commenting, providing perhaps the most revealing evidence of user attitudes.

4.6 Procedures

Permission to proceed with this study was granted by the University's Institutional Review Board (IRB) on 31 August 2012, filed under IRB #1115. All necessary disclosures and forms specific to that process are included in Appendix B.

4.6.1 Participants

Upon study enrollment, a signed consent form was collected for each of the 25 participants, all of whom indicated their consent to be video taped. Each subject completed a demographics survey (seen in Appendix C) regarding age, gender, military service, and experience with relevant technologies. The majority of participants were recruited from the undergraduate and graduate engineering populations at the University of Oklahoma. They ranged in age from 18 to 51, with a median age of 26. The subject population consisted of 21 men and four women, recruited to represent the military gender split (approximately 85% male and 15% female). While 4 participants reported military experience, none were exposed to unmanned systems during their service. Only one individual reported being left-handed.

All participants were compensated a total of $40 for their five-hour time commitment spread across three experiment phases. No previous experience with either smartphones or robots was necessary to participate, although 76% reported good or excellent proficiency with mobile devices, with 22 respondents reporting

at least 12 hours of weekly use. Questions regarding use of remote controlled systems of varying intelligence indicated that nine participants had little to no experience with ground vehicles, and only one reported excellent proficiency. More participants reported exposure to systems using gestural based inputs, the most common being Nintendo's Wii; 20 participants reported experience with that system, and 14 participants had played tilt-based games on a mobile device, such as the iPhone. Finally, a spatial reasoning test [122] was administered to assess a user's natural spatial ability, a proven predictor of user performance in tele-operation tasks [121, 120, 25]. The mean score for study participants was 13.4 out of 20, with a median of 14, matching the referenced average for all test takers. Only two subjects tested notably below average, with scores of 8 and 9.

4.6.2 Training

All users began the study by participating in a formal pre-interview, where they were introduced to the project and its time requirements. Trials were scheduled individually, with each user required to attend a 1.5 hour session for each of the three experiment phases. Upon arrival to the test location, users were briefed on the experiment tasks. This always began with a physical walkthrough of the indoor course, pointing out key features and familiarizing the user with the types of obstacles which he/she might encounter. This method of course familiarization was chosen over map reconnaissance (or a completely unknown environment) in an attempt to control for individual differences in spatial ability. Providing firsthand, physical knowledge of the course and its layout was determined to be the best way to limit the task to *driving* without *navigating*; where driving requires only *manipulation* of the robot, whereas navigating implies some measure

of finding one's way.

Levels of the independent variable, controller type, were presented one at a time in the order prescribed by the counterbalance table. A specific training regimen was followed for each controller with a training check used to confirm when users were ready to commence timed trials. This ensured that all users began with a similar set of baseline skills, adjusting for biases which might have existed due to user experience, or lack thereof, with similar technologies. Following formal training, which included a walkthrough, verbal instructions, and/or instructional videos, the remainder of training time was devoted to hands-on practice. Considered a flexible practice period, hands-on practice time was intended to provide a dependent variable measuring controller ease of use, where actual time used (up to 15 minutes for Phases 1 and 2, and 30 minutes for Phase 3) was recorded in the experiment log. Exact training procedures changed slightly between experiment phases; Table 4.5 provides a full description.

Table 4.5: Training Procedures by Experiment Phase

Phase 1	*Walkthrough*	Yes; to familiarize users with the course and point out path points.
	Controller Familiarization	5-min video instruction for each controller type; users then had three minutes to ask questions/receive clarification.
	Hands-on Practice	Users were given a maximum of 15 minutes for hands-on practice and were asked to instruct the experimenter when they felt competent enough to proceed
	Practice Type	Users were limited to beyond line of sight operation only after the first three minutes.
	Training Check	Drive robot around the wall in the center of the course in less than two minutes without any major collisions.
Phase 2	*Walkthrough*	Yes; re-familiarized users with the course and pointed out the addition of three scanning boxes.
	Controller Familiarization	5 min video instruction for each controller type; no time restrictions on question/answers.
	Hands-on Practice	Users were given a maximum of 15 minutes for hands-on practice and were asked to instruct the experimenter when they felt competent enough to proceed.
	Practice Type	Strictly tele-operation. Users were permitted to practice both driving and scanning tasks, as objects were not present on the course during training.
	Training Check	Experimenter observed practice to ensure user was competent with new camera controls; no formal checks.
Phase 3	*Walkthrough*	A walkthrough was offered for indivduals who required re-familiarization.
	Controller Familiarization	10 min hands-on demonstration by the experimenter where settings menu items were discussed and demonstrated in detail.
	Hands-on Practice	Users had 30 minutes to practice and configure their robot controller. Questions were answered at any time.
	Practice Type	Users could be anywhere on or off the course either with the robot or beyond line of sight for the duration of the hands-on practice period.
	Training Check	Experimenter observed practice to ensure user was competent with custom controls; no formal checks.

4.6.3 Timed Trials

Official timed trials commenced on each controller immediately following training using that same controller. Participants remained seated at the user workstation, just below the main room where the the test course was located, to ensure that users were beyond robot line of sight, therefore relying on video feedback to maneuver their robot through the course. Ensuring users were seated also served to eliminate accidental movements which might errantly impact the attitude controls i.e. body rotation while standing, shifting from foot to foot, etc. The motion algorithm described in Section 3.3.1 acts to level and filter most of these motions; however, for experimental purposes, controlling for noise in the attitude inputs was deemed prudent. The remainder of task completion varied with phase and will be discussed in the chapters that follow.

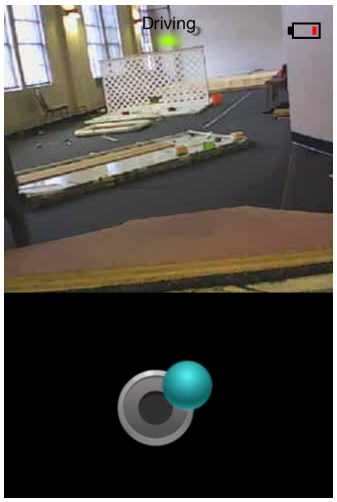
Chapter 5

Experiment & Results – Phase 1

5.1 Task

Phase 1 experiments were conducted using Controllers A and B (see Figure 5.1), described in detail in Chapter 4. Official timed trials commenced on each controller immediately following training using that same controller. The participant remained seated at the user workstation just below the main room where the the test course was located (see Figure 5.2) to ensure that all driving was via tele-presence while maintaining a safe distance between the robot and subject during operation. Users were videotaped at their workstation to capture facial expressions, major movements, etc., and the robot was filmed from above while traversing the course as a redundancy measure.

The robot started the course at the bottom of an eight foot ramp while the participant waited for the signal to begin. Users were instructed to complete the course as quickly and as accurately (based on bath points hit) as possible while minimizing collisions. Users maneuvered their robot around the entirety of the course while the experimenter followed along collecting performance data. When users reached the finish point at the base of the ramp, total trial time was noted and all video recordings were stopped.



(a) Controller A:
Joystick v1

(b) Controller B:
Tilt v1

Figure 5.1: Phase 1: Controllers A and B

Figure 5.2: User Workstation

Users then transitioned to post-iteration data collection, filling out the NASA TLX via desktop application and completing a web-based questionnaire regarding controller features and usability. Once both of these were complete, the next controller option (if applicable) was prepared for the participant. He/she trained with the new controller in the same manner previously described, executed a timed trial, and finished with the same post-iteration questionnaire and TLX. Once both timed trials were complete, the user answered a final post-experiment questionnaire, primarily to identify which controller, of the two presented, was *preferred*.

5.2 Independent/Dependent Variables

In Phase 1, the independent variable was the controller type: A (joystick) or B (tilt-based); making this a one-factor experiment with two levels. The 14 dependent variables used for statistical analysis were a combination of performance results, formal measures of workload and usability (NASA TLX and System Usability Scale), and informal measures of user satisfaction collected via survey. These metrics were collected for each user trial: 25 users x 2 controller levels = 50 trials. Specifically,

- **ptime.** Practice time was the duration of hands-on practice for each subject, not to exceed 15 minutes, collected as a quantifiable measure of ease of use and controller intuitiveness.

- **ttime.** Trial time, in seconds, was the amount of time it took each user to drive the robot from start to finish. There was no upper limit to the amount of time a subject could take to complete this task.

- **majerror.** Major errors were those requiring intervention on the part of the experimenter. They primarily ensured that users who struggled with specific portions of the tele-operation task, e.g. depth perception, disorientation, etc., could still complete the course as intended (i.e. following the correct route).

- **minerror.** Minor errors were small collisions of the robot against some obstacle, generally requiring the user to engage reverse. They served as a measure of driving precision and encouraged users to consider metrics other than speed alone.

- **pathpts.** 32 path points existed along the course as red, duct taped arrows. Users were instructed to hit as many of them as possible with some portion of their robot. Results were recorded as a raw number between 0 and 32, representing driving accuracy.

- **tlx.** The NASA TLX is a standardized measure of mental workload administered via desktop application. Users completed it immediately following each controller trial. Responses are totaled to a workload score between 0 and 100; 100 being maximum mental demand.

- **sus.** The System Usability Scale (SUS) is a similarly standardized tool built to be a generic, quick, and simple measure of usability for industrial systems [18]. Comprised of ten questions on a five-point Likert scale, overall usability is resolved to a number between 0 and 100; 100 being most usable.

- **move.** Post-iteration surveys were intended to help identify failings in the training approach while also allowing the user to rate his/her abilities without quantifiable knowledge of their trial performance. The primary

questions asked referenced the user's ability to complete specific tasks with each controller. For instance, users ranked their ability to "move in the correct direction" on a five-point scale from *extremely difficult* (1) to *extremely easy* (5).

- **obstacles.** Next, users rated their ability to "avoid obstacles."
- **slow, med, hi.** Users also rated their ability to "maintain control when driving at slowest speeds...medium speeds...and fastest speeds."
- **drive.** The final question regarding driving tasks required users to rate their "overall ability to perform driving tasks" on the *extremely difficult* to *extremely easy* scale, or one to five.
- **usability.** Likewise, users informally rated each controller's overall usability, as a secondary, but more direct, measure of usability.

5.3 Statistical Analysis

Phase 1 is formally a one-way repeated measures design, where each user attempts each level of the independent variable, controller type. It can be analyzed using a standard t-test or with a two-way analysis of variance (ANOVA), where both *controller* and *subject* are factors. In experiments with human subjects, where several measurements are taken on the same experimental unit (person), measurements tend to be correlated. When these measurements are responses to levels of the experimental factor (controller type), correlation can be captured using ANOVA [29]; therefore, the F-statistic was used to identify significant effects of the independent variable on the results.

5.3.1 Hypothesis

In ANOVA, the null hypothesis, H_0, is defined as the condition where population group means are equal; specifically, it states that the means of the dependent variables for Controllers A and B are equal. The alternate hypothesis, H_1, is NOT the null hypothesis, although it may not specify which factor is favored [29]. When $p < 0.05$, the observed effect is likely not due to chance, meaning it can be attributed to the independent variable, although this alone does not confirm causation.

The research questions (R.1 and R.2), introduced in Chapter 1, developed into three hypotheses, the first of which was tested in Phase 1. All were developed under the assumption that virtual joystick controls would be more familiar to users, and were therefore likely to be preferred. Users often resist new technologies, such as tilt-based controls, which have not yet achieved widespread use. The hypothesis, H.1, intends to test these assumptions and biases.

(H.1) *Over time, and after reasonable training, users will be able to perform surveillance and reconnaissance tasks to a reasonable standard and* ***equally as well*** *with tilt inputs as with a virtual joystick.*

(H.2) *Tilt-based controls are intuitive enough a control modality to be used for a number of robotics applications with multiple degrees of freedom,* ***without significant degradation of performance****.*

(H.3) *Permitting users to manipulate certain controller settings will lead to* ***more satisfied users who perform better with less errors****.*

Hypotheses H.2 and H.3 will be revisited in Chapters 6 and 7 respectively; also, note that Phase 3's defined hypothesis takes the form of an *alternate,* not the null.

5.3.2 Analysis of Variance (ANOVA): The F-Statistic

The F-statistic (Eqn. 5.1) compares group variance to global variance, where MS = the mean square deviations (variances)–derived by dividing the sums of squares by degrees of freedom.

$$F = \frac{MS_{between}}{MS_{within}} \tag{5.1}$$

The F-distribution is indexed by two parameters, degrees of freedom (DOFs):

1. $j - 1$ where j is number of groups
2. $N - j$ where N is number of trials

F is expected to be 1.0, and large values speak against the null hypothesis. The critical value of F required to formally reject the null hypothesis is found in a table of the F-distribution, indexed by the two DOFs defined. What results is the probability of obtaining a value greater than or equal to that $F_{critical}$ by chance under the null hypothesis. More simply, if the probability (p) is low, generally less than 0.05, the null hypothesis can be rejected when $F \geq F_{critical}$.

This statistic is subject to some important limitations, which should be considered before confidently accepting its results. First, populations from which each group is drawn are assumed to be normal. Second, variances of these populations are assumed to be equal. Third, error components should be independent of one another within trials (in tests with replication), as well as

between trials [29]. *Replication* refers to the same factor and task completed by the same individual more than once e.g. a user drives a robot under the same conditions with the same controller multiple times. A *repeated measure* indicates that the same individual conducted a task twice, but does not imply that the level of the independent variable remained the same.

The first two assumptions regarding normality are generally accepted if sample sizes are sufficiently large, $N > 30$. Therefore, this experiment's sample size ($N = 50$), should be satisfactory for ANOVA; however, a normality check was conducted to confirm. The histogram in Figure 5.3 shows the distribution of trial times from all 50 samples in Phase 1 and has the traditional bell curve expected of a normal distribution. The mean and median are also sufficiently close together to prevent skew. The statistics of normality in Table 5.1 support this assessment.

Table 5.1: Normality Statistics

Basic Statistical Measures			
Location		*Variability*	
Mean	322.72	Std Deviation	87.05899
Median	314	Variance	7579
Mode	314	Range	412
Interquartile Range	88		

Tests for Normality			
Test	*Statistic*		*p Value*
Shapiro-Wilk	W	0.952891 $Pr < W$	0.0449
Kolmogorov-Smirnov	D	0.114579 $Pr > D$	0.0975
Cramer-von Mises	W-Sq	0.123395 $Pr > W - Sq$	0.0531
Anderson-Darling	A-Sq	0.708619 $Pr > A - Sq$	0.0633

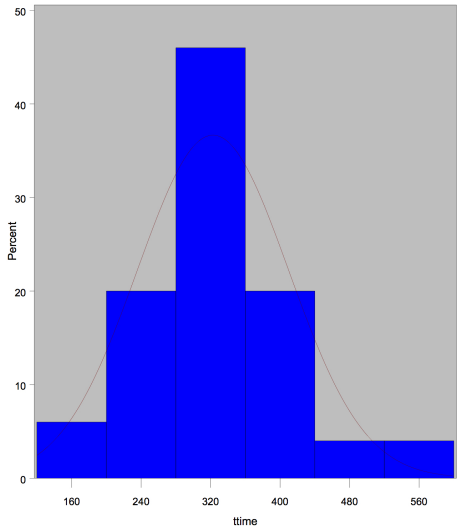


Figure 5.3: Histogram of Phase 1 Trial Times ($N = 50$)

The final limitation is dependent on the error estimation technique(s) available given the experiment design. In tests with replication, error is accounted for by analyzing the variance between replicates; in tests without replication, it must be estimated in other ways. For instance, repeated measures designs estimate error using the variation among (not between) individuals [76]. The repeated measures present in this experiment are sufficient to provide these error estimates–handled by the statistical software package used for analysis.

5.4 Results & Discussion: The F-Statistic

To test the null hypothesis (H.1), the F-statistic was calculated for each of the 14 dependent variables defined. $p < 0.05$ was the critical value used to denote significance. Table 5.2 shows the means, standard deviations, and statistics for each dependent variable. Those achieving significance are highlighted in yellow.

Table 5.2: The F-Statistic: Phase 1

		ptime	ttime	majerror	minerror	pathpts	tlx	sus
Controller A	*mean* ($\bar{x}$)	687.72	305.88	0.28	5.64	28.6	55.31	64.9
	std dev (s_d)	213.19	69.4	0.61	2.6	2.02	14.78	16.9
Controller B	*mean* ($\bar{x}$)	648.68	339.56	0.4	4.16	28.28	53.89	68.7
	std dev (s_d)	245.01	100.34	0.76	2.75	2.67	20.09	17.08
	p (< 0.05)	0.445	*0.046*	0.417	*0.049*	0.55	0.695	0.258
	F	0.6	4.41	0.68	4.29	0.36	0.16	1.34

		move	obstacles	slow	med	hi	drive	usability
Controller A	*mean* ($\bar{x}$)	2.44	2.52	3.48	2.625	1.72	2.96	3
	std dev (s_d)	0.8206	0.8226	1.005	0.7697	0.9798	0.8888	1.04
Controller B	*mean* ($\bar{x}$)	3.52	3.08	3.68	2.8	1.88	3.2	3.56
	std dev (s_d)	0.9626	0.9539	1.1445	1.08	0.8327	1.041	1.08
	p (< 0.05)	*0.0001*	*0.0076*	0.4091	0.5884	0.3563	0.282	*0.045*
	F	20.68	8.49	0.71	0.3	0.88	1.21	4.46

5.4.1 Performance

Five variables achieved significance at the $p < 0.05$ level: **ttime, minerror, move, obstacles,** and **usability**, meaning the *null hypothesis is rejected* given that controller means are not equal. Trial time favored the joystick controller, matching observations made during experiment trials. Joystick runs were notably faster than those with tilt controls (in fact, significantly so). This appeared to be the result of a small virtual joystick with limited travel between center and bezel. Users struggled to find the joystick's "sweet spot," resulting in most operating at top speed, or close to it, for the duration of their runs. Many adapted to those conditions by driving in short bursts, although collisions/minor errors were significantly higher as a result ($\bar{x}_A = 5.64$ vs. $\bar{x}_B = 4.16$, $F = 4.29$ at $p = 0.049$).

The three other variables resulting in significance, all favoring the tilt controller, resulted from user responses to the post-iteration questionnaires. **move** references users' rated ability to "move in the correct direction." Multiple individuals noted that the joystick controller failed to operate as expected. Many found it difficult to drive straight ahead. While users were quick to blame this

on the control mapping, most instances of drift were actually caused by user error; some experienced a disconnect between where they *thought* their thumb (or finger) was in contact with the joystick and where it actually was. Given the relatively small size of the joystick [75 x 75 pixels], a small variance in touch location yielded a noticeable effect on robot behavior. This phenomena is believed to be a contributing factor to the difference in user ratings, leading to the significance observed in the **move** variable.

Users also rated the attitude aware controller significantly better with respect to their ability to avoid **obstacles**, indicating that obstacle avoidance is not only tied to general tele-operation skills, but also related to the control type in use. It is not immediately clear why obstacle avoidance was superior with the tilt controls, as users did complain that turning was more difficult with it. Observations indicated that most problems when turning in place resulted when users accidentally engaged reverse. It was natural for users to tilt the controller back towards themselves while rotating the device, and if, in doing so, the device tilted beyond the set neutral point, the robot began to drive backwards. That said, users with tilt controls did report better control over the entire range of the robot's throttle, as indicated in answers to survey questions regarding control at **slow** ($\bar{x} = 3.68$), **med** ($\bar{x} = 2.8$) and **hi** ($\bar{x} = 1.88$) speeds (see Figure 5.4b). It could be this perceived sense of control that had the largest impact on users' self-rated ability to avoid obstacles.

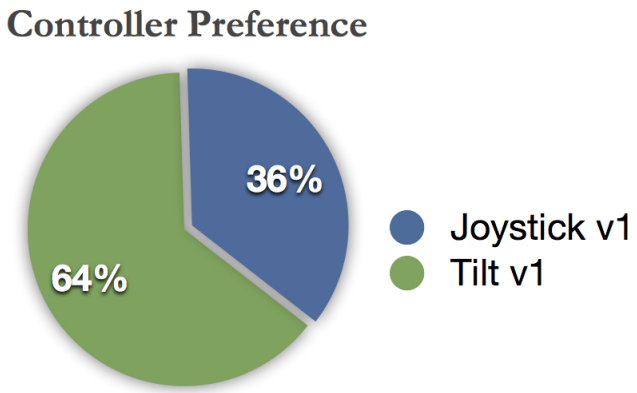
While the formal measure of usability, **sus**, did not achieve significance, it was closely related to user-rated **usability**, which did. Given that SUS was developed as a general measure of system usability (in an era of desktop-based systems), it may not provide the most accurate picture of controller usability. Lending confidence to the more informal measure of user-rated usability, on a scale of

1 to 5, is the fact that both it ($\bar{x} = 3.56,\ p = 0.045$) and user preference (A: 36%, B: 64%) indicated support for the same controller (tilt-based). When, at the conclusion of the experiment phase, participants were asked to choose which controller they preferred, 16 of 25 chose Controller B (see Figure 5.4a).

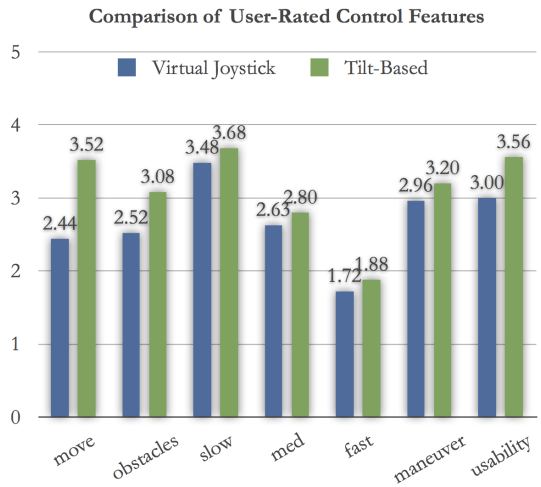
5.4.2 User Preference

The role of user preference is important to any usability assessment, and here perhaps even more so. Given the lack of experience with tilt controls in the general population, and a task that few have previously attempted (robot tele-operation), the hypothesis (H.1) was conservative in assuming that users might *prefer* the virtual joystick regardless of performance outcomes. Joystick control was posited to be more familiar to users and a direct mapping of current tactile controls. Lack of familiarity can sometimes doom even the most usable new systems, as consumer resistance to change is well documented [64, 104]. That said, ARL's study made clear that users had a number of issues with virtual joystick control; problems which the tilt controller was designed to overcome. Looking at numbers alone, it would appear to have done so, given the higher percentage of participants preferring tilt controls; but how is that preference tied to performance?

Table 5.2 shows that trial time was significantly faster for the joystick controls, and as one of three quantifiable measures of performance, it could be expected that users would summarize their own performance with Controller A superior to that of Controller B. Instead, survey results indicate the opposite–users rated their abilities superior with the tilt controls in all categories: **move, obstacles, slow, med, hi** and **drive**. Additionally, while 18 users performed better with



(a) Phase 1 Overall User Preference
(Virtual Joystick Control vs. Tilt Controls)



(b) Phase 1 Results Showing User-Rated Driving Abilities
(on a scale of 1-5, with 5 being *extremely easy*).

Figure 5.4: Phase 1 Results Summary

the joystick controller in terms of trial time, only nine preferred it. Examining the percent improvement between each user's best and worst controller trial averaged an 18% difference, regardless of whether a participant drove faster with Controller A or Controller B. This confirms that preference is not obscured by percent improvement in trial times, as was assumed.

Instead, it seems to suggest that one or more of the other performance metrics ranked higher in users' own determination of their abilities. Participants were not informed of their trial times, nor were any stopwatches or clocks visible to them during their runs. However, knowledge of performance regarding path points hit and driving errors was more difficult to shield from users. While final numbers were not shared, users generally knew how many times they had run in to something and could easily compare one run against another. Likewise, while participants could not always be certain that their robot ran over a path point, users almost certainly knew if they missed one entirely. Therefore, it makes sense that individuals might use those metrics to judge overall performance as much, if not more so, than trial time. Of the nine users who drove faster with the joystick yet preferred the tilt controls, eight of them committed less minor errors with Controller B. Path points indicated less relationship to preference, as only three of nine users improved their driving accuracy with tilt-based controls; five hit fewer path points, and one user's performance held constant. Finally, both the NASA TLX and SUS results indicated a relationship to preference when performance metrics alone could not explain a user's choice. It appears as if most users preferred the controller which yielded the lower workload score and higher usability ratings (both, again, hidden from the user). Ten users reported higher workload and lower SUS scores with Controller B, and of those, only three still stated they preferred the tilt controls. Clearly no one variable is enough to

predict a user's preference, but this provides some insight in to how users rate the various performance metrics in their own assessments. Further relationships between variables are examined using statistical correlation.

5.5 Results & Discussion: Correlation

Correlation is another important statistical tool that can help identify relationships between dependent variables. It provides a measure of association between two variables on a scale from -1 to +1. -1 implies a negative correlation, meaning that as variable X increases, variable Y decreases. +1 implies a positive correlation, meaning that as variable X increases, variable Y does as well; zero indicates no correlation. Pearson's correlation coefficient (ρ) is the most widely used, where correlation reflects a *linear* relationship; however, it is not well suited to this data given the number of ordinal variables present.

Alternatively, Spearman's coefficient (ρ) can be applied, where the ρ value between -1 and +1 indicates only a *monotonic* relationship, not a linear one. By ranking data before evaluating the correlation function, Spearman's coefficient is better equipped to handle variables with repeated values or those with non-linear, discontinuous relationships [83]. Table 5.3 shows the significant relationships using Spearman's coefficient. Those with strong correlations ($-0.5 > \rho > 0.5$) and achieving significance are highlighted.

Table 5.3: Spearman's Correlation Coefficient (ρ): Phase 1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1. ptime	——													
2. ttime	0.00524	——												
3. majerror	-0.02290	*0.505*	——											
4. minerror	-0.14001	0.1065	0.1583	——										
5. pathpts	-0.11139	0.0397		-0.396	——									
6. tlx	0.06653	0.27603	0.1383	0.2899	-0.1315	——								
7. sus	-0.05831	-0.327	-0.1696	-0.216	-0.2317	*-0.622*	——							
8. move						-0.4502	0.4603	——						
9. obstacles				*-0.4174*		-0.403	0.438	*0.5411*	——					
10. slow						-0.404	*0.6009*	*0.5447*	*0.5092*	——				
11. med	-0.3006	-0.296					0.4284	*0.5568*	0.443	*0.58011*	——			
12. hi								*0.5094*	0.354	0.3818	*0.7352*	——		
13. drive	-0.07277	-0.14934		-0.2005	-0.08006	*-0.589*	*0.7147*	*0.665*	*0.50677*	*0.6477*	*0.587*	*0.5147*	——	
14. usability	-0.09791	-0.15354		-0.328	-0.1237	*-0.569*	*0.8006*	*0.6786*	0.4558	*0.75632*	*0.5283*	0.3407	*0.783*	——

Usability and SUS are the most strongly associated variables, with $\rho = 0.8006$. This confirms that user-rated usability matches closely with results of the ten question System Usability Scale. These, in turn, are also closely related to both **drive** and **slow**, indicating those measures' importance to a user's perspective of ease of use. **drive**'s relationship to usability makes sense, as it is how users rate their *overall* driving ability. **slow**, on the other hand, is less obvious; however, it is much more strongly correlated ($\rho_{slow-sus} = 0.6009$) to both measures of usability than answers to either of the other speed questions ($\rho_{med-sus} = 0.4284$, ρ_{hi-sus} = non-significant). This implies that users relied on their ability to control the robot at the slowest speeds to, at least partially, inform their overall impression regarding usability. Given users' struggles to maintain slow speeds with Controller A, it is not entirely surprising that Controller B achieved better mean scores on all user-rated metrics. As further evidence of the importance of the **slow** variable, consider that **med** and **hi** are closely related ($\rho = 0.7352$), but neither is strongly tied to **slow**. This demonstrates that the slowest speeds reside in a category all to themselves, set apart from the faster throttle settings which are most likely to result in loss of control. **slow** was also most closely tied to results of the NASA TLX workload index and the System Usability Scale, further implying its importance to the results. SUS represents the formal measure of usability and, as expected, is negatively correlated with the NASA TLX workload score. More usable systems should rank lower in terms of workload, and that appears to be the case here.

ttime and **majerror** are correlated, indicating that as major errors increased, total trial time did as well. This relationship is supported by the amount of time it takes the experimenter to reset the robot after a major error, naturally increasing trial time. Variable means for **ttime** and **majerror** are lower for the

joystick controller than for the tilt controller, although nearly all other metrics favor Controller B. This, when paired with data regarding minor errors, indicates that users were faster, whether due to a lack of major errors or some other factor, *and* that speed and accuracy (e.g. **pathpts**) are not mutually exclusive given that drivers with the joystick controller hit more path points, on average, than with the tilt controls. In fact, while not an overwhelmingly strong correlation, minor error and path points show a slightly negative association, meaning that as errors go down, path points go up, and vice versa. One might believe that better robot operators are both more accurate and less prone to driving errors. Why, if these values are related, does the tilt controller result in significantly fewer minor errors and the joystick controller result in (on average) more path points hit?

The scatterplot shown in Figure 5.5, illustrates a lack of variance in the path point term. In fact, while $\bar{x}_A = 28.6$ was slightly higher than $\bar{x}_B = 28.28$, the scatterplot indicates that most all users achieved between 27-31 data points in their trials. Additionally, the joystick controller's ability to out-perform with regards to path points, on average, does not imply likewise minimization of minor errors. The relationship between path points and minor errors is more likely reliant upon individual users than the level of the independent variable. Users prioritized the balance of speed, accuracy, and precision differently, demonstrated during trial execution. Some sacrificed speed for accuracy, while others placed speed above all else, and committed the driving errors to prove it.

5.6 Analysis of Covariance (ANCOVA)

The ANOVA already presented assumes no significant effect of pre-treatment or existing conditions within the participant population; however, accounting

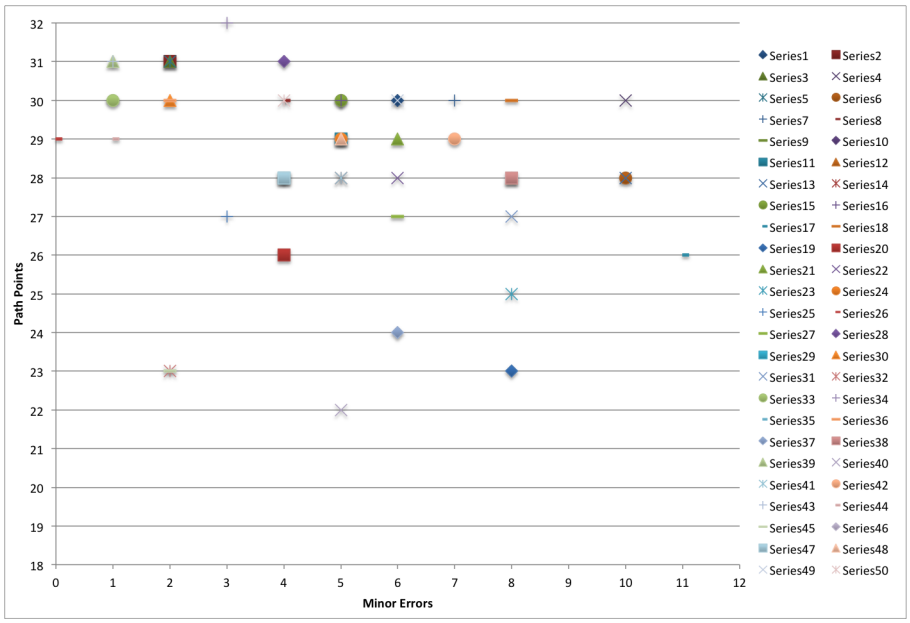


Figure 5.5: Scatterplot of Minor Errors vs. Path Points

for such conditions can greatly influence final results. To do so, ANOVA is augmented with covariates (the pre-treatments or participant characteristics), commonly referred to as analysis of covariance (ANCOVA).

Participant demographics were treated as covariates, and tests were conducted to filter their effect on results. Four covariates were examined: spatial reasoning scores, gender, age, and technical proficiency with mobile devices; however, none had a significant effect on the outcomes of the dependent variables. Given previous research confirming the role of spatial ability in tele-operation [22, 120, 121], it is somewhat surprising that those scores did not have a meaningful impact here. That said, additional analysis of covariance is presented in Chapter 6, where trials of greater complexity are examined.

5.7 Summary & Other Results

The results already presented clearly indicate that users preferred the tilt controller for driving tasks. Users made significantly fewer minor driving errors with Controller B, supporting their perception of improved control. Likewise, users rated their driving ability superior with the tilt controller for all tasks. These results led to rejection of the null hypothesis, and exceeded expectations which anticipated that attitude aware controls were inferior to virtual joysticks. Rather, Phase 1 implies that tilt controls are *better*, and certainly feasible, control options. In fact, in answer to the research question motivating this experiment, tilt controls are *not* negatively perceived by users when properly implemented and certainly prove that smartphones can be implemented as small, lightweight controller options without heavy reliance on the touchscreen interface.

A great deal of data was collected to inform the results presented in this chapter, where they were summarized for ease of presentation. Full results, including user surveys, comments, and experimenter observations, are available in Appendix C.

Chapter 6

Experiment & Results – Phase 2

6.1 Task

Phase 2 experiments were conducted with Controllers C and D (see Figure 6.1), described in detail in Chapter 4. Official timed trials commenced on each controller immediately following training with that same controller. Training in Phase 2 consisted of a course walkthrough, to re-acquaint the users to the course, and video demonstrations of the two controllers presented. The driving task in this phase remained identical to Phase 1: follow the path points, hitting as many as possible, while avoiding collisions and completing the course in a timely manner.

(a) Controller C: Tilt + Joy v2

(b) Controller D: Tilt v2

Figure 6.1: Phase 2: Controllers C and D

Visual identification tasks were added to test users' abilities using two different control modes (joystick and tilt) to manipulate the robot's pan/tilt camera. In order to limit the effect of individual scanning strategies, pre-determined *scanning boxes* were located along the course, one in each room (see Figure 6.2). Users were asked to drive their robot into the scanning boxes and confine searching tasks to those locations; by surveying the entire room from those points, users *should* have been able to identify the colored foam balls depicted in Figure 4.6a.

Figure 6.2: Scanning Boxes for Visual Identification Tasks

Once an object was located, users were asked to take a photograph of it using the controller's snapshot tool, then map it on the worksheet in Figure 6.3. This mapping task was graded on a nine-point scale, with three points available for each object placed properly. Following trial completion, users once again transitioned to post-iteration data collection, filling out the NASA TLX and web-based questionnaires regarding controller features and usability. Once both of these were complete, the next controller option (if applicable) was prepared for the participant. He/she trained with the new controller in the same manner previously described, executed a timed trial, and finished with the same post-iteration questionnaire and TLX. Once both timed trials were complete, the

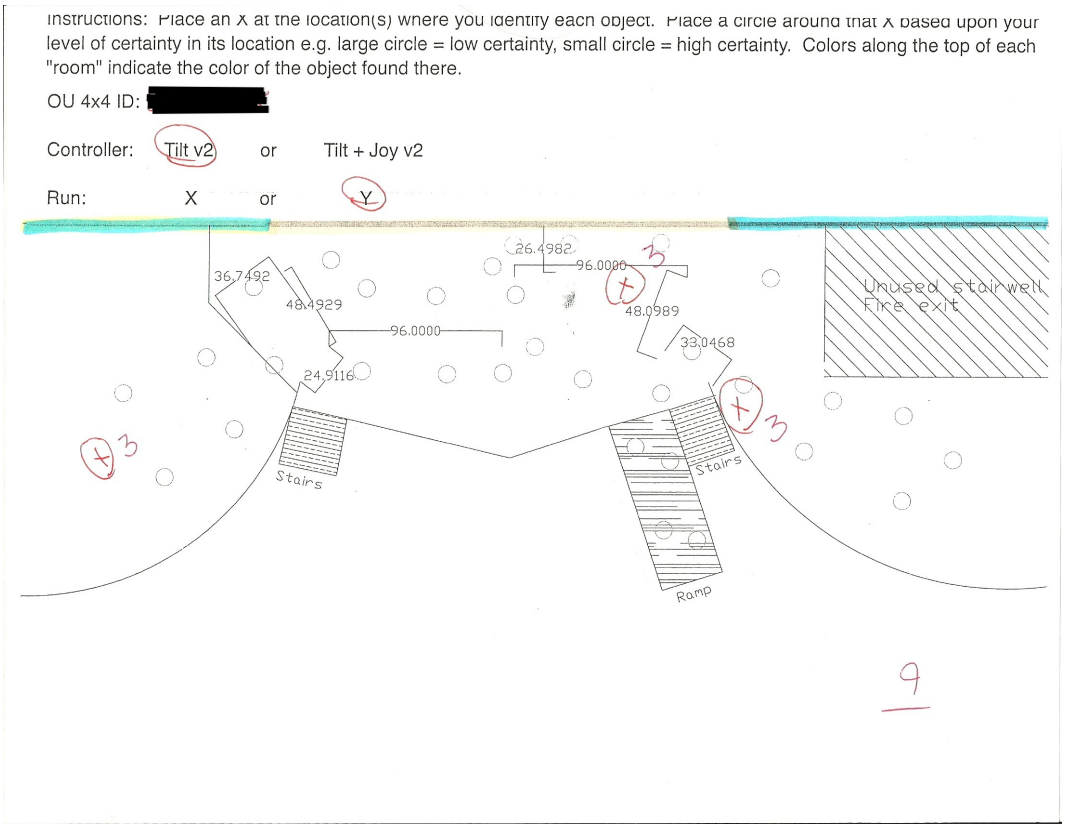


Figure 6.3: Completed User Worksheet from Phase 2

user answered a final post-experiment questionnaire, which primarily required them to identify their preferred controller of the two presented. The Phase 2 post-experiment questionnaire also asked users to rate the importance of certain user settings anticipated as an expansion to the controllers already presented. These are discussed in more detail in Section 6.7.

6.2 Independent/Dependent Variables

The independent variable in Phase 2 was the controller type, C (tilt to drive; joystick camera) or D (tilt to drive; tilt-based camera); once again a one-factor experiment with two levels. The dependent variables used for statistical analysis

were a combination of performance results, formal measures of workload and usability (NASA TLX and System Usability Scale), and informal measures of user satisfaction collected via survey. **ptime, ttime, majerror, minerror, pathpts, tlx, sus,** and **usability** were collected in the same manner specified in Phase 1, and are only briefly re-defined below:

- **ptime.** Practice time was the duration of hands-on practice for each subject, not to exceed 15 minutes.

- **ttime.** Trial time, in seconds, was the amount of time it took each user to complete the course. It included all scanning, picture taking, and mapping subtasks associated with visual identification.

- **majerror.** Major errors were those requiring intervention on the part of the experimenter.

- **minerror.** Minor errors were small collisions of the robot against some obstacle, generally requiring the user to engage reverse.

- **pathpts.** 32 path points existed along the course as red, duct taped arrows. Total hit with some portion of the robot was recorded as a raw number between 0 and 32, representing driving accuracy.

- **localization.** Localization is the result of user-mapped object locations (like those in Figure 6.3). Each "X" could be worth up to three points, for a total of nine per trial: zero for failing to map/identify the object, one for mapping it on a wall/surface adjacent to the object's actual location, two for mapping it on the correct wall/surface but in the incorrect location, and three points for mapping it within 10% of its actual location.

- **tlx.** The NASA TLX is a standardized measure of mental workload administered via desktop application. Users completed it immediately following each controller trial. Responses are totaled to a workload score between 0 and 100; 100 being maximum mental demand.

- **sus.** The System Usability Scale (SUS) is a similarly standardized tool built to be a generic, quick, and simple measure of usability for industrial systems. Comprised of ten questions on a five-point Likert scale, overall usability is resolved to a number between 0 and 100; 100 being most usable.

- **pan.** Post-iteration surveys were intended to help identify failings in the training approach while also allowing the user to rate his/her abilities without quantifiable knowledge of their trial performance. The primary questions asked referenced the user's ability to complete specific visual identification tasks with each controller. For instance, users ranked their ability to "pan the camera" on a five-point scale from *extremely difficult* (1) to *extremely easy* (5). *Pan* is defined as side to side motion.

- **tilt.** The next relevant survey question asked users to rate their ability to "tilt the camera." *Tilt* is defined as up and down motion.

- **looking.** Next, users rated their ability to "identify which direction the camera is looking with respect to the robot."

- **scanning.** Similarly, users rated their ability to use the controller in "scanning surroundings."

- **stills.** The final question regarding camera manipulation tasks asked users to rate the ease of "capturing still photographs" on the *extremely difficult* to *extremely easy* scale, or one to five.

- **usability.** Likewise, users informally rated each controller's overall usability.

6.3 Hypothesis

As always in ANOVA, the null hypothesis is the condition in which population group means are equal, implying no statistically significant difference between the independent variables in terms of the dependent variables described. The hypothesis motivating Phase 2's experiment was:

(H.2) *Tilt-based controls are intuitive enough a control modality to be used for a number of robotics applications with multiple degrees of freedom,* ***without significant degradation of performance****.*

Degrees of freedom here refers to unique robot channels which control dissimilar robot activities, e.g. driving and camera manipulation. There is some indication that using similar controls for dissimilar tasks can lead to mode confusion [26, 69], hence the need to test tilt-based controls specifically under such conditions. The statement "without significant degradation of performance" indicates that equal performance between the two controller levels is expected, while implying that the alternate hypothesis would NOT favor the tilt-based condition. In most cases throughout this research, the virtual joystick control is assumed to be the more familiar operation mode, and hence the control mode most likely to result in higher user satisfaction and performance. In all instances thus far, tilt-controls are deemed successful if they are *at least as competent* as their joystick counterparts.

6.4 Results & Discussion: The F-Statistic

Analysis of variance was once again used as the statistical test to examine the effect of the independent variable on the 14 dependent variables already defined. In this phase, none of these relationships proved significant (see Table 6.1), leading to a failure to reject the null hypothesis, thereby confirming that tilt-based controls are suitable for multiple robot degrees of freedom, performing no worse than the joystick alternative.

Table 6.1: The F-Statistic: Phase 2

		ptime	**ttime**	**majerror**	**minerror**	**pathpts**	**localization**	**tlx**
Controller C	*mean* ($\bar{x}$)	193.00	539.96	0.04	3.32	25.44	7.2	44.72
	std dev (s_d)	140.31	154.64	0.2	1.8	3.42	1.58	13.93
Controller D	*mean* ($\bar{x}$)	152.32	556.56	0.12	3.12	26.08	6.64	46.07
	std dev (s_d)	94.08	124.75	0.33	1.99	3.7	1.96	14.73
	p (< 0.05)	0.228	0.584	0.327	0.639	0.227	0.223	0.393
	F	1.53	0.31	1	0.23	1.54	1.57	0.76

		sus	**pan**	**tilt**	**looking**	**scanning**	**stills**	**usability**
Controller C	*mean* ($\bar{x}$)	77.40	4.00	4.08	3.72	4.04	4.48	4.13
	std dev (s_d)	12.47	1.08	0.99	0.79	0.84	0.77	0.54
Controller D	*mean* ($\bar{x}$)	75.10	4.12	4.21	4.00	3.96	4.44	4.04
	std dev (s_d)	13.12	0.78	0.83	0.82	0.93	0.82	0.86
	p (< 0.05)	0.389	0.600	0.524	0.183	0.714	0.814	0.479
	F	0.77	0.28	0.42	1.88	0.14	0.06	0.52

6.4.1 Performance

Practice time notably decreased compared to Phase 1, mostly due to learning effects. Users were already acquainted with the tilt-based driving controls and, aside from a brief reintroduction to the course, spent most of their hands-on practice familiarizing themselves with the task conditions and camera controls. Trial times were very similar for the two controllers, failing to achieve significance ($\bar{x}_C = 540s, \bar{x}_D = 557s$). Likewise, the **pathpts** DV failed to reject the null hypothesis ($\bar{x}_C = 25.44, \bar{x}_D = 26.08$); it did, however, decrease substantially from

Phase 1, where 28 points were found, on average, across 50 trials. Users struggled with the increased task complexity of Phase 2, and the shared "priorities" of speed, driving accuracy, avoidance of collisions, *and* visual identification tasks, led to degraded performance in the path points variable. Amazingly, this was the only performance area that suffered between the two phases, as both error variables (**majerror** and **minerror**) exhibited improved driving precision.

Within Phase 2, performance conditions were not expected to vary greatly, as the driving task remained the more time intensive portion of the trial, and stayed constant between Controller C and D. Many users were able to identify the objects from their robot's position in each scanning box in approximately one minute, devoting about three minutes per trial to visual identification tasks. In cases where users struggled to find an object, they were asked to continue along the course after three minutes had elapsed in one location. For some individuals, a non-trivial amount of time was then spent identifying the location of the found object for mapping; this differed greatly from person to person–user narratives describing each user's own scanning strategy are presented in Appendix C.

localization is used to describe the dependent variable measuring the accuracy with which users identified where items within the robot's view were positioned in space. Multiple user comments implied that localization was equally difficult with both controllers, with a mixture of users indicating a clear preference for one over the other. In the case of camera pan/tilt, a visually intense task, a *visual* representation of the camera's position might seemingly satisfy users more than the *physical* representation provided by the tilt-based controls, but user-rated **looking,** which answered the question "identify which direction the camera is looking with respect to the robot," demonstrates otherwise ($\bar{x}_C = 3.72, \bar{x}_D = 4.00$). Interestingly, the value of **localization** (performance

based), slightly favored the joystick control, and is not strongly associated with the the **looking** variable. Similar to what was seen in Phase 1, users' own assessments of their abilities and preferences do not always match performance results!

6.4.2 User Preference

Survey results indicate that 14 users preferred the joystick camera controls, while the remaining 11 preferred the tilt-based camera controls. Results from Phase 1 indicated that trial time was not the best indicator of user preference, and results here support that conclusion. 13 users completed their trial faster with Controller C, and 12 with Controller D, with an average trial time improvement of 21%. However, *13 users* also preferred the controller on which they did not perform most quickly. Six of those participants performed best with Controller D, yet preferred Controller C, while the other seven participants did the opposite. Neither **pathpts** nor the error variables showed a relationship to user preference; likewise, TLX and System Usability Scores, partly due to a lower variance in those values in Phase 2 trials. Instead, preference seems more closely related to the *order* in which controllers were presented to each user. While theoretically this ordering effect was accounted for by counterbalancing presentation of the independent variables, in the case of user preference it cannot be wholly eliminated. 20 users preferred the controller they used last, indicating that learning effects, perhaps magnified due to the control similarities, did play a role in user preference.

pan, tilt, and **scanning** results slightly favored the tilt-based camera controls, with a number of users describing them as "smoother." Joystick

controls were described as "intuitive" and "less physical," contributing to lower mental workload scores, although several individuals complained that the camera movement should be slowed down considerably. None of these preferences were dramatic enough to prove statistically significant, but should be taken in context with the whole of user comments provided in Appendix C.

6.5 Results & Discussion: Correlation

While the F-statistic failed to reject the null hypothesis, interesting statistical associations did exist between dependent variables when examined via Spearman's correlation coefficient, ρ, in Table 6.2. NASA TLX scores and results of the System Usability Scale were again negatively correlated, reinforcing expectations first confirmed in Phase 1–when usability improves, mental workload decreases (see Figure 6.4a). Usability and SUS scores were also related, indicating that user-rated usability tends to reflect the trend in formal usability scores.

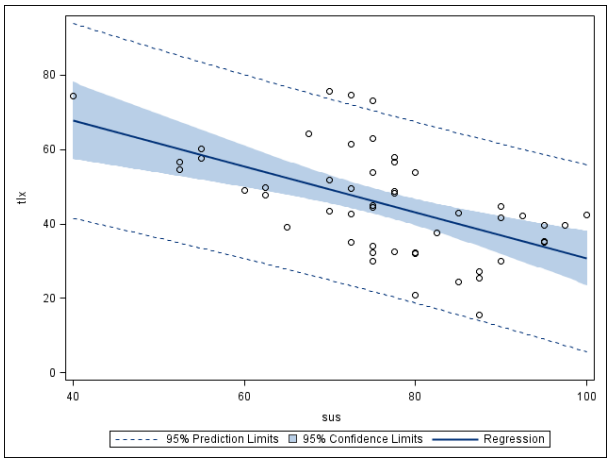
Results regarding specific camera control characteristics indicate that pan and tilt are closely related ($\rho = 0.82$); meaning of the two controllers in this phase, whichever is good at one of these tasks will likely be good at the other. **scanning** was the "summary" variable for camera control, encapsulating all of the manipulation subtasks, asking users to rate their ability in "scanning [their] surroundings." It was positively correlated with both measures of usability ($\rho_{\textbf{sus-scanning}} = 0.5371; \rho_{\textbf{usability-scanning}} = 0.6196$), and presents itself as the most indicative measure of usability with regards to camera control features. The graphical relationship between SUS and **scanning** is shown in Figure 6.4b.

Table 6.2: Spearman's Correlation Coefficient (ρ): Phase 2

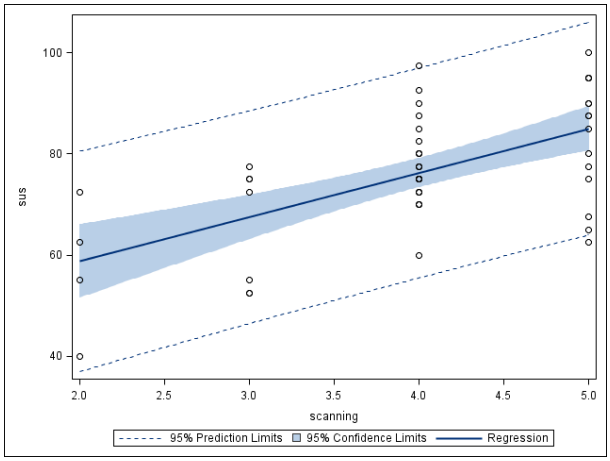
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1. ptime	———													
2. ttime	0.2206	———												
3. majerror	-0.2018	0.1277	———											
4. minerror	-0.0479	0.1807	0.3433	———										
5. pathpts	-0.047	-0.2681	-0.1592	-0.1696	———									
6. localization	-0.1459	-0.327	-0.2054	-0.0009	0.179	———								
7. tlx	0.0867	0.4567	0.0255	-0.0455	-0.2607	-0.1947	———							
8. sus	-0.2299	-0.3107	0.0666	0.1787	0.0759	0.30336	*-0.594*	———						
9. pan	-0.0855	-0.0855	0.2415	0.0972	0.091	0.2279	-0.4609	0.3979	———					
10. tilt	-0.1178	-0.2913	0.1523	0.1086	0.0205	0.1073	-0.4722	0.3296	*0.8245*	———				
11. looking	-0.1924	-0.1924	-0.2179	0.0253	0.0499	0.0654	-0.2684	0.1234	0.3718	0.4118	———			
12. scanning	-0.0947	-0.1385	0.2783	0.2371	-0.0336	0.1402	-0.4361	*0.5371*	*0.7574*	*0.6748*	*0.5611*	———		
13. stills	0.2764	-0.0842	0.1028	-0.0149	0.0771	0.0735	-0.1404	0.2588	0.0138	0.0837	0.2363	0.2531	———	
14. usability	-0.2815	-0.1945	0.1898	0.2962	0.0228	0.0918	-0.4592	*0.6658*	0.4986	*0.5197*	0.3182	*0.6196*	0.3197	———

The results in Figure 6.4 show graphs where one dependent variable is plotted against another. The solid blue line shows the strength of the correlation (linear regression) in the *sample*. The shaded blue area indicates the 95% confidence limits, or the range within which the *population* correlation coefficient is likely to reside. The dashed blue lines show the 95% prediction limits, within which the next data point sampled is expected to fall. Neither of these is crucial to understanding these results; however, they provide an indication of the population's expected correlation (confidence) and its distribution/scatter (prediction).

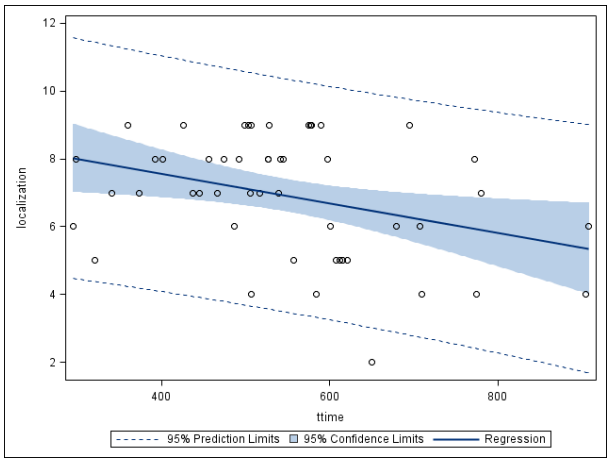
Figure 6.4c depicts another interesting relationship, between **ttime** and **localization**. While not an overtly strong correlation ($\rho = -0.327$), it is worth another look because of the negative association and what it implies about user abilities. One would expect that the most spatially aware users could localize objects in space more easily, resulting in faster overall trial times. Users for whom localization was more difficult would likely produce lower localization scores while taking longer to do so. That said, this correlation implies more about the role of spatial ability in user performance than it provides insight regarding any specific controller option.



(a) **sus** vs. **tlx**



(b) **scanning** vs. **sus**



(c) **ttime** vs. **localization**

Figure 6.4: Spearman's Correlation Plots: Phase 2
Solid blue lines = regression/correlation; Dashed blue lines = prediction limits;
Blue shaded regions = confidence limits (95%)

6.6 Analysis of Covariance (ANCOVA)

Analysis of covariance was therefore conducted, with a specific interest in the aforementioned relationship between a user's spatial ability and his/her performance on the object-mapping task, **localization**. Figure 6.5 and Table 6.3 show, surprisingly, that no statistically significant relationship is present. This does not entirely negate the possibility of a relationship, since previous studies have proven one exists, but instead indicates that the chosen *measure* of spatial ability in this study may have been poorly suited to the task type.

Table 6.3: Analysis of Covariance: Controller & Users' Spatial Abilities

Source	**DF**	**Type III SS**	**Mean Square**	F **Value**	$Pr > F$
controller	1	3.92	3.92	1.27	0.2657
cov_spatial	1	6.56	6.55	2.12	0.1519

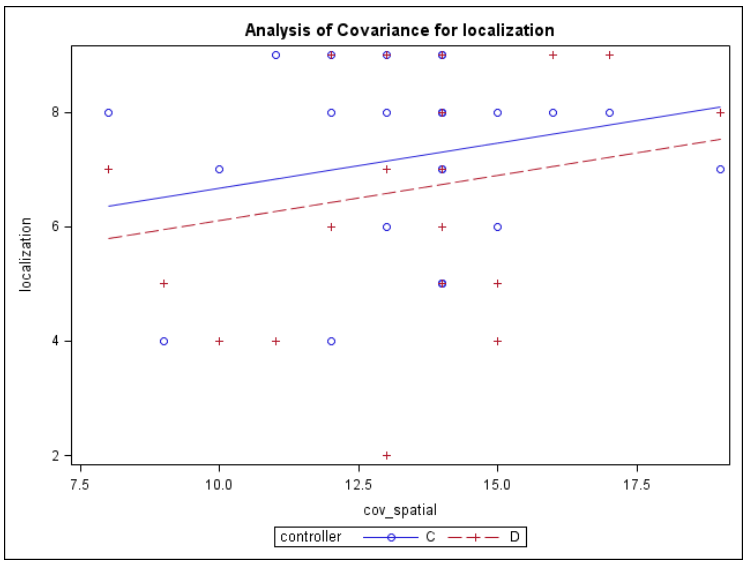


Figure 6.5: Analysis of Covariance:
spatial ability vs. **localization**

6.7 Summary & Implications for Phase 3

Results from Phase 2 failed to reject the null hypothesis, indicating that tilt-based and joystick controls performed equally well when used with additional robot degrees of freedom (pan/tilt camera). User satisfaction demonstrated very little difference between the two, as both proved more than adequate to complete the visual identification tasks required in this experiment phase. On the whole, users improved at all performance metrics between Phases 1 and 2, with the exception of driving accuracy (measured by **pathpts**). This was primarily due to the increase in task complexity and the difficulty users experienced trying to prioritize multiple subtasks. While more users preferred the joystick-based camera controls, none of the dependent variables indicated a statistically significant relationship to that level of the independent variable.

User feedback regarding ability to scan with the camera appeared as the strongest indicator of controller usability, as usability scores in total increased from Phase 1. SUS scores in Phase 1 averaged 66.8 over 50 trials, while in Phase 2 SUS scores averaged 76.2. Time spent with the controls likely affected this improvement, as learning effects were present. They were believed to have contributed to user preferences, where 20 of 25 users favored the second of the two controllers presented to them in Phase 2.

6.7.1 Mode Confusion

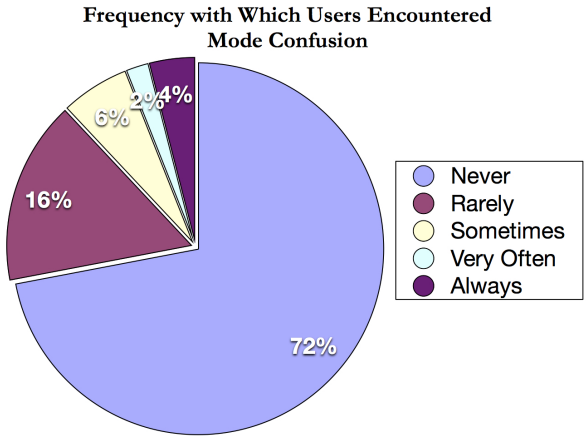
Aside from determining user performance and satisfaction with each of the two controllers, Phase 2 was additionally designed to test the suitability of tilt-based controls for two dissimilar robot actions (driving and camera manipulation). In complex control interfaces, it is not uncommon for users to encounter confusion

regarding which controls are currently active and/or which buttons and gestures result in the outcome they are expecting; this is called mode confusion. This research raises a question asking whether using the same control mode to operate distinctly different robot actions eases this confusion or exacerbates it. Controller C presented a situation where two control modes (joystick and tilt) separated and defined the two robot actions they controlled (camera and driving), while Controller D used a single control mode, accessed through two separate activation buttons (deadman switches) to control both robot camera and driving via tilt inputs.

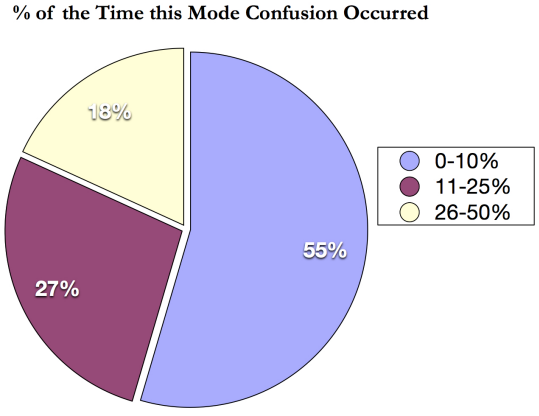
To capture these effects, users were each asked, in post-iteration surveys, whether they encountered any mode confusion in their trials. Users could respond *never, rarely, sometimes, very often*, or *always.* These instances of confusion could include activating the wrong part of the robot, e.g. moving the camera when the user intended to drive, or confusion regarding input mode, e.g. trying to drag the deadman switch like a joystick! Those that answered anything but *never* were then asked to rate how often the confusion occurred, as a percent of all control activations. Figure 6.6 graphically illustrates these results.

Eight individuals reported some mode confusion with Controller C, while six individuals reported confusion with Controller D. Four of these individuals reported confusion with both controller types. Severity of confusion does not appear to be significant, and frequency of confusion remained low. In all, mode confusion appears much more closely related to the user than it does the controller. User-reported technical proficiency indicated no deficiency on the part of the four participants who experienced persistent mode confusion; however, their gaming experience ranks less adept than the participant population, on average. Users rated their skills with video game systems (like the XBox) and

smartphone games on a scale from *no experience* (1) to *excellent* (5). The sample population averaged 3.38 on video game skills (slightly better than *average*), while the four participants examined here averaged 3, or *average*. On smartphone game skills, the disconnect was even greater, with the sample population reporting a 3.28 average and the participant subset averaging just 2.5 (between *poor* and *average*). Gaming skills have been previously linked to tele-operation abilities by Chen [22]; however, their relation to mode confusion, specifically, has not been documented.



(a) Frequency with which Mode Confusion Occurred



(b) Mode Confusion as a Percent of All Control Activations

Figure 6.6: Mode Confusion Charts

6.7.2 Customization

Phase 2 also helped setup investigation of research question (R.2), asking "Can controller customization options improve user satisfaction and performance?" Already it is apparent that users find the tilt-based controls satisfactory, but can the user experience be further improved (and as a result, performance) by allowing users to choose their own settings? Pettitt indicated that controller *sensitivity* was the most common complaint regarding virtual joystick usability [96]; user comments exhibited individuals with varying expectations and suggested that a user-defined sensitivity variable might be needed.

In order to confirm Pettitt's findings, and in addition identify other settings which users were apt to change, the post-experiment survey for Phase 2 collected information specific to customization. Table 6.4 rank orders the proposed customization options by average user score, from *not at all important* (1) to *very important* (5).

Table 6.4: User-Rated Importance of Customizable Controller Options

Rank	**Feature**
4.52	Sensitivity
4.28	Choice of control mode
3.24	Camera's neutral position
3.24	Camera's behavior upon driving
3	Feedback regarding camera limits
2.96	Driving switch location on screen
2.84	Camera switch location on screen
2.8	Driving switch type (hard or soft button)
2.64	Camera switch type (hard or soft button)

As expected, sensitivity rated highly, with nearly all users rating it as *very important*. Sensitivity can be thought of as a combination of throttle sensitivity

and controller responsiveness, both of which will be addressed in more detail in Chapter 7.

Choice of control mode refers to a user's ability to pick between joystick or tilt-based controls for both types of robot operation (driving or camera). Users generally exhibited strong controller preferences, illustrated in comments to the experimenter and frustration levels throughout the trials. While there was no general consensus to this polarity, it does explain why most users desired a choice over control mode. The interesting question is, which will they choose?!

As discussed in Chapter 3, the camera was designed with a specific driving neutral point in mind, which matched best practices supported by literature. On this robot, the camera was situated at the robot's midline, approximately 4.5" above the robot chassis, and it's field of view included the robot's front "bumper." Not all users liked this setup, and several requested the camera look further ahead of the robot; one of the questions asked users if they desired control over this camera trim position, or if they were satisfied with the default provided. Most users were fairly *neutral* in this regard (score = 3).

The last few custom features polled were deemed *less than important*, including the ability to move deadman switch buttons on the control screen. Assuming that in landscape mode most users would manipulate the touchscreen controls with their thumbs, the deadman switch defaults were within easy reach at the bottom corners of the screen. Only a few individuals stated a desire to alter that location, and even then only slightly, e.g. nudging the button further from the phone's bezel to account for larger fingers. Finally, the question regarding switch type was included to test users' desire (or lack thereof) for a hard button deadman switch, available via device volume inputs. Given the laboratory conditions of these experiments, users regarded this option as *less than important*;

however, this would likely change if/when tests are moved to an outdoor location and/or the controls are handled by gloved users, such as soldiers. A hard button option was presented to appease that audience, and reception of said choice was unsurprisingly *neutral* in an audience where tasks and conditions were tightly controlled, unlike "field" operations.

Chapter 7

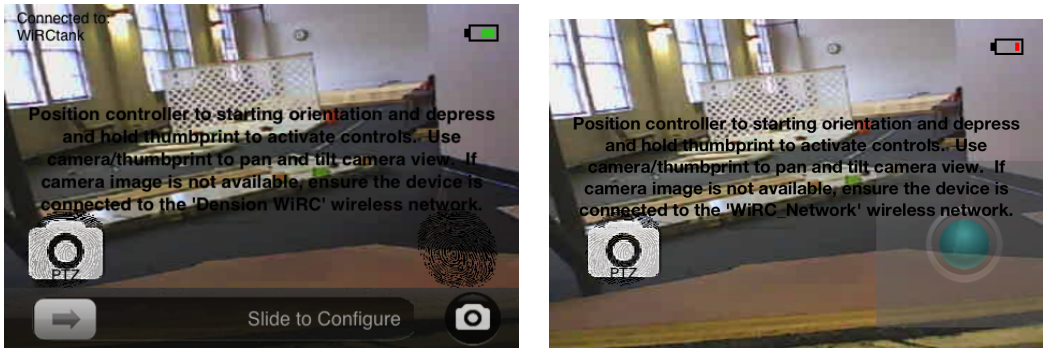
Experiment & Results – Phase 3

7.1 Task

Phase 3 tasks and procedures were identical to Phase 2, introducing only one new level of the independent variable–Controller E (see Figure 7.1). Users were directed to drive the robot, stopping in the scanning boxes to conduct visual identification tasks, and mapping the objects found. They were again instructed to balance speed and accuracy while avoiding collisions. The controller used for training was Controller D, identified as the *default.* The controller opened to this interface for each user, such that all participants started with the same "baseline" configurations.

Users were individually briefed on each aspect of the default configuration and shown the new customizable settings interface, accessed by double tapping the center of the screen and sliding to unlock. The primary task in Phase 3 was establishing a personalized controller using the settings available to each user. This was intended to measure the suitability of the default controller, while providing insight into the control features most closely tied to individual usability (satisfaction and performance). After approximately 10 minutes of training, users were granted 30 minutes to configure their controller while practicing with

the robot anywhere in the lab space. Participants were encouraged to try all combinations of control modes before making a final decision, and were informed that settings in place at the beginning of the timed trial could not be further modified. Each user's final control configuration became their Controller E.



(a) Controller D: Tilt v2

(b) Controller E: Final Configuration Example

Figure 7.1: Phase 3: Controller E

Within the control interface, an option existed to save user configurations to file for later analysis. These files were used to catalog the controller settings by user, available in Appendix C and summarized in Section 7.6. Like all previous trials, users conducted post-experiment data collection via NASA TLX and web-based questionnaire, this time answering a single survey, as only one trial was executed by each participant in Phase 3.

7.2 Independent/Dependent Variables

The independent variable in Phase 3 was the controller type, D (tilt to drive; tilt-based camera) and E (customized). Controller E's dependent variables were compared to Phase 2's Controller D results, serving as the default against which each customized option is measured. The dependent variables used for statistical analysis were a combination of performance results, formal measures of workload

and usability (NASA TLX and System Usability Scale), and informal measures of user satisfaction collected via survey. All of the dependent variables examined in Phase 3 have been thoroughly introduced in Chapters 5 and 6.

ptime is not used here, as it is not directly comparable between Phases 2 and 3 due to the difference in training approach and the time limits imposed. **ttime, majerror, minerror, pathpts, localization, tlx** and **sus** (listed along the top row of Table 7.1) are drawn from performance and survey measures specific to Controllers D and E, while **move, obstacles, slow, med, hi** and **drive** are user-rated abilities for the tilt-based driving controls from Phase 1. Results of surveys from Controller B's trials were used for comparison, as Phase 2 (Controller D) questionnaires failed to have users rate control features specific to *driving.* The bottom section of Table 7.1 shows the **pan, tilt, looking, scanning** and **usability** rankings for both Controllers D and E, collected in Phases 2 and 3, respectively.

7.3 Hypothesis

The null hypothesis for Phase 3 was the only one in this multi-phase experiment that was written to be rejected. The *alternate* hypothesis is presented below, which adequately summarizes the expected effects of a customizable interface on user performance and satisfaction.

(H.3) *Permitting users to manipulate certain controller settings will lead to* ***more satisfied users who perform better with less errors.***

Note that this hypothesis expects improvement in both performance *and* satisfaction, implying that all (or at least the majority of) dependent variables

should favor Controller E. This was based on two assumptions: 1) that users would be reluctant to accept tilt-based controls and would require some measure of personalization to become invested in the outcome, and 2) that users understood, to some degree, their own weaknesses at the tasks assigned and would make intelligent choices intended to improve performance to the greatest extent possible. Assumption #1 inspired the custom controller used in Phase 3, although results from previous phases imply that users were not as reluctant to accept tilt controls as originally expected. Regardless, users were heavily encouraged to experiment with the custom controller options, "forcing" users to consider modifications that they might not otherwise have been motivated to explore. Doing so gave them not only a better understanding of the controls, but also a greater sense of ownership over their controller, a posited contributor to performance/task completion. Assumption #2 is yet to be fully tested; results from Phases 1 and 2 imply that users are only partly aware of their own performances, given that several participants preferred controllers that did not yield the best results. That said, preference and performance were not always expected to be monotonically related, and the results of this phase should provide further evidence to clarify the validity of that expectation. Will users prefer customized controls over a default, regardless of which yielded the better quantifiable outcome, or will the two be closely related?

7.4 Results & Discussion: The F-Statistic

Analysis of variance was once again used as the statistical test to examine the effect of the independent variable on the 18 dependent variables used in this phase. **ttime** was the only performance metric that proved significant, but five of six

"ability" ratings (**obstacles, slow, med, hi** and **drive**) achieved significance as well.

Table 7.1: The F-Statistic: Phase 3

		ttime	**majerror**	**minerror**	**pathpts**	**localization**	**tlx**	**sus**
Controller D	*mean* ($\bar{x}$)	556.56	0.12	3.12	26.08	6.64	46.07	75.1
	std dev (s_d)	124.74	0.33	1.99	3.7	1.95	14.73	13.12
Controller E	*mean* ($\bar{x}$)	469.40	0.08	3.6	26.96	6.28	41.76	77.2
	std dev (s_d)	131.55	0.28	3.04	3.78	2.28	15.52	11.78
	$p(< 0.05)$	*0.0089*	0.5743	0.3680	0.1910	0.4165	0.1534	0.3050
	F	8.11	0.32	0.84	1.81	0.68	2.17	1.10
		move	**obstacles**	**slow**	**med**	**hi**	**drive**	
Controller B	*mean* ($\bar{x}$)	3.52	3.08	3.68	2.80	1.88	3.20	
	std dev (s_d)	0.96	0.95	1.14	1.08	0.83	1.04	
Controller E	*mean* ($\bar{x}$)	3.52	3.60	4.16	3.46	2.60	3.68	
	std dev (s_d)	1.05	0.82	0.62	1.08	1.15	0.75	
	$p(< 0.05)$	1.0000	*0.0344*	*0.0429*	*0.0102*	*0.0042*	*0.0308*	
	F	0.00	5.03	4.57	7.83	10.02	5.27	
		pan	**tilt**	**looking**	**scanning**	**usability**		
Controller D	*mean* ($\bar{x}$)	4.12	4.21	4.00	3.96	4.04		
	std dev(s_d)	0.78	0.83	0.82	0.93	0.86		
Controller E	*mean* ($\bar{x}$)	4.00	4.20	4.24	4.20	4.04		
	std dev (s_d)	1.25	0.91	0.83	1.00	0.61		
	$p(< 0.05)$	0.6406	1.0000	0.2071	0.2471	1.0000		
	F	0.22	0.00	1.68	1.41	0.00		

7.4.1 Performance

Controller type did not prove to have a significant effect on any performance metric outside of trial time. Users improved, on average, 16% between Phases 2 and 3, with an average trial time of 469.4 seconds. Variance in **majerror, minerror** and **pathpts** was too small to result in significance, but none illustrated that Controller E had any detrimental effects on overall performance. In fact, user-rated driving abilities improved significantly from Phase 1, indicating that users felt more capable with Controller E. Learning effects cannot be entirely discounted, as the experiment relied on their existence to incrementally increase the task complexity; however, such large improvements imply at least some contribution on behalf of the controllers assessed. The statistical significance

of the improvement in **med, hi**, and **drive** are especially notable given what was learned in previous trials reference users' struggle to maintain control at the highest throttle speeds. This newly discovered sense of control is almost entirely due to the user's ability to manipulate sensitivity settings, bringing actual throttle behavior in line with individual expectations. This will be explored in more detail in Section 7.6. The results already presented are enough to *reject the null hypothesis*, although whether or not (H.3) is upheld is best left for later discussion.

7.5 Results & Discussion: Correlation

Spearman's correlation coefficient (ρ) was used to ascertain the relationships between the 18 variables in Phase 3. Table 7.2 depicts these coefficients. Given the large number of relationships to examine, usability metrics (i.e. rated abilities) were not compared to one another in the table, as just like in previous phases, they were all closely related. Instead, the table focuses on the relationships between usability and performance.

Table 7.2: Spearman's Correlation Coefficient (ρ): Phase 3

	1	2	3	4	5	6	7	8
1. ttime	——-							
2. majerror	0.2009	——-						
3. minerror	0.1762	0.1409	——-					
4. pathpts	-0.2351	-0.14395	-0.28894	——-				
5. localization	-0.2284	-0.27683	-0.1444	0.2342	——-			
6. tlx	*0.51934*	-0.0947	0.0687	-0.38364	-0.164	——-		
7. sus	-0.1635	-0.0602	0.2245	-0.0102	0.34688	-0.38820	——-	
8. usability	-0.0518	0.2786	0.1248	0.2156	0.1926	*-0.4906*	*0.5499*	——-
	1	2	3	4	5	6	7	8
9. move	-0.2323	0.0633	-0.2293	*0.42554*	0.30025	-0.38756	0.223	0.3206
10. obstacles	-0.2144	-0.0075	-0.1484	0.1542	0.1326	-0.1835	0.1146	0.1494
11. slow	0.0273	0.0719	0.0117	-0.0945	0.29637	-0.0288	0.28074	0.3258
12. med	-0.18592	0.015	-0.2106	0.2684	0.2201	-0.1589	-0.0356	0.034
13. hi	-0.2540	0.1132	-0.2135	0.36808	0.2565	-0.277	0.0399	0.1421
14. drive	-0.31744	0.1033	-0.1015	0.29245	0.2212	-0.35486	0.32329	0.4228
15. pan	-0.0939	0.3719	0.00623	0.0871	0.1179	-0.31076	0.2324	0.3185
16. tilt	-0.18522	0.31969	0.1373	0.1652	0.1244	*-0.45568*	0.198	0.4183
17. looking	-0.3197	0.31095	0.0925	0.1496	0.0367	-0.28002	0.1824	0.4327
18. scanning	-0.1640	0.40268	0.1127	-0.0759	0.0175	-0.2601	0.2094	0.2651

Fewer significant relationships existed in Phase 3 than in earlier phases; however, the notable associations present solidify those previously discussed. System Usability Scores and NASA TLX ratings were negatively correlated, indicating yet again that as usability improves, mental workload decreases. NASA TLX results were also correlated with trial time, indicating a strong positive relationship ($\rho = 0.51934$) between that performance metric and perceived mental demand (see Figure 7.2). Results in Phase 2 indicated a similar relationship. This demonstrates the expected effect where users who struggle with the task (either driving or visual identification), resulting in higher total trial times, are also likely to feel more mentally taxed and frustrated. Overall workload scores decreased from approximately 54 to 45 to 43 over the course of the three experiment phases, but differences *within* phases still indicated an effect on behalf of the independent variable. A slightly weaker association exists between **tlx** and **pathpts** ($\rho = -0.38364$) but supports the observation already made indicating performance's effect on workload; in this case, as the path point variable decreases (less driving accuracy), workload measures increase.

The relationship between **pathpts** and **move** ($\rho = 0.42554$) provides evidence that the control feature most affecting a user's driving accuracy is, unsurprisingly, their ability to "move [the robot] in the correct direction." This relationship did *not* exist in Phase 1, likely due to the lack of variance overall in the path point measure. Users in Phase 1 were focused on a driving only task, with many prioritizing accuracy in their trials. When visual identification tasks were added, users had to re-prioritize their efforts due to task complexity. This increased variance in the **pathpts** term, leading to stronger statistical associations.

All of the variables describing user-rated driving abilities were closely related, especially **move** and **drive** ($\rho = 0.73072$). Previous phases noted the same

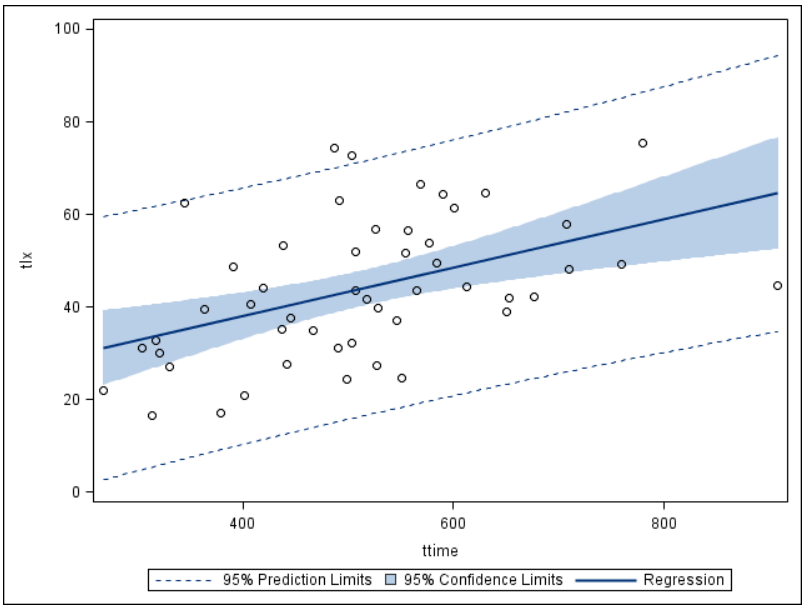


Figure 7.2: Spearman's Correlation Plot:
ttime vs. **tlx**

correlations. Abilities regarding camera manipulation were likewise closely related, although none were individually related to any variables representing performance or usability. **pan** and **tilt** were associated at $\rho = 0.81399$, and **pan** and **scanning** at $\rho = 0.73178$. Unlike Phase 2, where **scanning** informed usability, Phase 3 results showed no such effect. This is likely due to the change of trial focus; in Phase 2, users were closely analyzing the camera controls as the "new" control feature, whereas in Phase 3 users re-focused on driving controls (which many spent the bulk of their time customizing). The only variables even weakly associated with usability (**sus**) were **drive** ($\rho = 0.32329$) and **slow** ($\rho = 0.28074$), like in Phase 1.

In all, the independent variables in Phase 3 appear to have a less statistically significant effect over *performance* and a clear effect over *user satisfaction*, especially with regards to specific driving characteristics. Performance leveled

off, with users performing at approximately the same standard in both Phase 2 and 3 trials; in contrast, satisfaction increased measurably with the addition of the customizable interface. This has implications that go beyond optimizing controls for performance, begging the question of how much one should design instead for satisfaction.

7.6 Customization

A customizable control interface was presented for user consideration with the intent of identifying the existence/strength of three main effects: 1) how satisfaction related to performance, 2) which configurations were chosen most often and how initial preferences informed those designs, and 3) the suitability of controller defaults. These will each be addressed in turn.

7.6.1 Effect of Satisfaction on Performance

Several publications cite user satisfaction as a key contributor in performance [45, 44], yet other studies have found no evidence to support that claim [65]. Such a relationship is closely tied to the experiment protocols and metrics used, so it comes as no surprise that there remains some question about just how strongly (if at all), and in what systems, satisfaction and performance are correlated. The hypothesis (H.3) posited a positive association in this study, with both satisfaction and performance expected to improve with the introduction of user-customizable control features.

While results thus far show weak relationships between some measures of performance and individual measures of usability/satisfaction, a new approach was necessary to achieve a cumulative picture of performance vs. usability.

Nielsen [85] proposed a method for combining usability and performance metrics and recoding raw numbers to a uniform scale permitting comparison. Three variables were selected to represent performance (**ttime, minerror** and **pathpts**), and three to represent satisfaction (**drive, scanning** and **sus**). These were chosen based on relationships exhibited in Phases 1 and 2, and offered the best representative summary of satisfaction and performance given the dependent variables. Data for Controllers D and E were examined, with each value of the dependent variable being restated in terms of *standard deviations from the mean*. This was done by subtracting the individual value from the controller's mean and then dividing by the standard deviation for each factor. Negative values imply poor performance or lower satisfaction and signs were adjusted to ensure that all metrics were consistently scaled, accounting for several measures where lower numbers were desirable. The resulting values (in terms of standard deviations) were then summed to a *performance factor* and a *satisfaction factor*, plotted against one another in Figure 7.3.

When reading the graph, points to the right of the y-axis represent users who performed better than average with the given controller; points to the left represent user-controller pairings that yielded worse than average performance. Similarly, dots above the x-axis represent better than average user satisfaction with the given controller, while dots below that axis indicate lower satisfaction levels. Controllers D and E are represented by blue and red series, respectively, allowing analysis of both the *overall* relationships between performance and user satisfaction, as well as the controller-dependent relationships.

The r^2 goodness of fit measure (from 0-1) for both Controllers D and E are near zero, implying a lack of association between performance and satisfaction in this data. What slight relationship does exist (illustrated via trend lines

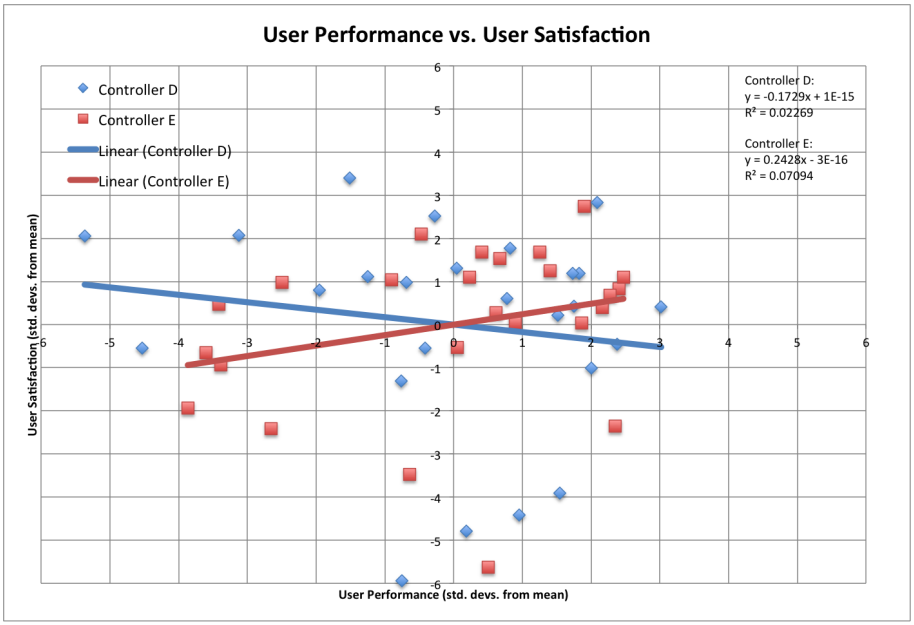


Figure 7.3: User Performance vs. User Satisfaction

in Figure 7.3) is not even consistent between the two controller conditions. Controller D exhibits a slightly negative association, while Controller E shows a slightly positive relationship. Statistical results presented in Section 7.4.1 led to *rejection of the null hypothesis*, but examination of user performance versus satisfaction does **not** provide the evidence necessary to *accept the alternate hypothesis*, as those factors appear to be independent of one another. While this does not mean that performance and satisfaction cannot improve together, it also does not assure it. Phase 3 illustrates conditions under which individual measures of performance and satisfaction have improved but, on their own, are not enough to conclusively accept the hypothesis, (H.3).

7.6.2 Controller E Configurations vs. Preferences

As already discussed, Controller E was not a single controller configuration,

but rather individual variations of the default, Controller D. On average, users changed 3.5 settings (of 10 available), and only one individual made no changes at all. That individual struggled with the tasks throughout the phases, and wisely chose to use the 30 minutes of hands-on practice to continue refining his/her tele-operation skills rather than convoluting performance by making changes to a controller with which he/she was not yet competent. This example indicates that personalization is best implemented by more confident and experienced users, as presenting an inexperienced user with too many choices may be overwhelming. While the controller was designed to be simple and intuitive in an effort to reduce training time and improve ease of use, experiences in Phase 3 indicate that personalized settings for complex controls are best executed over time and under the guidance of an experienced operator or instructor. Chapter 8 will present several recommendations for controller improvements which could make the system less training-intensive, but customizing controls will likely need to be done in tandem with some type of guided assistance (whether it be automated, e.g. adaptive, or trainer-led).

Users were asked to identify in their post-experiment survey the two most important customizable features used to build their controller. Complete responses can be found in Appendix C; however, the majority indicated that *sensitivity* as well as control *mode* were the most important to their design efforts. Depending on which control modes were active, sensitivity was represented by *acceleration scaling* and either *joystick size* or *responsiveness values*. Table 7.3 shows the average user-selected value for each setting, as well as how frequently each was changed. Since not all settings applied to all control modes, # times changed denotes the number of controllers (of 25) where that specific feature *did* differ from the default, out of the number of controllers *eligible* to have done so.

Table 7.3: Controller E Configurations and Frequency of Customization

	% Tilt	% Joystick	Value	# times changed
Camera mode	80	20		5 of 25
Driving mode	60	40		10 of 25
Joystick scale			1.178	12 of 14
Accel scale			0.293	21 of 25
Responsiveness straight			0.382	18 of 23
Responsiveness turn			0.38	19 of 23
# of settings changed	3.52			

Driving Mode

60% of users opted to use tilt controls for the driving mode on Controller E. This is in line with the 64% of users who preferred tilt-based driving controls in Phase 1; however, it does not indicate that those percentages are comprised of the *same* users. In fact, four of the nine users who originally preferred joystick driving controls ended up using tilt controls in Phase 3. This was likely due to extensive practice and exposure to the tilt-based driving controls (in Controller B, C, and D), as well as their ability to now manipulate sensitivity settings. All four users decreased acceleration scaling and manipulated responsiveness values, seemingly improving their control experience to a point where it was then preferred over the previously favored joystick.

The remaining 40% of users drove using the virtual joystick, only half of whom had declared that their preference in Phase 1. The remaining five users presumably found the joystick easier to use in Phase 3 after manipulating the acceleration scaling and joystick size, with most increasing the thumbstick by up to 50%. While some users experienced noticeable frustrations with specific control types, most were able to drive competently with any controller version after limited training. That said, several users were outspoken against tilt controls

due to their own gaming experience predisposing them to joysticks instead. They reluctantly selected the tilt controls in Phase 1 due to their superior presentation and smoother operation; however, once options like joystick size and sensitivity were made available, the virtual joystick again became their control mode of choice.

These effects at least partially explain why users may have changed their mind (as 56% of them did). Initial preferences were clearly not a strong indicator of actual preferences over time, especially following the addition of options rated most important to users e.g. sensitivity scaling. No pre-study measure exists regarding a user's perceived preference, but it would be interesting to examine whether those could more accurately predict final control configurations than the Phase 1 preferences used here.

Camera Mode

In Phase 2, users favored the virtual joystick controls (14 of 25) for camera manipulation, despite grumblings about the camera moving too quickly. In Phase 3, only five individuals opted to use the virtual joystick for camera control! This is almost surely due to the chosen defaults. Users prioritized customization of driving controls: mode, scaling, and responsiveness. Doing so required substantial time and repetition, leaving little time or energy for customizing camera controls. Given the relative simplicity of the scanning task, users likely felt their time was better served perfecting the driving controls. Camera manipulation proved simpler for most, as users averaged just 173 seconds of practice with camera controls in Phase 2, compared to 668 seconds with driving controls in Phase 1. If the default controller had utilized a joystick-based camera, the results would likely be reversed, with 20 users opting not to change the camera channel mode.

7.6.3 Suitability of Controller Defaults

Table 7.3 indicates the mean value of the control options most frequently changed. Comparing those to the original default values should provide some evidence of which, if any, should be updated for future controller versions. As described in Chapter 3, a combination of pilot experiments and best practices drove the adoption of original controller defaults; however, user testing is the preferred method to determine such values. While most users made only minor adjustments, even these small modifications affected their sense of control. Table 7.4 shows how the defaults stacked up against the final user-manipulated configuration values.

Table 7.4: Comparing User-Adjusted Configuration Values to Controller Defaults

	Default	**Modified Mean**
Joystick scale	1	1.178
Accel scale	0.4	0.293
Responsiveness straight	0.45	0.382
Responsiveness turn	0.45	0.38

As previously discussed, joystick size appears to have made the virtual joystick more usable, encouraging users to adopt it for driving more frequently than expected. The original 75 x 75 pixel joystick was increased to approximately 125% of that size (or 94 x 94 pixels). Apple's *Human Interface Guidelines* gave only vague guidance on button size, encouraging designers to make buttons at least 40 x 40 pixels to accommodate an average adult finger [6]. While a button that small may suffice for simple touch interfaces, it is clearly inadequate for objects which must be manipulated (e.g. touched and dragged), even moreso when considering that users frequently used their thumbs in landscape mode. While consideration must be given to how much of the screen a joystick obscures, ease of operation

dictates that it must be *larger* than the finger manipulating it. Therefore, virtual joysticks used in future control iterations should be approximately 100 x 100 pixels, decreasing opacity to permit easier viewing of the device screen when controls are not active.

Arguably the most important user setting, acceleration scaling, affected the robot's top throttle speed. Most users scaled this down further from 40% of the robot's capable speed to 30%. That value is best suited towards the specific conditions in this experiment: indoors, tight spaces, relatively level. That said, this value would likely differ significantly for robot missions executed outdoors, in larger spaces, and/or where driving precision was less important. This setting might even have reason to be adjusted "on the fly," as conditions warrant. Currently the settings interface is not conducive to quick changes; however, several users recommended something like a **turbo** setting be integrated onto the control interface. For example, some users preferred to operate the robot at very low speeds (10% of throttle), but found that at those speeds the robot could not easily climb the ramp to begin the course. In those cases, users found themselves wanting to temporarily increase acceleration sensitivity, reverting to 0.10 upon clearing the ramp. This could be easily accomplished via a turbo button which might ramp up the robot's throttle to clear an obstacle. Implementations could include a set increase for a set amount of time, or a set increase for a variable amount of time (based on length of touch).

Responsiveness settings were another key component of controller sensitivity, although observations imply that users were not always aware of their exact role in that relationship. Essentially, responsiveness was inversely proportional to the degree of tilt, or device range of motion, along a particular axis. Users that wanted responsive controls could reduce the device range of motion, whereas

users that wanted less responsive controls could increase the range of motion. The limits in place on tilt and rotation (min: 0.10, max: 0.6) were intended to prevent users from manipulating the controller in a way that would make video viewing unmanageable. Responsiveness is best considered in combination with acceleration sensitivity, because one affects the other. For instance, if a user scales acceleration way down and degree of tilt way up, the deadspace at controller neutral increases, requiring a larger input motion to initiate movement than if acceleration and responsiveness remained closer to defaults. That said, considering an average acceleration sensitivity of 0.30, responsiveness in both directions (tilt and rotation) should be approximately 0.38 given user inputs. Participants clearly preferred a slightly more responsive controller than was originally provided, correcting some complaints regarding excess deadspace in early versions of the controls.

7.6.4 Controller Operation with Gloved Hands

Given the military problem inspiring this experiment, the controller was designed with gloved users in mind. It was hypothesized that gloved users would prefer tilt-based, gestural inputs primarily due to the proprioceptive nature of their feedback. Gloved hands are larger and bulkier, with gloves limiting dexterity in individual digits–both issues that could hinder effective touchscreen control. In order to test this hypothesis, at the end of Phase 3 users were asked to don a pair of touchscreen-capable utility gloves. The first two fingers and thumb of each glove were equipped with conductive tips (see Figure 7.4). Following configuration, trials, and the post-experiment questionnaire for Controller E, users were granted 10 minutes to re-configure their controls under the new, gloved

operating condition. Any resulting changes to the configurations were saved for analysis.

Figure 7.4: Touchscreen Capable Military Work Gloves
Youngstown Glove Company's Military Work Glove (MWG)
[Online] at `www.ytgloves.com`

Surprisingly, users modified very little! Of the ten users whose Controller E included virtual joysticks to drive, only three of them converted to tilt-based controls while wearing gloves. One individual changed the location of both their driving and camera deadman switches to accommodate thicker fingers. This particular participant had built a controller where both controls were stacked on the right side of the display. When wearing gloves, the user felt he/she had better control when inputs were further separated, and reverted back to the default layout where the camera switch was on the bottom left and the driving switch on the bottom right.

Only one user chose a controller with a hard button deadman switch–a configuration selected for the official trial–meaning no changes were made when gloves were added. Many other users considered the hard buttons as deadman switches, but complained about their location on the device. With the phone

in landscape left orientation (home button to the left), the volume buttons are accessible at the top of the phone's bezel just right of center. Users found it awkward to reach for these and complained further that the thick, protective Otterbox case made manipulation more physically demanding. The hard button option may have received more positive reviews had it utilized the home button or a tactile peripheral in a more ergonomic location.

Discussions with participants indicated that, while tilt-based controls would seem a more natural choice for gloved control, user preference had more bearing in dictating the lack of changes observed. Users invested significant time in learning the nuances of the robot and controls over the first two phases, and many looked forward to making the changes and adjustments afforded in Phase 3. After spending 40+ minutes examining the settings interface and configuring the controls, users were truly invested (and satisfied) with their final setup. Even a major environment change, like adding gloves, was not enough to convince them to make wide-scale changes. Instead, they opted to *adapt* to the new conditions, using practice to overcome the added difficulties of poor dexterity and imprecise pointing precision. While unexpected, it does imply a strong sense of ownership over user-selected features. Several users indicated that adapting to the new conditions was perceived to require less effort than modifying controller settings to values/modes with which they were not comfortable.

7.7 Summary

Phase 3 results encompassed not only the performance and usability metrics analyzed in Phases 1 and 2, but also provided data regarding customization of control and its effect on both. ANOVA produced a statistically significant

effect on trial time, as users completed the course with Controller E (customized) notably faster than with Controller D (default). User-rated abilities on five of six driving characteristics also proved significant, favoring Controller E. Combined, these outcomes were enough to *reject the null hypothesis*.

The alternate hypothesis that this phase aimed to prove anticipated improvement in satisfaction alongside performance. Two factors were derived which combine several dependent variables into comprehensive representations of performance and satisfaction. By plotting these two factors in terms of standard deviations from the mean, one could determine how closely they were related (via linear regression). Surprisingly, performance and satisfaction proved independent of one another in this experiment, a relationship that did *not conclusively accept the alternate hypothesis, (H.3)*.

In terms of customization, users spent nearly 30 minutes modifying their controls, most often changing *acceleration sensitivity* (23 out of 25 users) and *responsiveness values* (18 of 25 users). Control *mode* was also an important custom feature, with users choosing multiple combinations of tilt and joystick controls. Interestingly, Phase 1 preferences did *not* inform the custom options selected. A number of users diverted from their earlier stated preferences, whether due to learning effects or the availability of specific custom features which made previously unusable controls more usable. The modified custom options, when averaged, provided a new set of controller defaults. Acceleration scaling is recommended to decrease from 0.4 to 0.3, along with both responsiveness values (from 0.45 to 0.38). Joysticks should be resized to 125% of their previous size, or approximately 100 x 100 pixels.

Finally, an exercise asking users to modify their customized controls to accommodate gloved hands yielded interestingly un-varied results. Users, by and

large, made no changes to their controls to account for gloves, finding practice alone enough to overcome the challenges of the new control condition. This implies 1) that gloves and touchscreens are not an impossible combination, and 2) that user investment in custom controls is high. While common sense dictates that proprioceptive controls are more suited for gloved operations, users instead chose to commit to their original custom layout, feeling confident in its use.

Chapter 8

Conclusions & Future Work

8.1 Project Summary

The purpose of the research presented here was to assess the usability of *attitude aware controls* as an alternative to more commonly employed, touch-intensive operator control unit methodologies. The goal was to create a controller prototype which promoted simple, efficient control of small reconnaissance robots used by dismounted ground forces during contingency operations. Smartphones offer a powerful, lightweight, multi-modal platform for such control and have the potential to become powerful battlefield tools if usability issues specific to field operations are addressed. Results indicate that performance gains using attitude aware controls were minimal and inconsistent, i.e. time improvements not always met with improved accuracy or precision. Satisfaction metrics almost universally supported tilt-based controls even when preference and system usability did not. System usability improved dramatically over the course of the experiment, paired with an expected decrease in mental workload, confirming that attitude aware controls are an intuitive and user-friendly smartphone-based control option.

8.1.1 Multi-Phase Usability Experiment

Five levels of the independent variable, controller type, were designed to test the hypotheses presented in Chapters 5, 6, and 7. They were introduced to users over the course of three experiment phases. The first phase aimed to re-create ARL's experiment [96], pitting virtual joystick control against attitude aware controls. Users drove the robot through an indoor obstacle course while performance and satisfaction measures were collected. In Phase 2, to identify the suitability of attitude aware controls for operating multiple robot degrees of freedom, visual identification tasks were added to test users' ability to manipulate a pan/tilt camera in addition to Phase 1 driving tasks. Phase 3 introduced a customizable controller, allowing users to manipulate sensitivity and responsiveness, as well as selecting control modes. Phases were designed to build upon one another over the course of approximately six weeks.

8.1.2 Experimental Findings

Analysis of variance (ANOVA), with controller and subject as factors, was used to identify the effects of the independent variable on each dependent variable at the $p < 0.05$ level. In Phase 1, where users were asked only to drive the robot through the course using joystick and tilt controls, significant effects were observed in trial time and minor errors. While virtual joystick trials resulted in faster overall completion times, users made significantly more errors ($\mathbf{minerror}_{joy}$ = 5.64 vs. $\mathbf{minerror}_{tilt}$ = 4.16), decreasing their perception of control.

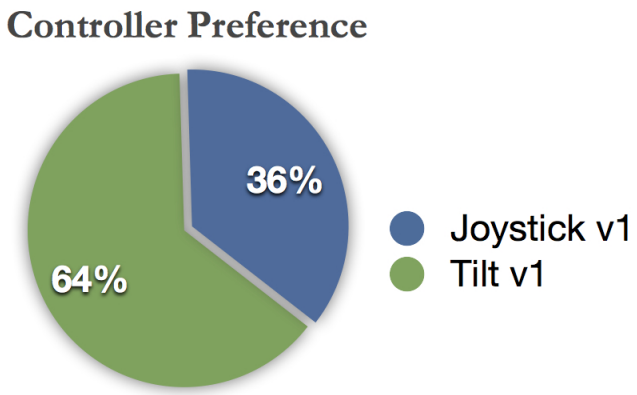
Driving abilities were rated superior for all subtasks (see Figure 8.1b) and exceeded expectations given early assumptions that attitude aware controls might be inferior to virtual joysticks. Comments indicate these preferences were rooted

in the intuitiveness of the tilt controller, with several participants noting its interface was more "natural, smoother, consistent, and familiar. Users exhibited a clear preference for tilt controls, with 64% favoring them over the joystick option (see Figure 8.1a).

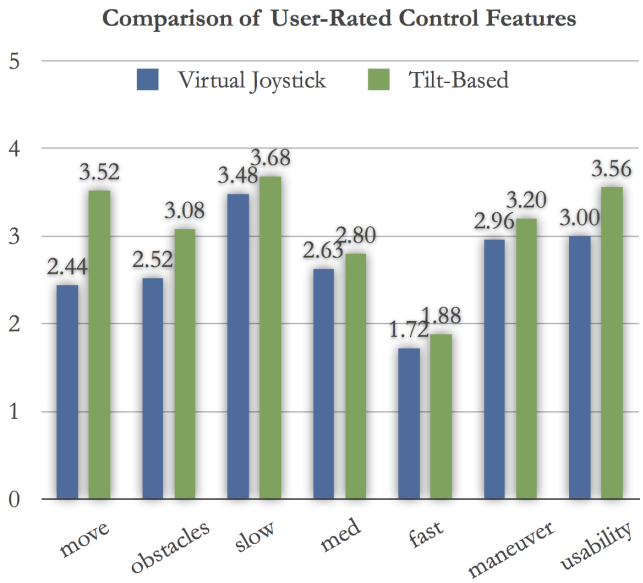
In Phase 2, users added visual identification tasks and took control of the robot's camera, again using both a virtual joystick and attitude aware version of control. ANOVA failed to reject the null hypothesis, implying that both control modes were equally usable and that attitude aware controls are able to provide an interface for both tasks crucial to reconnaissance. Mode confusion was insignificant, likely due to the interface's consistent simplicity, avoiding the use of hierarchical menus and/or screen switching. What little did exist appeared to be related to user pre-study conditions, specifically experience with video and smartphone gaming. Final results implied a preference for the joystick-based camera controls (56%), although 20 of 25 users "preferred" the second controller presented to them in their trials (see Figure 8.2a).

The most interesting results regarding attitude aware control usability came out of Phase 3, where users were given the freedom to customize control modes and sensitivity before conducting both driving and visual identification tasks as before. Figure 8.3 shows that System Usability increased and NASA TLX workload scores decreased, even as tasks grew more complex.

Users spent nearly 30 minutes modifying controller settings, most often manipulating *acceleration sensitivity* (23 out of 25 users) and *responsiveness values* (18 of 23 users). Users indicated that those alone could make either control mode equally **usable**, although individuals maintained strong **preferences**! The pie chart in Figure 8.4 indicates those preferences, depicting the frequency with which each control combination was selected. A plurality of users (48%) chose

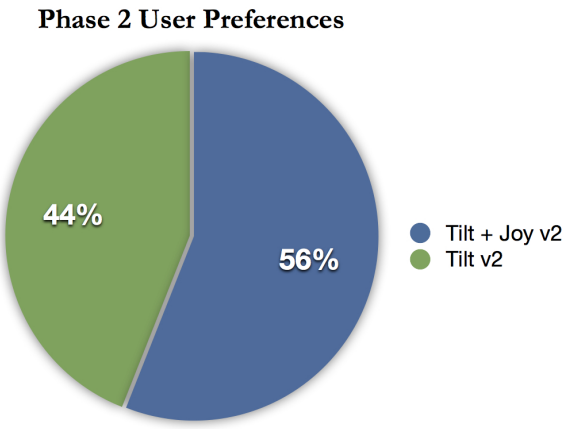


(a) Phase 1 Overall User Preference
(Virtual Joystick Control vs. Tilt Controls)

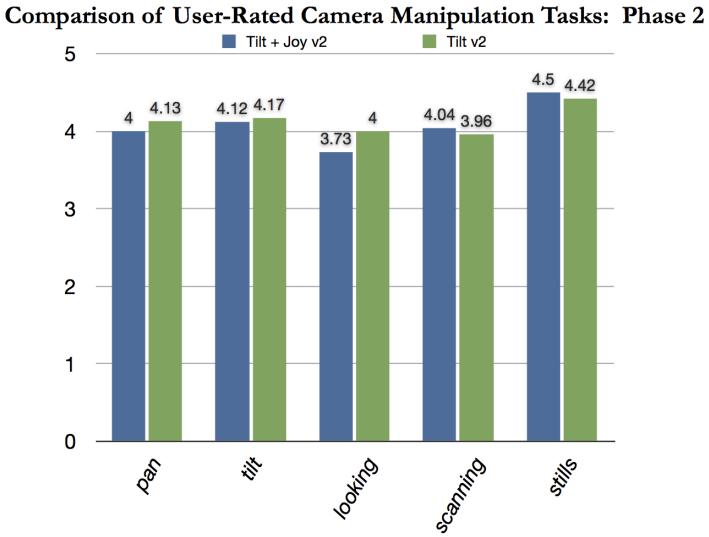


(b) Phase 1 Results Showing User-Rated Driving Abilities
(on a scale of 1-5, with 5 being *extremely easy*).
*Definitions for variables along the x-axis are available in Chapter 5.

Figure 8.1: Phase 1 Results Overview



(a) Phase 2 Overall User Preference
(Joystick Camera Control vs. Tilt Controls)



(b) Phase 2 Results Showing User-Rated Camera Manipulation Abilities
(on a scale of 1-5, with 5 being *extremely easy*).
*Definitions for variables along the x-axis are available in Chapter 6.

Figure 8.2: Phase 2 Results Overview

tilt-based controls for driving *and* camera control, with another 12% choosing them for driving only. These numbers are expected to more heavily favor attitude aware controls in future experiments, using a new pool of novice users, operating in gloves, to better mimic the conditions of military users.

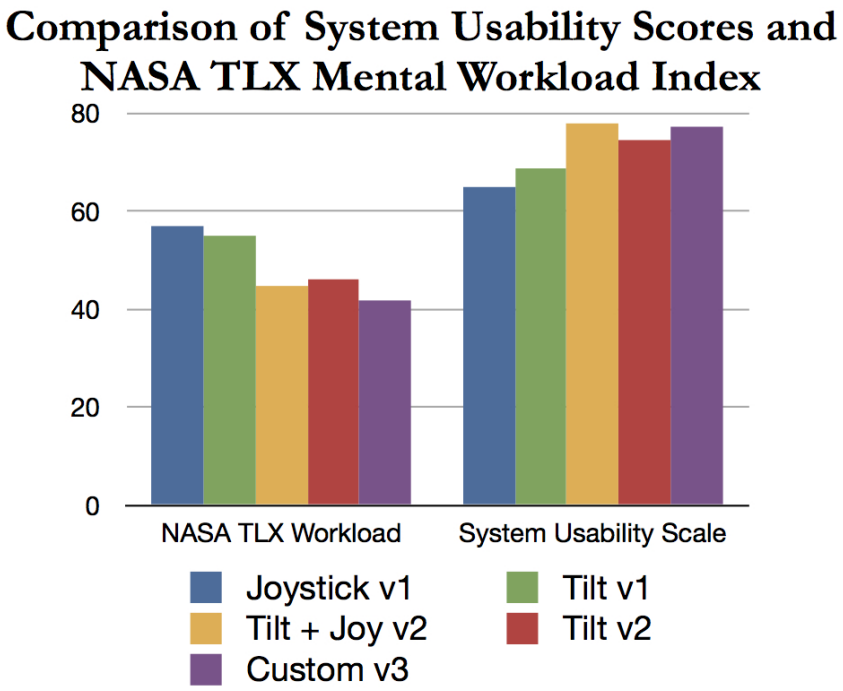


Figure 8.3: Comparison of System Usability Scores and NASA TLX Mental Workload Index Across all Five Levels of the Independent Variable, Controller Type

The preference for tilt-based driving controls (60% of users) is in line with expectations born of results from Phases 1 and 2, where tilt to drive (64%) and joystick camera (56%) controls were most popular. Performance did improve, as customized controls resulted in significantly faster trial times than the default. Users also rated their abilities superior on five of six driving subtasks (indicating improved satisfaction). Additionally, while one might worry that the additional mental effort of customizing controls would yield higher total workload scores,

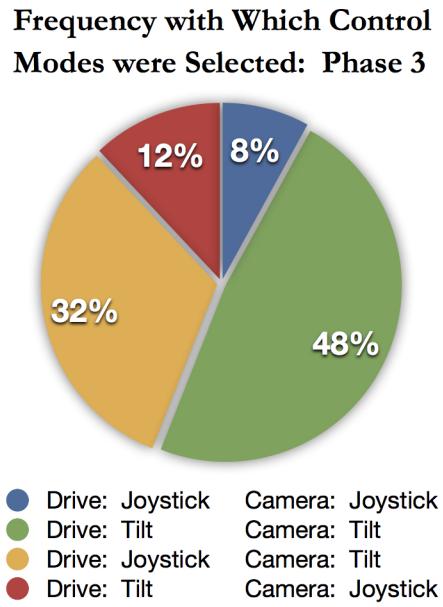


Figure 8.4: User Preferred Control Mode Combinations

that effect was not observed here.

In summarizing these results, analyses were done to examine the role of user preference on performance and, in Phase 3, choice of custom controls. Interestingly, user preference was not consistently related to usability. In Phase 1, a majority of users preferred the attitude aware controller even though average joystick trial times were significantly faster. Phase 2 results indicate that users preferred the joystick for camera control yet rated the tilt-based interface superior in three of five camera subtasks (see Figure 8.2).

Given these discrepancies, it is difficult to discern what, if any, one factor most contributes to usability. It appears as if a combination of factors, including user biases, are responsible for the mismatch between preference and performance; however, certain telling correlations do exist between dependent variables across the three trials. **drive** and **slow** are the maneuver metrics most closely related

to system usability, particularly in the first phase, while **scanning** is the only camera-specific metric which appears to affect usability. Performance-wise, **ttime** and **localization** show moderate associations with usability in Phases 1 and 2 but do not accurately predict preferences in either case.

The sum of these results prove that tilt-based controls are suitable for tele-operating small ground robots and provide superior user experiences (in terms of satisfaction) with better, or equal, performance when compared to virtual joystick controls. Custom control significantly decreases trial times and further improves individual metrics of driving satisfaction, while overall system usability remains high. Additionally, these controls successfully overcome a number of issues regarding touchscreens in field environments; they can eliminate the need for touchscreen interaction during operations, provide a control mapping that is intuitive to users (steering wheel), reduce mental workload, improve usability, and are hosted on adaptable systems significantly smaller than the OCUs currently deployed. This provides strong evidence to suggest that **attitude aware controls (with adjustable sensitivity) are superior smartphone-based control implementations for tele-operated ground robots, suitable for both maneuver and camera manipulation tasks.** As the military expands smartphone use, applications in robotics should absolutely be considered, as indicated by the success of the glove friendly control application presented here.

8.2 Future Work

As in any ambitious project, even after three phases of user research and countless control iterations, work still remains. The sections below attempt

to identify both areas that need additional *research* and those that need additional *work*. In some cases, changes/modifications to control or experimental approach are recommended and, based on research already conducted, identify tasks to complete, i.e. update controller defaults, add a feature, etc. Other recommendations identify gaps in the current research and proposes future projects which aim to fill them!

8.2.1 Continue Army Research Lab's Experiment

Primarily, this usability experiment proved that better smartphone control implementations do exist, rendering virtual joystick alternatives less desirable. Attitude aware controls with adjustable sensitivity overcome almost all of the user complaints from Pettitt's 2011 experiment (introduced in Chapter 4), including lack of haptic feedback and joystick sensitivity [96]. At the time, the authors had suggested providing larger buttons while incorporating some type of control status indicator, as many users encountered difficulties identifying when the virtual joystick was "engaged." The authors also cited previous research using transparent buttons on the screen to maximize limited screen real estate. The controller prototypes in this research considered all of these issues and recommendations. Haptic feedback is inherent to attitude aware controls, sensitivity and responsiveness are available via custom options, and partially transparent switches and buttons towards the bottom edges of the screen obscure very little of the full-screen display.

ARL's results (comparing virtual joystick control to the XBox 360 controller) overwhelmingly favored the latter, reaching significance *($p < 0.001$)* in mean time to complete the courses, mean number of off course errors, and mean number

of driving errors. Additionally, users reported a higher total workload score with the joystick controller, specifically on the mental, effort, and frustration scales. Finally, participants rated their own performance with the XBox 360-style controller superior to that with the touchscreen joystick, with 26 of 30 participants preferring the former. The question now remains whether the attitude aware controls would stand up better to testing against the XBox 360 controls currently in use with PackBot systems. A natural extension of the experiments already conducted would be a "fourth phase," where customizable controllers are compared directly to XBox 360 controllers under similar tasks and conditions. A key consideration would be how to train users on the more complex customizable system, as newly recruited participants would not have the advantage of a multi-phase "lead-in."

8.2.2 Controller Modifications

Key modifications regarding controller defaults were already presented in Chapter 7. As discussed, *acceleration sensitivity, responsiveness*, and *joystick size* should all be updated to the user-averaged values observed in Phase 3. Such changes would be seamless to future users and would simply effect default controller start points, not the range of values available. Additional controller modifications should address slight deficiencies in visual controller feedback. Originally, a red crosshair was investigated for use with camera controls, to help users track the camera's center point; however, difficulty aligning crosshair movement to camera movement proved more difficult than expected. Image handling techniques, which pinpoint landmarks in the camera space, would be required to appropriately track the camera's servo position with respect to the

device view. Instead, small lights could be implemented along the vertical edges of the screen to inform users when the camera is approaching its left and right limits. By enabling certain features (pinch to zoom) of the `UIView` which hosts the video feedback, livestream images could also be manipulated to aid in surveillance and reconnaissance.

Users were first introduced to the driving lights which provide control status during Phase 1 video-led training; however, users still occasionally encountered confusion regarding the use of the green and red lights in tandem with text reading "driving" or "reverse." A further search of the literature should be conducted to identify better, well-accepted approaches. In the meantime, driving text should include a "stopped" status, and a "turning" status and light should be added to differentiate zero-point turns from driving forward. Given that reverse lights are red and driving lights are green, it would make sense to make turning lights amber.

In addition to the visual feedback updates recommended, use of tactile feedback, via vibration, should also be investigated. Similar to how the XBox 360 shakes when users are shot at during video game play, the iDevices could vibrate when robot bump sensors are depressed. This would require a slight modification to robot hardware, but would ease the burden on users attempting to identify how/why their robot is no longer moving. Ideally vibration could be localized on the device, mapped to the side of the robot stuck; unfortunately, current iPhone hardware includes only a single motor, limiting vibration to all or nothing. As hardware grows more sophisticated, proximity sensors could be used to further refine obstacle detection, manifesting in vibrations of increasing strength as the robot neared an obstacle. Paired with visual indicators, like red flashing lights around the screen's perimeter, this type of feedback is both attention-grabbing

and natural, providing the user cues to an environment he/she can just barely see. The robot's current camera field of view (FOV) is approximately 60°, much less than the 140° available to drivers of automobiles, making users feel as if they are looking through a tube. van Erp suggests a field of view somewhere between the two (100°) for the best tele-operation experience [123]. Designers could experiment with adding a fisheye lens to the existing setup to increase FOV, or switching to a more sophisticated camera.

While van Erp also encourages manual robot camera control as it "enables optimal use of human expertise concerning information gathering tasks" [123], users here recommended adding a feature to *automatically* scan a room, designed such that the camera is guaranteed to cover its entire viewing range. This would not replace manual manipulation, but rather augment it as conditions permit. Similarly, a "turbo" button could be created to provide short bursts of increased acceleration to temporarily compensate for instances where users might have the throttle sensitivity very low. If either, or both, of these macros are added, careful consideration must be given to how and where on the control screen they will be placed. Identifying hand/arm movements that initiate these macros may be more beneficial, and more in line with the attitude-based nature of the controls already developed.

Lastly, the role of robot *trim* was introduced in Chapter 3, where hard-coded values of trim for the two robots used in this study were presented. While setting these values programmatically in a global variables file does work, the ideal approach would provide the user with an adjustable trim interface within the controller. Early development looked at the feasibility of doing so; however, due to the multiple runtime threads handling link establishment, message structure, and error handling, modifying the application to handle adjusting trim in realtime

proved too time consuming given the limited benefits immediately applicable to this work. Adding this feature in the future would make the controller inherently more adaptable, as well as offering yet another form of user customization for those desiring an alternative camera neutral.

8.2.3 Additional Research

Platform Expansion and Simulation

Part of what initially drove this project's use of smartphones was the idea that software-based controls are inherently adaptable and customizable in a way that hardware-based controls are not. Several companies advertise hardware-based "all-in-one" controllers [100, 3, 12, 118, 101], but none are feasibly configured and maintained at operator-level. The advantage of software-based controls hosted on a familiar device is the ease with which new missions and platforms can be accommodated, without requiring the user to contact a contracted representative for support. The current control prototypes were written to accommodate two vehicles, a tracked robot and a 4WD Jeep. Future work should focus on expanding this library of supported vehicles, identifying the control mapping necessary for each. Ideally, an aerial platform would be added, demonstrating that not only is the controller capable of commonality across *platforms*, but also across *domains*. It is unrealistic to think that, in its current form, this controller could accommodate simultaneous control of an air and ground vehicle, but with appropriate improvements to the display and communications links, it might serve as the most readily available way to pair the two in one controller. Doing so would require one of the vehicles to be at least semi-autonomous, but could provide the "exocentric-added" view coveted by many researchers, including Chen [25].

This view provides a split screen with the ground vehicle's display beside the air vehicle's display, providing both an egocentric and exocentric view of a robot's surroundings. Doing so without simultaneous control, i.e. piggybacking off of another user's aerial robot feed, is called *passive control*, sometimes also regarded as hierarchical control. In these instances, one individual **controls** the robot, while one (or many) others, monitors the robot's feed. Implementing common controllers that can be carried by many users increases the opportunity for passive control, improving situational awareness for individuals further from the battlefield.

While usability assessments are most reliable when done under realistic conditions with physical systems and human operators, there comes a time, especially when conducting training and repetitive tasks, to include simulations. This project initially hoped to link the controller prototype to a Linux-based rover simulator, so that prospective users could train in a virtual environment and use guided vignettes to identify appropriate controller settings. Doing so would be invaluable to the military, where there is a push (and budgetary demand) to move towards more virtual reality/simulation-based training. This would be best accomplished by removing the WiRC-specific communications protocols from the current application and creating a generic robot API using Linux sockets, allowing the smartphone-based controller to connect to a variety of robot platforms or server-based simulators.

Communication and Lag

The robot used in this study was modified to connect controller to robot via wireless network, as the smartphones used for development were only packaged with WiFi and Bluetooth connectivity. Unfortunately, WiFi is plagued by

interference and low-powered portable antennas, severely limiting range. Both ad hoc (point to point) and infrastructure networks were tested over the course of this system's development, and neither proved robust. The military has generally avoided wireless technologies for these very reasons, in addition to the difficulty securing wireless communications. Instead, the military relies primarily on radio frequency (RF) communication links in various spectrums: HF, UHF, VHF. This does not mean that smartphones are not still capable of addressing the needs; however, they would have to be modified to do so. At least one company, Applied Research Associates (ARA), is already looking at adding RF transmitters to iPhones and tablets, as seen/discussed at the U.S. Army's Robotics Rodeo 2012 [11].

While waiting for RF technology to catch up with smartphone development, additional research can be conducted to address/mask the communications lag present between controller and robot. Often, there is a significant lag in video feedback (1 - 3s) and a less obvious lag in the controls (300 - 500ms) [32]. Training is generally used to familiarize users with the degree of disconnect between the robot and controller; however, more advanced methods could be investigated. For instance, outgoing controls to the robot could be artificially delayed by an additional 500ms - 1s in order to sync the video feedback to the robot's movement.

Users throughout this study logged several complaints/observations regarding not just video lag, but *variable* video lag. On average these did not exceed 1s; however, combined with brief periods of reduced refresh rate, many users were relegated to operating the robot in a "choppy" manner, mimicking the discontinuity in the video feed. Scaling robot speed based on lag/frame rate is one option to consider which might address this problem. This would not serve as a time delay, but rather a forced throttle governor when the robot encountered

moments of poor video quality. The non-trivial aspect of this approach is measuring system lag accurately (and in a timely manner).

Motion Algorithm Updates

Finally, while extensive time went into developing the sophisticated motion algorithm governing the tilt-based controls currently in use, several updates should be considered. First, controller range of motion was initially designed around comfortable device viewing angles, ensuring that users could always see the screen; little allowance was made for human physiology. Rahman provides an extensive look at wrist-based control interactions implemented to improve comfort and user performance based on the pronation/supination traits of the human wrist. Using quadratic discretization to map wrist range of motion to controls (like described) would likely only slightly alter the current prototype, but could meaningfully improve the user experience [103].

Secondly, approaches to sensor filtering should be re-examined. Jang provides an accelerometer filter which uses both a low-pass filter and a threshold comparison to help recognize gestures [61]. While a complementary (or balanced) filter is used here to address gyro drift, the addition of a threshold value could prove beneficial by explicitly setting the deadspace, or controller neutral zone, which is currently dependent on a combination of several customizable variables. Such filters might also be the key to expanding these controls beyond stationary use, allowing users to operate their robot "on the move," either mounted or dismounted. While current doctrine does not require operators to move while tele-operating due to challenges with situational awareness, security, and concealment, future operations might. The move towards more autonomous systems makes this even more likely, as soldiers will do more multi-tasking.

References

[1] Julie A. Adams. Critical Considerations for Human-Robot Interaction Development. Technical report, Rochester Institute of Technology, 2002.

[2] Paul V. Alpo. Warfare Has Changed, So Should Have Methods. *Complete Guide by Armada: Urban Warfare 2009*, 2009.

[3] AMREL. Common Control, Here and Now. [Online], `http://www.commoncontrolnow.com`, 2010.

[4] John Antal. I Fight the Body Electric! *Military Technology*, 33(7):22 – 30, 2009.

[5] Apple. Object-Oriented Programming with Objective-C: Tools Languages: Objective-C, Nov 2008.

[6] Apple. Apple Human Interface Guidelines, Aug 2009.

[7] Apple. The Objective-C Programming Language, Oct 2009.

[8] Apple. Core Motion Framework Reference. [Online] at `http://developer.apple.com/library/ios/#documentation/CoreMotion/Reference/CoreMotion_Reference/_index.html`, October 2011.

[9] Apple. Event Handling iPhone OS, March 2011.

[10] Applied Research Associates. Pointman: Purpose-Built for Tactical Missions. [Online] at `http://www.ara.com/robotics/Small-Unmanned-Ground-Vehicle.html`, 2011.

[11] Applied Research Associates. Pointman Robot and Modified Samsung Tablet Control via Radio Frequency. Discussion/Demonstration at TARDEC's Robotics Rodeo 2012, Fort Benning, GA, June 2012.

[12] U.S. Army. AN/PSW-2 Common Controller, September 2009.

[13] Michael Barr. Pulse width modulation. *Embedded Systems Programming*, 14(10):103–104, 2001.

[14] Eric Bland. Wii-controlled robots made for combat. *DiscoveryNews on MSNBC.com*, December 2008.

[15] John G. Blitch. Adaptive Mobility for Rescue Robots. In Edward M. Carapezza, editor, *Proceedings of SPIE 5071, Sensors and Command, Control, Communications and Intelligence (C3I) Technologies for Homeland Defense and Law Enforcement II*, volume 5071, 2003.

[16] Jan Blom. Personalization: a taxonomy. In *CHI '00 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '00, pages 313–314, New York, NY, USA, 2000. ACM.

[17] Nick Bobic. Rotating Objects Using Quaternions. [Online] at `http://www.gamasutra.com/view/feature/3278/rotating-objects-using-quaternions.php`.

[18] John Brooke. *SUS: A Quick and Dirty Usability Scale*. Taylor and Francis, 1996.

[19] Jonathan Brown, Chris Blanco, Jeffrey Czerniak, Brian Hoffman, Orin Hoffman, and Amit Juneja. Soldier Experiments and Assessments using SPEAR speech control system for UGVs. In *Proceedings of SPIE, Detection and sensing of mines, explosive objects, and obscured targets XV*. SPIE, 2010.

[20] Steve Bryson. Effects of lag and frame rate on various tracking tasks. In *Proceedings of the International Society for Optical Engineering*, volume 1915, pages 155–166, 1993.

[21] J. Carlson and R. R. Murphy. How UGVs Physically Fail in the Field. *IEEE Transactions on Robotics*, 21(3):423 – 437, 2005.

[22] Jessie Y. C. Chen. UAV-guided navigation for ground robot tele-operation in a military reconnaissance environment. *Ergonomics*, 53(8):940–950, 2010.

[23] Jessie Y. C. Chen, Ellen C. Haas, and Michael J. Barnes. Human Performance Issues and User Interface Design for Teleoperated Robots. In *IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews*, volume 37, November 2007.

[24] Jessie Y. C. Chen and Jennifer E. Thropp. Review of Low Frame Rate Effects on Human Performance. In *IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans*, volume 37, pages 1063–1076, 2007.

[25] T. Chen, Y. Yesilada, and S. Harper. What input errors do you experience? Typing and pointing errors of mobile web users. *International Journal of Human Computer Studies*, 68:138–157, 2010.

[26] Victor Vui-Kiat Chong. Heuristics for Mitigating Mode Confusion in Digital Cameras. Master's thesis, University of Victoria, 2000.

[27] Alistair Cockburn. Structuring use cases with goals: Part 1. *Journal of Object Oriented Programming*, pages 35–40, Sep-Oct 1997.

[28] Alistair Cockburn. Structuring use cases with goals: Part 2. *Journal of Object Oriented Programming*, pages 56–62, Nov-Dec 1997.

[29] Paul R. Cohen. *Empirical Methods for Artificial Intelligence*. The MIT Press, 1995.

[30] Shane Colton. The Balance Filter: A Simple Solution for Integrating Accelerometer and Gyroscope Measurements for a Balancing Platform. Rev. 1: Submitted as a Chief Delphi White Paper, June 2007.

[31] Joe Conway and Aaron Hillegass. *iPhone Programming: The Big Nerd Ranch Guide*. Addison-Wesley Professional, 2010.

[32] M.L. Cummings, K. Jackson, P. Quimby, and D. Pitman. Development and Testing of a Quad Rotor Smartphone Control System for Novice Users. *International Journal of Micro Air Vehicles*, 4(3), September 2012.

[33] James Davis and Christopher Smyth. Mitigating the Effects of Time Lag on Driving Performance. In *Proceedings of the 2009 Ground Vehicle Systems Engineering and Technology Symposium*, 2009.

[34] Dension. WiRC User's Manual v2.0. [Online] at `http://www.wirc.dension.com/support`, 2011.

[35] Dension Audio Systems Kft. Dension WiRC Communication Protocol Specification Version 1.3, 2011.

[36] Dension Audio Systems Kft. WiRC Communication Messages, 2011.

[37] Department of Defense. FY2009-2034 Unmanned Systems Integrated Roadmap. [Online] at `http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA522247`, April 2009.

[38] B. Donmez, P. E. Pina, and M. L. Cummings. Evaluation criteria for human-automation performance metrics. In *Proceedings of the Performance Metrics for Intelligent Systems Workshop*, 2008.

[39] D. Eberly. Quaternion algebra and calculus. *Magic Software, Inc.*, 21, 2002.

[40] William R. Ferrell. Remote Manipulation with Transmission Delay. Technical report, NASA, February 1965.

[41] T Fong, C Thorpe, and C Baur. Multi-robot remote driving with collaborative control. *IEEE Transactions on Industrial Electronics*, 50(4):699 – 704, 2003.

[42] Terrence Fong and Charles Thorpe. Vehicle Teleoperation Interfaces. *Autonomous Robots*, 11:9–18, 2001.

[43] Nicholas Fung. Light weight, portable operator control unit using an Android-enabled mobile phone. In Douglas W. Gage and Charles M. Shoemaker, editors, *Unmanned systems Technology XIII, Proceedings of SPIE*, volume 8045, 2011.

[44] Amy W. Gatian. Is user satisfaction a valid measure of system effectiveness? *Journal of Information and Management*, 26:119–131, 1994.

[45] Maarten Gelderman. The relation between user satisfaction, usage of information systems and performance. *Journal of Information and Management*, 34:11–18, 1998.

[46] Michael A. Goodrich and Jr. Dan R. Olsen. Seven Principles of Efficient Human Robot Interaction. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 3942–3948. IEEE, October 2003.

[47] Michael A. Goodrich, Jr. Dan R. Olsen, Jacob W. Crandall, and Thomas J. Palmer. Experiments in Adjustable Autonomy. In *2001 IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pages 1624–1629. IEEE, 2001.

[48] Michael A. Goodrich and Alan C. Schultz. Human-Robot Interaction: A Survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275, 2007.

[49] Joe Gould. Army to field new network tools to 8 BCTs. [Online] at `http://www.armytimes.com/news/2012/07/army-network-tools-fielding-8-brigade-combat-teams-070312w/`, 2012.

[50] Vicki Haberman. *Designing for Diverse Users–A Case Study on Touchscreen Smartphone Customization*. PhD thesis, Georgia Institute of Technology, 2012.

[51] S.G. Hart and L.E. Staveland. Development of NASA-TLX (Task Load Index: Results of empirical and theoretical research. In P.A. Hancock and N. Meshkati, editors, *Human Mental Workload*, chapter 7, pages 139–183. Elsevier, 1988.

[52] Aaron Hillegass. *Objective-C Programming: The Big Nerd Ranch Guide*. Addison-Wesley Professional, 2011.

[53] Ken Hinckley. The Past and Present Future: Patents. [Online] at `http://kenhinckley.wordpress.com/patents/`, 2012.

[54] Ken Hinckley, Jeff Pierce, Mike Sinclair, and Eric Horvitz. Sensing Techniques for Mobile Interaction. In *Symposium on User Interface Software and Technology, CHI Letters*, volume 2, pages 91–100, 2000.

[55] Ken Hinckley and Daniel Wigdor. *Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications, Third Edition (Human Factors and Ergonomics)*, chapter Input Technologies and Techniques. CRC Press, 2012.

[56] W. H. Huang and E. P. Krotkov. Optimal stereo mast configuration for mobile robots. In *IEEE International Conference on Robotics and Automation*, number 3, pages 1946 – 1951, 1997.

[57] Human Performance Research Group, NASA. NASA Task Load Index (TLX) v 1.0: Pen and Pencil Package. [Online] at `http://humansystems.arc.nasa.gov/groups/TLX/paperpencil.html`, 1988.

[58] International Organization for Standardization (ISO). Ergonomics of human-system interaction - multiple Parts. [Online] at `http://www.iso.org/iso/home/store/catalogue_tc/catalogue_tc_browse.htm?commid=53372`, 2008.

[59] iRobot Corporation. iRobot 110 FirstLook. [Online] at `http://www.irobot.com/us/learn/defense/firstlook.aspx`, 2012.

[60] Edmond Israelski and Arnold M. Lund. The human-computer interaction handbook. chapter The evolution of human-computer interaction during the telecommunications revolution, pages 772–789. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 2003.

[61] Ikjin Jang and Wonbae Park. A Gesture-Based Control for Handheld Devices Using Accelerometer. In *9th Iberoamerican Congress on Pattern Recognition*, volume 3287, pages 259–266, 2004.

[62] Timo Jokela, Netta Iivari, Juha Matero, and Minna Karukka. The standard of user-centered design and the standard definition of usability: analyzing iso 13407 against iso 9241-11. In *Proceedings of the Latin American conference on Human-computer interaction*, CLIHC '03, pages 53–60, New York, NY, USA, 2003. ACM.

[63] Brenden Keyes and Holly A. Yanco. Camera placement and multi-camera fusion for remote robot operation. In *IEEE Int'l Workshop on Safety, Security and Rescue Robotics*, 2006.

[64] Kamran Khan and Kim Hyunwoo. Factors Affecting Consumer Resistance to Innovation: A study of Smartphones. Master's thesis, Jonkoping International Business School, May 2009.

[65] Dae-Jin Kim, Rebekah Hazlett, Heather Godrey, Greta Rucks, David Portee, John Bricout, Tara Cunningham, and Aman Behal. On the Relationship between Autonomy, Performance, and Satisfaction: Lessons from a Three-Week User Study with post-SCI Patients using a Smart 6DOF Assistive Robotic Manipulator. In *2010 International Conference on Robotics and Automation*, pages 217–223. IEEE, 2010.

[66] Desmond King-Hele. Erasmus darwin's improved design for steering carriages–and cars. *Notes and Records of the Royal Society of London*, 56(1):41–62, 2002.

[67] Michael Knox. Interview with CPT Michael Knox, U.S. Army. E-mail Correspondence, December 2012.

[68] J. Corde Lane, Craig R. Carignan, Brook R. Sullivan, David L. Akin, Teresa Hunt, and Rob Cohen. Effects of Time Delay on Telerobotic Control of Neutral Buoyancy Vehicles. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*. IEEE, 2002.

[69] Axel Lankenau. Avoiding Mode Confusion in Service Robots - The Bremen Autonomous Wheelchair as an Example. In *Proc. of the 7th Int. Conf. on Rehabilitation Robotics (ICORR 2001)*, pages 162–167, 2001.

[70] Chen Ling, Wanil Hwong, and Gavriel Savendy. A survey of what customers want. *Behaviour Information Technology*, 26(2):149–163, March-April 2007.

[71] Matthew Loper, Nathan Koenig, Sonia Chernova, Chris Jones, and Odest Jenkins. Mobile Human-Robot Teaming with Environmental Tolerance. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 157–164. ACM, 2009.

[72] Wendy E. Mackay. Patterns of sharing customizable software. In *Proceedings of the 1990 ACM conference on Computer-supported cooperative work*, pages 209–221. ACM, 1990.

[73] Wendy E. Mackay. *Users and Customizable Software: A Co-Adaptive Phenomenon*. PhD thesis, Massachusetts Institute of Technology, May 1990.

[74] Wendy E Mackay. Triggers and barriers to customizing software. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, pages 153–160. ACM, 1991.

[75] Douglas E. McGovern. Experiences in teleoperation of land vehicles. Technical report, Sandia National Labs., Albuquerque, NM (USA), 1987.

[76] Steve McKillup. *Statistics Explained: An Introductory Guide for Life Scientists*. Cambridge University Press, 2011.

[77] Mark Milian. U.S. Army may soon equip troops with smartphones. [Online] at `http://www.cnn.com/2011/TECH/mobile/07/12/army.smartphones/index.html?hpt=hp_t2`, July 2011.

[78] David P. Miller and Kyle Machulis. Visual aids for lunar rover tele-operation. In *Proceedings of the 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, September 2005.

[79] Jack Morrow. Interview with CPT Jack Morrow, U.S. Army. E-mail Correspondence, December 2012.

[80] Florian Müller. iPhone-Robot Command Interface: A Case Study. Master's thesis, Eidgenössische Technische Hochschule Zürich, 2009.

[81] Robin R. Murphy. *Introduction to AI Robotics*. The MIT Press, 2000.

[82] Robin R. Murphy. Human-Robot Interaction in Rescue Robotics. In *IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews*, volume 34, May 2004.

[83] Jerome L. Myers and Arnold D. Well. *Research Design and Statistical Analysis*. Lawrence Erlbaum Associates, 2nd edition, 2003.

[84] Hoa G Nguyen and John P Bott. Robotics for law enforcement: Applications beyond explosive ordnance disposal. In *SPIE International Symposium on Law Enforcement Technologies*, 2000.

[85] Jakob Nielsen. User Satisfaction vs. Performance Metrics. Electronic article, Nielsen Norman Group, [Online] at `http://www.nngroup.com/articles/satisfaction-vs-performance-metrics/`, October 2012.

[86] Donald A. Norman. *The Design of Everyday Things*. Basic books, 2002.

[87] Donald A. Norman. *The Design of Future Things*. Basic Books, 2007.

[88] Kristian Nymoen, Arve Voldsund, and Ståle A. Skogstad. Comparing Motion Data from an iPod Touch to an Optical Infrared Marker-Based Motion Capture System. In *The 12th International Conference on New Interfaces for Musical Expression*, 2012.

[89] Ian Oakley and Sile O'Modhrain. Tilt to scroll: evaluating a motion based vibrotactile mobile interface. In *Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005. World Haptics 2005. First Joint*, pages 40–49. IEEE, 2005.

[90] Alex Okafor. Parade of Rain Blog. [Online] at `http://www.paradeofrain.com/2010/07/lessons-learned-in-tilt-controls/`, July 2010.

[91] Dan R. Olsen and Michael A. Goodrich. Metrics for Evaluating Human-Robot Interactions. In *In Proc. NIST Performance Metrics for Intelligent Systems Workshop (2003) Key: citeulike:9886596 In Proc. NIST Performance Metrics for Intelligent Systems Workshop (2003)*, 2003.

[92] Orbotix. iOS Developer Quick Start Guide. ReadMe in Open Source SDK, `https://github.com/orbotix/Sphero-iOS-SDK`, 2012.

[93] Stanely R. Page, Todd J. Johnsgard, Uhl Albert, and C. Dennis Allen. User Customization of a Word Processor. In *Computer Human Interaction 1996 (CHI '96)*, 1996.

[94] Nick Pannuto and CJ Hanson. Sneakyjoystick. Open Source, `https://github.com/cjhanson/SneakyJoystick`, 2010.

[95] Boeun Park, Scott Song, Joonhwan Kim, and Wanje Park Hyunkook Jang. User customization methods based on mental models: modular ui optimized for customizing in handheld device. *Human-Computer Interaction. Interaction Platforms and Techniques*, pages 445–451, 2007.

[96] Rodger A. Pettitt, Elizabeth S. Redden, Nicholas Fung, Christian B. Carstens, and David Baran. Scalability of Robotic Controllers: An Evaluation of Controller Options–Experiment II. Technical Report 5776, Army Research Laboratory, September 2011.

[97] Stephane Piskorski, Nicolas Brulez, and Pierre Eline. AR.Drone Developer Guide SDK 1.7. [Online] at `https://projects.ardrone.org`, May 2011.

[98] David Pitman. Collaborative Micro Aerial Vehicle Exploration of Outdoor Environments. Master's thesis, Massachusetts Institute of Technology, February 2010.

[99] David Pitman and Mary L. Cummings. Collaborative Exploration with a Micro Aerial Vehicle: A Novel Interaction Method for Controlling a MAV with a Hand-Held Device. *Advances in Human-Computer Interaction*, 2012.

[100] D. Powell, G. Gilbreath, and M. Bruch. Multi-robot operator control unit for unmanned systems. *Defense Tech Briefs*, April 2008.

[101] QinetiQ North America. USMC Tactical Robotic Controller (TRC). [Online] at `https://www.qinetiq-na.com/products/unmanned-systems/trc/`, 2012.

[102] Andy Qua. Cube runner. [Online] at `http://andyqua.co.uk/CubeRunner/Welcome.html`.

[103] Mahfuz Rahman, Sean Gustafson, Pourang Irani, and Sriram Subramanian. Tilt Techniques: Investigating the Desterity of Wrist-based Input. In *IEEE International Conference on Computer Human Interaction*, 2009.

[104] S. Ram and Jagdish N. Sheth. Consumer Resistance to Innovations: The Marketing Problem and its solutions. *Journal of Consumer Marketing*, 6(2):5–14, 1989.

[105] Recon Robotics. Throwbot XT - With Audio. [Online] at `http://www.reconrobotics.com/products/Throwbot_XT_audio.cfm`, May 2012.

[106] Elizabeth S. Redden, Linda R. Elliott, Rodger A. Pettitt, and Christian B. Carstens. Scaling Robotic Systems for Dismounted Warfighters. *Journal of Cognitive Engineering and Decision Making*, 5(2):156–185, June 2011.

[107] Jun Rekimoto. Tilting Operations for Small Screen Interfaces. In *Proceedings of the 9th annual ACM Symposium on User Interface Software and Technology*, 1996.

[108] Michell M. Rhode, Victor E. Perlin, Karl D. Iagnemma, Robert M. Lupa, Steven M. Rhode, James Overholt, and Graham Fiorani. PointCom: Semi-Autonomous UGV Control with Intuitive Interface. In *Proceedings of SPIE 6962, Umanned Systems Technology X*, number 6962. SPIE, April 2008.

[109] Robotic Systems Joint Project Office. Unmanned Ground Systems Roadmap. Technical report, Robotic Systems Joint Project Office, 2011.

[110] M.S. Sanders and E.J. McCormick. *Human factors in engineering and design*. McGraw Hill, 7th edition, 1993.

[111] Charles Scalesse. Tutorial: Accelerometer Calibration and Optimizations. [Online] at `http://iphonedevsdk.com/forum/iphone-sdk-tutorials/39833-tutorial-accelerometer-calibration-optimizations.html`, January 2013.

[112] Michael David Schmidt. Simulation and Control of a Quadrotor Unmanned Aerial Vehicle. Master's thesis, University of Kentucky, 2011.

[113] T. B. Sheridan. Human enhancement and limitation in teleoperation. *Progress in Astronautics and Aeronautics*, 161:43, 1994.

[114] Thomas B Sheridan. *Telerobotics, automation, and human supervisory control*. MIT press, 1992.

[115] Ben Shneiderman and Catherine Plaisant. *Designing the User Interface*. Addison-Wesley, 2010.

[116] Aaron Steinfeld, Terrence Fong, David Kaber, Michael Lewis, Jean Scholtz, Alan Schultz, and Michael Goodrich. Common metrics for human-robot interaction. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 33–40. ACM, 2006.

[117] Synovision Solutions LLC. Common Robot Controller Study. Technical report, Synovision Solutions LLC, 2007.

[118] Textron Systems and AAI Corporation. The Next Generation in One System Technology. [Online] at `www.aaicorp.com/pdfs/ugcs41709a.pdf`, 2009.

[119] Dawn Tilbury and Galip A. Ulsoy. A New Breed of Robots that Drive Themselves. *Mechanical Engineering*, 133(2):28, 2011.

[120] Z. A. Tomlinson. Influence of spatial ability on primary and secondary space telerobotics operator performance. *Aviation, Space, and Environmental Medicine*, 80:221, 2009.

[121] M R Tracey and C E Lathan. The interaction of spatial ability and motor learning in the transfer of training from a simulator to a real task. *Studies In Health Technology And Informatics*, 81:521 – 527, 2001.

[122] University of Kent Careers and Employability Service. Non-Verbal Reasoning Test. http://www.kent.ac.uk/careers/tests/spatialtest.htm, 2013.

[123] Jan B.F. van Erp. Controlling Unmanned Vehicles: the Human Factors Solution. In *RTO SCI Symposium on "Warfare Automation: Procedures and Techniques for Unmanned Vehicles"*. April 1999.

[124] Jan BF van Erp and Pieter Padmos. Image parameters for driving with indirect viewing systems. *Ergonomics*, 46(15):1471–1499, 2003.

[125] Luc Vandal and Ortwin Gentz. In App Settings Kit (version as of 1 AUG 2012). Open Source (BSD) at www.inappsettingskit.com/, 2009.

[126] Balint Viragh. Dension WiRC SDK. Limited Distribution, December 2011.

[127] Jijun Wang, Michael Lewis, and Stephen Hughes. Gravity-referenced attitude display for teleoperation of mobile robots. In *Proceedings of the Human Factors and Ergonomics Society 48th Annual Meeting*, 2004.

[128] Alan Ft Winfield. Future Directions in Tele-operated Robotics. Technical report, University of the West of England, 2000.

[129] Rosemarie E. Yagoda and Susan G. Hill. Using Mobile Devices for Robotic Controllers: Examples and Some Initial Concepts for Experimentation. Technical Report ARL-TN-436, Army Research Laboratory, 2011.

[130] Gerry Yarrish. Look, Ma! No Radio! *ModelAirplaneNews.com*, January 2011.

[131] Fareed Zakaria. Warrior robots in afghanistan. http://afghanistan.blogs.cnn.com/2011/04/05/warrior-robots-in-afghanistan/, April 2011.

Appendix A

Interviews with Company Commanders using Ground Robots in Afghanistan

A.1 Correspondence with CPT Michael Knox, U.S. Army

264th Engineer Clearance Company,
27th Engineer Battalion (Combat)(Airborne)
FOB Fenty, Jalalabad, Afghanistan
March 2012 - March 2013

Question 1: How much training did your unit receive on robot operations? This includes anything from training regarding TTPs to individual operator training on specific robotic platforms. I'm primarily interested in formal training that took place in CONUS prior to deployment.

> I have multiple Soldiers that have deployed to different locations throughout both Iraq and Afghanistan and all have different levels of experience/training with different types of robotics. We primarily use the Talons and were actually able to get our hands on a pair from our 18th ABN CORPS G3 CI2C to do a couple of training sessions. The training was primarily aimed at individual operation of the equipment itself as opposed to TTPs. NCOs and Soldiers with prior experience were able to pass on different TTPs that they had either picked up from EOD or deployment experience. We were only able to get the robots 2 or 3 times.

Question 2: How did you choose which Soldiers would operate the robots? Education, rank, luck of the draw? Or are all of your Soldiers expected to have some proficiency with the robots?

> Most of my Sappers have at least some sort of experience operating Talons. Typically Soldiers with demolitions experience or Soldiers that had been to either EOCA or R2C2 Sapper were primaries for robotic training. Also, typically you have the type of Soldier that is a driver/gunner or a dismount/Sapper. The dismount/Sapper type is usually the one out on the ground and is typically the operator of the robotics. Generally, everyone has some sort of experience due to pulling them out and playing with them on down days.

Question 3: You stated that you don't use the robots all of the time, how many hours per week would you estimate they are in operation? What are the primary missions for which they are deployed? Route clearance, reconnaissance of villages/locations/buildings? Are they deployed directly from the outpost/base or transported (either by vehicle or dismounted Soldier) to a pre-determined location and deployed from there?

> Like I said, we have marcbots, packbots, talons, and scouts. We really only use the talons due to their robust size and capabilities. I wouldn't be able to estimate an hourly usage per week as they are generally only used to either interrogate suspected IEDs that can't be visually observed by gyrocams or the naked eye or to emplace demolitions charges used to blow in place any sort of explosive hazard that needs reduced. Reducing explosive hazards by emplacing pre-prepared charges is the primary mission for which the talons are employed. We have a Talon RDS (remote delivery system) that was primarily designed for the Iraq theater where Sappers chose not to dismount. The Talons are typically prepped pre-mission and stowed in the RDS until an explosive hazard that needs to be reduced is located or identified at which point dismounts are dropped out of the patrol and the robot is employed to either interrogate or reduce the explosive hazard.

Question 4: Do Soldiers generally like the robots? What are the major complaints?

> The Soldiers typically like the talon robots because they are robust and reliable. The smaller robots typically are not even used because

the terrain where we are deployed is not conducive to the smaller robots. Mud, tall grass, water, trash all inhibit the use of the smaller robots such as the Scout. It was typically designed to throw into qalat structures to recon them but with the restrictive ROE in effect in the Afghanistan theater there is not really any need for us to enter/recon the inside of qalat structures. I would say the biggest complaint is the time it takes to place it into operation and to recover it.

Question 5: Do you personally see a need for research/improvement of robot controllers? If so, what would be the first thing you would look to change?

I don't personally think that implementing a fragile smaller touchscreen controller for any of the robotic systems is a feasible solution. The larger, current OCUs are more durable and easier to use with gloves. Also, the problem you run into when you start making smaller, technologically advanced controllers is you increase the need for civilian FSR support to keep the hardware running. My biggest complaint with the equipment that RD is providing the Army right now is that it is so advanced that it requires a small Army of civilian contractors to keep them running. Often, there is only 1 or 2 FSRs in theater and it takes days or weeks to get them to your location to repair or troubleshoot equipment. I think that my soldiers like to have a robust controller that is easier to work without removing gloves and that is less susceptible to being either crushed or lost.

A.2 Correspondence with CPT Jack Morrow, U.S. Army

Commander, 693rd Engineer Company (Sapper)

FOB Azizullah, Maiwand District, Kandahar Province

October 2012 - July 2013

Question 1: How much training did your unit receive on robot operations? This includes anything from training regarding TTPs to individual operator training on specific robotic platforms. I'm primarily interested in formal training that took place in CONUS prior to deployment.

> - Explosive Ordnance Clearance Agent (EOCA) Course – 7 slots. About 2 weeks worth of training on the PackBot and Talon robots. School runs for about one month total.
> - Route Reconnaissance Clearance Course (R2C2) Sapper – 6 slots. About 1 week of training on the Talon robot. School runs for about two weeks total.
>
> In theater school house training prior to RIP/TOA at Kandahar Airfield:
>
> - Blow in Place (BIP) Course – 12 slots for Soldiers who had already complete EOCA and/or R2C2 Sapper. 2 days are spent working with the Talon robot.
> - DOKING – 8 slots. 1 day course.

Question 2: How did you choose which Soldiers would operate the robots? Education, rank, luck of the draw? Or are all of your Soldiers expected to have some proficiency with the robots?

> I wish I could tell you I had a deliberate plan for this, but my choice of which Soldiers to send to the courses was really a matter of availability. I had approximately 45 schools (totaling about 200 slots) I had to send my Soldiers to prior to deploying. Some of these schools were FORSCOM Route Clearance Pre-deployment Requirements and some were developed by my battalion, the 7th Engineer Battalion. Most of

my school dates corresponded with a time when my company was at 55% strength. So basically if you met the minimum rank requirement (in some cases I had to get waivers for rank), you were going to go to one of the mandatory schools. When you got back from one of those schools, you were probably going to go to another and possibly a third. I was using names of Soldiers still in basic training (but on my gains roster) as place holders to secure the school dates. When a new group of Soldiers arrived, we just verified that the ones who showed up were the ones we slotted for schools (or changed names slotted for the school if our projected gains never made it out of Basic/AIT) and we immediately sent them to school. It was ridiculous, but it was the only way we were going to meet the minimum TDY school training requirements necessary for my company to deploy and assume a route clearance mission in Afghanistan.

Question 3: You stated that you don't use the robots all of the time, how many hours per week would you estimate they are in operation? What are the primary missions for which they are deployed? Route clearance, reconnaissance of villages/locations/buildings? Are they deployed directly from the outpost/base or transported (either by vehicle or dismounted Soldier) to a pre-determined location and deployed from there?

We currently use the Talon robot for charge placement, interrogation of suspected IEDs, and cutting 550 cord when manually emplacing MICLIC charges with the DOKING. We use Rapid Deployment System mounted back of RG-31 to carry and deploy the Talon on patrol. Robot backs into the mount and is locked in place, ready to deploy.

While the DOKING has a number of great uses (flail for brush clearing), we have been primarily using it to manually emplace MICLIC charges. We do this because we can guarantee the charge is centered on the road where we suspect a large number of IEDs are. By doing so, we also eliminate the possibility of the rocket (normal deployment of MICLIC) veering to the left or right and pulling the MICLIC into an area where it might cause collateral damage. Our technique is simple. We simply tie the MICLIC to the DOKING pintle with 550 cord and send the Talon downrange with a knife taped to

the arm to cut the 550 cord. Weve emplaced 21 MICLICs this way, and it has work well.

Question 4: Do Soldiers generally like the robots? What are the major complaints?

SSG Thomas Gribble: The battery life of laptop style remote controls is a big problem for PackBot. Currently, you need a reliable 110 outlet in vehicle to keep the remote charged for the duration of a patrol. On the plus side, the PackBot is smaller and lighter than Talon and just as maneuverable, though you cant pull or lift as much weight with it as you can the Talon. The 360 degree camera system is great on the Talon. Line of Sight is a problem between the Talon remote control and the robot. The current OCUNeed far too large and cumbersome; you simply cannot move very far as a dismount while operating the Talon. In a firefight it would not be easy to stow the OCU, shoot your weapon, and continue operating Talon in one quick motion. You cant just take off running for cover dragging the briefcase while it is open. Need a smaller OCU, much smaller. Need longer battery life, batteries dont recharge well or hold charge well in combat environment. Joystick controls for the Talon work well. Could easily make this smaller and keep the same general control joystick concept. Touch screen would be OK if you could keep the controls for the upper and lower arm separate. Keep the 8 frequencies channels on the OCU the same. Having the ability to change frequencies in case someone jams you is critical. Keep the 360 degree camera on the Talon. This works great.

SPC Jeremy Meeks: Current OCU works well from a control standpoint as it is dual joystick driven. Its simple to use, and you could teach any Soldier how to operate it in 5 minutes. The current battery wont last for an entire patrol. The OCU is a little big, but the main issue is there is no internal battery compartment. The OCU has a power cord that connects to an external battery. If you had to run and take cover from direct fire while operating the DOKING on the ground, you would have a large battery in your pocket bouncing and cord dangling that could get caught on something. Dont change the controls, just make it smaller. I would rather have joysticks than a touch screen, but I feel like I could learn to use a touch screen with enough stick time. The DOKING itself works fine in my opinion, though it would be good to make it easier

to connect and disconnect components such as the flail and blade because it currently takes 3 Soldiers to change these components. The hydraulic system leaks quite a bit. After operation it is not uncommon to find a lot of hydraulic fluid collected in a pool on the belly plate. Currently there is a special key you need to open up the side compartments to check the battery and wiring of the actual DOKING. Given the turnover of equipment from unit to unit in theater and operator to operator during missions, there is the potential for the key to be lost easily and this isnt practical in combat.

SGT Fredrick Snook: The Talon robot itself is pretty much exactly what I want. The Talon OCU is very easy to control and the 360 degree camera is very good. Even making the joysticks smaller (operated with thumbs) would be an improvement. Currently there are three joysticks for the Talon (drive, upper arm, lower arm). Camera and the robot wrist are on toggle switches, creating five different components of control being used at any given time. Some of these functions could be consolidated into fewer joysticks/toggles, or at least moved so that it can be operated without having to pick up my hand and move it to a different part of the control. Every time I have to move my hand, I have to look down at the OCU and take my eyes off of what the camera is seeing. There are a lot of switches on the OCU that I never use. I dont see any reason why a lot of the controls couldnt be consolidated into a touch screen. If I had to choose, I would keep the joysticks, but I could deal with using a touch screen if that is what it took to make the OCU more transportable. You could make the OCU more like an X-box controller (similar to the Raven UAV) and that would make it more transportable.

CPL Sean ODonnell: Having an external joystick of some kind is important for easy operation. Sometimes touch screens (like we have for our rollers) tend to stick, and that delay in operation could mean the difference between our robot doing what it is supposed to do and ending up stuck in a ditch in the middle of an IED belt. An X-box type controller would work well in my opinion. The controller already is partially a touch screen, so I dont see any issue keeping a touch screen but not as a standalone means of controlling the DOKING. Having an internal battery pack for the DOKING OCU is common sense. Maybe even modeling the controller after Nintendo DS controller with a flip up screen could work. The front mounted camera on the DOKING should be an oscillating (360 degree) camera. This would make

it much easier to drive the DOKING particularly in restricted terrain.

SGT Denver Colin, Team 2, 2nd Platoon, 766th OD CO (EOD): We currently use the Talon robot when conducting mounted patrols and the PackBot 310 when patrolling dismounted with infantry. We also have an Armadillo, but we had problems with the arm, mainly that it would not deploy properly. The arm usually fell into the ground. I like the Talon because it is durable and can go almost anywhere. The arm is strong and can pull heavy jugs out of the ground. The OCU is big and bulky, but it is very easy to use. The one thing I would improve on the Talon is the clarity of the cameras. The OCU for the PackBot 310 is light, transportable, and easy to use, but the robot itself is not very good. The PackBot 310 has limited range, and does not do well in the mud. The arm does work, but not as well the Talons Army does. The current PackBot 310 OCU includes an eye piece display and an X-box type controller with the main OCU component carried on the back. The eye piece is difficult to see unless you cover your head with a t-shirt or something similar. You can also use a heads down monitor about 5 x 7 instead of the eye piece, and this works pretty well. If I needed to take off running for cover, I could shove the screen in my vest and find cover. I worry about the responsiveness and sensitivity of a tilt based control. Im more confident and comfortable with joystick operated controls and my ability to adjust the sensitivity/responsiveness of the controls to my needs. Perhaps having a combination of knobs and touchscreen could work. The important thing with a touch screen would be able to do everything on one screen without flipping back and forth.

Appendix B

Institutional Review Board #1115

The UNIVERSITY of OKLAHOMA

Institutional Review Board for the Protection of Human Subjects

Approval of Initial Submission – Expedited Review – AP01

Date: August 31, 2012 **IRB#:** 1115

Principal Investigator: Amber M Walker **Approval Date:** 08/31/2012

Expiration Date: 07/31/2013

Study Title: Attitude Aware Smartphones for Tele-operated Robot Control

Expedited Category: 6 & 7

Collection/Use of PHI: No

On behalf of the Institutional Review Board (IRB), I have reviewed and granted expedited approval of the above-referenced research study. To view the documents approved for this submission, open this study from the *My Studies* option, go to *Submission History*, go to *Completed Submissions* tab and then click the *Details* icon.

As principal investigator of this research study, you are responsible to:

- Conduct the research study in a manner consistent with the requirements of the IRB and federal regulations 45 CFR 46.
- Obtain informed consent and research privacy authorization using the currently approved, stamped forms and retain all original, signed forms, if applicable.
- Request approval from the IRB prior to implementing any/all modifications.
- Promptly report to the IRB any harm experienced by a participant that is both unanticipated and related per IRB policy.
- Maintain accurate and complete study records for evaluation by the HRPP Quality Improvement Program and, if applicable, inspection by regulatory agencies and/or the study sponsor.
- Promptly submit continuing review documents to the IRB upon notification approximately 60 days prior to the expiration date indicated above.
- Submit a final closure report at the completion of the project.

If you have questions about this notification or using iRIS, contact the IRB @ 405-325-8110 or irb@ou.edu.

Cordially,

Lara Mayeux

Lara Mayeux, Ph.D.
Vice Chair, Institutional Review Board

Figure B.1: IRB Approval Letter

Human-Robot Interaction User Study

ATTITUDE AWARE SMARTPHONES FOR TELE-OPERATED ROBOT CONTROL
Amber M. Walker
School of Aerospace & Mechanical Engineering
IRB # 1115

Have you ever wanted to drive a robot with your iPhone? Now's your chance!

- **Come join our study to examine various smartphone-based robot controllers for use with a tele-operated tracked ground robot. Participants will maneuver the robot on an urban reconnaissance course and report their impressions via survey to assess usability.**

- **Participants must be between 18-40 years old. We are recruiting both male and female users with visual acuity correctable to 20/20. No previous experience with robots or smartphones is required.**

- **Participants will conduct approximately 5 hours to testing over the course of 6 weeks, broken down into 3 experiments. *Participants must complete all 3 experiments to receive full compensation.***

- **Please contact Amber Walker directly at amber.m.walker@ou.edu [preferred] or (405) 325-0731.**

Attend one of our information sessions and complete the first experiment phase to receive a Starbucks gift card!

10 September @ 1200	**Felgar Hall CAD Lab (Rm 146)**
12 September @ 1700	**Felgar Hall CAD Lab (Rm 146)**

The University of Oklahoma is an equal opportunity institution. Accommodations on the basis of disability are available by contacting Amber Walker at (405) 325-0731 or amber.m.walker@ou.edu by 10 September 2012.

IRB APPROVED AAHRPP
IRB NUMBER: 1115
IRB APPROVAL DATE: 08/31/2012

Figure B.2: IRB Recruitment Material

University of Oklahoma
Institutional Review Board
Informed Consent to Participate in a Research Study

Project Title: Attitude Aware Smartphones for Tele-operated Robot Control
Principal Investigator: Amber M. Walker
Department: School of Aerospace & Mechanical Engineering

You are being asked to volunteer for this research study. This study is being conducted at the University of Oklahoma, Norman, Oklahoma. You were selected as a possible participant because of your interest in the project and your willingness to complete all three phases of the proposed study.

Please read this information sheet and contact me to ask any questions that you may have before agreeing to take part in this study.

Purpose of the Research Study

The purpose of this study is to conduct a usability assessment on several versions of an attitude aware (tilt-based) smartphone controller for use in tele-operated robot control, where the operator is not physically present with the robot. These specific controller prototypes are being examined with a particular focus on urban military reconnaissance operations, and the course and tasks will reflect such.

Number of Participants

About 40 people will take part in this study.

Procedures

If you agree to be in this study, you will be asked to complete 3 experiment phases, in addition to this pre-interview. An experiment phase will consist of two timed trials where users are trained on a version of the robot controller and then asked to complete an indoor "urban reconnaissance course" to the best of their ability. The course will test maneuver skills, as well as visual identification of targets via camera manipulation. Users will then complete a series of post-iteration/post-experiment questionnaires including a usability assessment and the NASA TLX Workload Index.

These 3 experiment phases will build upon one another in terms of controller and task complexity; however, the course and hardware will remain unchanged. We will attempt to complete one experiment phase every two weeks, for a total of six weeks.

Length of Participation

This pre-interview is scheduled for 30 minutes, and each individual experiment phase is expected to take 1 hour and 30 minutes per participant, for a total time commitment of 5 hours. Participants may voluntarily withdraw at any time. Participants would be involuntarily terminated from the study only if conditions so warrant, i.e. unable to reach the participant for over 7 days, participant misses more than 1 scheduled appointment, etc.

Risks and Benefits
Participants should expect few, if any, risks from participating in this study. Possible physical risks include eyestrain, headaches, tunnel vision, or motion sickness from using the smartphone based controller for up to one hour at a stretch; however, these effects are highly unlikely. Most participants are familiar with the hardware being used, and unless they have already suffered similar side-effects from smartphone use there is no reason to think this study will trigger them. Should a participant at any time suffer any ill-effects, they need only make it known to the researcher and the study will be halted.

Compensation
You will be reimbursed for your time and participation in this study. Participants will receive reimbursement according to the following schedule:

Pre-interview *AND* Experiment #1: $10
Experiment #2 *AND* Experiment #3: $30

Those completing through experiment #2 and not #3 will not receive the $30 compensation. The first $10 payment will be a Starbucks gift card. The second $30 payment will be a gift card to a choice of one of three stores (Wal-Mart, Amazon, or Target). Participants may withdraw from the study at anytime; however, compensation will be paid out only at the times outlined above and will not be pro-rated in cases of partial completion.

Confidentiality
In published reports, there will be no information included that will make it possible to identify you. Research records will be stored securely and only approved researchers will have access to the records.

There are organizations that may inspect and/or copy your research records for quality assurance and data analysis. These organizations include various Department of Defense entities including the United States Military Academy, the Army Research Labs, and the Tank and Automotive Research, Development, and Engineering Center, as well as the OU Institutional Review Board.

Voluntary Nature of the Study
Participation in this study is voluntary. If you withdraw or decline participation, you will not be penalized or lose benefits or services unrelated to the study. If you decide to participate, you may decline to answer any question and may choose to withdraw at any time.

Waivers of Elements of Confidentiality

Your name will not be retained or linked with your responses unless you specifically agree to be identified. The data you provide will be retained in anonymous form unless you specifically agree for data retention or retention of contact information beyond the end of the study. Please check all of the options that you agree to:

I consent to being quoted directly. ___ Yes ___ No

I consent to having my name reported with quoted material. ___Yes ___ No

I consent to having the information I provided retained for potential use in future studies by this researcher. ___Yes ___ No

I consent to having my contact information retained after the study so that I can be contacted to participate in future studies. ___ Yes ___ No

Video Recording of Study Activities

To assist with accurate recording of your responses, observations/experiment trials may be recorded on a video recording device. You have the right to refuse to allow such recording. Please select one of the following options:

I consent to video recording. _____ Yes _____ No

Video may be retained and used for conference proceedings, demonstrations, or seminars should the participant consent and sign a talent release. Every effort will be made to limit identifying data regarding participants, and names will not be recorded or included with any public release.

I consent to any video recording being retained for possible public release (solely for an academic purpose).

_____ Yes _____ No

Photographing of Study Participants/Activities

In order to preserve an image related to the research, photographs may be taken of participants. You have the right to refuse to allow photographs to be taken without penalty. Please select one of the following options:

I consent to photographs. _____ Yes _____ No

Contacts and Questions

If you have concerns or complaints about the research, the researchers conducting this study can be contacted at:

Amber Walker: amber.m.walker@ou.edu or (405) 996-0270
Professor David P. Miller: dpmiller@ou.edu or (405) 325-5011.

Contact the researcher(s) if you have questions or if you have experienced a research-related injury.

If you have any questions about your rights as a research participant, concerns, or complaints about the research and wish to talk to someone other than individuals on the research team or if you cannot reach the research team, you may contact the University of Oklahoma – Norman Campus Institutional Review Board (OU-NC IRB) at 405-325-8110 or irb@ou.edu.

701-A-1

You will be given a copy of this information to keep for your records. If you are not given a copy of this consent form, please request one.

Statement of Consent

I have read the above information. I have asked questions and have received satisfactory answers. I consent to participate in the study.

Participant Signature	Print Name	Date

Signature of Person Obtaining Consent	Date

Print Name of Person Obtaining Consent

Signature of Witness (if applicable)	Date

Print Name of Witness

Figure B.3: IRB Informed Consent

Appendix C

Experiment Results: User Questionnaires, Experiment Logs, and Participant Comments

Results presented in Appendix C are presented by phase, averaged for all users. For by-user results, please reference Appendix D.

C.1 Demographics

25 participants were recruited as experiment subjects and reimbursed $40 for their time. Demographic information was collected via survey during pre-interviews, the results of which are reported here (Figure C.1). A five-point Likert scale was used to measure participant experience and proficiency (see Table C.1).

Table C.1: Population Demographics: Proficiency/Experience Likert Scale

Excellent	5
Good	4
Average	3
Poor	2
No experience	1

Question	Answers	# Responded
Participants:		25
Gender:	*male*	21
	female	4
Age:	*mean*	27
	median	26
Which hand is your dominant hand?	*left*	1
	right	24
Do you wear prescription lenses?	*yes*	15
	no	10
If you are currently pursuing a college degee, what is your area of concentration?	*Chemical Engineering*	1
	Mechanical Engineering	7
	Economics	5
	Aerospace Engineering	2
	Computer Engineering	2
	Computer Science	2
	Mathematics	1
	Physics	1
	not applicable	4
Have you ever served in any branch of the armed forces?	*yes, Army*	1
	yes, Marines	1
	yes, Reserves	1
	yes, Army National Guard	1
	no	21
How would you rate your overall technical proficiency on mobile devices (smartphones and tablets)?	*mean*	4.24
How would you rate your overall technical proficiency on computers (laptops or PCs)?	*mean*	4.44
How many hours per week do you spend using mobile technology (smartphones and tablets), on average?	*0-12*	3
	12-24	9
	24-48	7
	48-80	5
	80+	1
Rate skill level on…		
Playing commercial video games (XBox360, Playstation, Wii):	*mean*	3.32
Playing smartphone-based games:	*mean*	3.28
Operating remote controlled ground vehicles:	*mean*	2.84
Operating remote controlled aerial vehicles:	*mean*	2.36
Annotate which, if any, gesture-based control systems you have used/experienced:		
Nintendo Wii	*yes*	20
Parrot A.R. Drone quadcopter	*yes*	3
Tilt-based iPhone games i.e. Tilt to Live, Jetman	*yes*	14
Sphero	*yes*	0
Other		1
Spatial reasoning test:	*mean*	13.4

Figure C.1: Population Demographics Survey Results

C.2 Performance Measures: Experiment Log

In the following figures, results are presented by controller level [A-E], by phase.

- Phase 1 - Figure C.2
- Phase 2 - Figure C.3
- Phase 3 - Figure C.4

Controller A performance metric:		***mean***	***median***
Total practice time	*(sec)*	688	714
Results of training run	*Passed (Y/N)*	Y: 23	
	Time (sec)	67	61
Total trial time	*(sec)*	306	292
# driving errors	*major*	0.28	0
	minor	5.64	5
# path points tagged	*out of 32*	28.6	29

Controller B performance metric:		***mean***	***median***
Total practice time	*(sec)*	649	699
Results of training run	*Passed (Y/N)*	Y: 23	
	Time (sec)	76	72
Total trial time	*(sec)*	340	323
# driving errors	*major*	0.4	0
	minor	4.16	4
# path points tagged	*out of 32*	28.2	29

Figure C.2: Phase 1 Performance Results

Controller C performance metric:		***mean***	***median***
Total practice time	*(sec)*	193	152
Total trial time	*(sec)*	542	542
# driving errors	*major*	0.04	0
	minor	3.32	3
# path points tagged	*out of 32*	25.44	26
# objects identified	*out of 3*	2.92	3
Accuracy of object location	*out of 9*	7.2	8

Controller D performance metric:		***mean***	***median***
Total practice time	*(sec)*	152	140
Total trial time	*(sec)*	556	527
# driving errors	*major*	0.12	0
	minor	3.12	3
# path points tagged	*out of 32*	26.08	27
# objects identified	*out of 3*	2.88	3
Accuracy of object location	*out of 9*	6.64	7

Figure C.3: Phase 2 Performance Results

Controller E performance metric:		***mean***	***median***
Total practice time	*(sec)*	953	910
Total trial time	*(sec)*	469	441
# driving errors	*major*	0.08	0
	minor	3.6	3
# path points tagged	*out of 32*	26.96	27
# objects identified	*out of 3*	2.96	3
Accuracy of object location	*out of 9*	6.28	7

Figure C.4: Phase 3 Performance Results

C.3 NASA TLX Results

The following figures show NASA TLX Workload scores by phase [1-3].

- Phase 1 - Figure C.5
- Phase 2 - Figure C.6
- Phase 3 - Figure C.7

The **tlx_score**, or total weighted workload, served as the dependent variable summarizing this data in the results chapters.

Controller A NASA TLX:		***mean***
Weighted workload_mental	*weighted % of 100*	12.61
Weighted workload_physical	"	2.16
Weighted workload_temporal	"	7.35
Weighted workload_perform	"	7.7
Weighted workload_effort	"	15.51
Weighted workload_frustration	"	11.81
Total tlx_score	*out of 100*	56.93

Controller B NASA TLX:		***mean***
Weighted workload_mental	*weighted % of 100*	14.04
Weighted workload_physical	"	3.34
Weighted workload_temporal	"	6.97
Weighted workload_perform	"	8.23
Weighted workload_effort	"	13.87
Weighted workload_frustration	"	8.52
Total tlx_score	*out of 100*	54.96

Figure C.5: Phase 1 NASA TLX Results

Controller C NASA TLX:		***mean***
Weighted workload_mental	*weighted % of 100*	11.46
Weighted workload_physical	"	3.37
Weighted workload_temporal	"	6.95
Weighted workload_perform	"	6.7
Weighted workload_effort	"	12.34
Weighted workload_frustration	"	3.89
Total tlx_score	*out of 100*	44.72

Controller D NASA TLX:		***mean***
Weighted workload_mental	*weighted % of 100*	12.22
Weighted workload_physical	"	2.62
Weighted workload_temporal	"	8.13
Weighted workload_perform	"	6.41
Weighted workload_effort	"	11.44
Weighted workload_frustration	"	4.49
Total tlx_score	*out of 100*	46.07

Figure C.6: Phase 2 NASA TLX Results

Controller E NASA TLX:		***mean***
Weighted workload_mental	*weighted % of 100*	10.9
Weighted workload_physical	"	2.61
Weighted workload_temporal	"	7.07
Weighted workload_perform	"	7.28
Weighted workload_effort	"	10.73
Weighted workload_frustration	"	3.17
Total tlx_score	*out of 100*	41.76

Figure C.7: Phase 3 NASA TLX Results

C.4 System Usability Scale Results

The following figures show System Usability Scale results by phase [1-3].

- Phase 1 - Figure C.8
- Phase 2 - Figure C.9
- Phase 3 - Figure C.10

These were collected as part of each post-iteration survey. The **SUS** score, or total usability, served as the dependent variable summarizing this data in the results chapters. Questions 1 through 10 [**Q1-Q10**], which comprise the SUS, are rated by users on a scale of one (*strongly disagree*) to five (*strongly agree*). All numerical results are scaled such that larger numbers indicate a desirable outcome (i.e. more usable) [18].

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.

8. I found the system very cumbersome to use.

9. I felt very confident using the system.

10. I needed to learn a lot of things before I could get going with this system.

Controller A SUS:		***mean***
Q1	*out of 5*	1.72
Q2	"	2.92
Q3	"	2.56
Q4	"	3.44
Q5	"	2.88
Q6	"	2.08
Q7	"	2.88
Q8	"	2.12
Q9	"	2.16
Q10	"	3.2
Overall SUS	*out of 100*	64.9

Controller B SUS:		***mean***
Q1	*out of 5*	2.16
Q2	"	3.04
Q3	"	2.56
Q4	"	3.2
Q5	"	2.76
Q6	"	2.68
Q7	"	2.92
Q8	"	2.56
Q9	"	2.44
Q10	"	3.16
Overall SUS	*out of 100*	68.7

Figure C.8: Phase 1 System Usability Scale Results

Controller C SUS:		mean
Q1	*out of 5*	2.88
Q2	"	3.16
Q3	"	3.2
Q4	"	3.38
Q5	"	3.04
Q6	"	3.08
Q7	"	3.2
Q8	"	2.96
Q9	"	2.92
Q10	"	3.28
Overall SUS	*out of 100*	77.4

Controller D SUS:		mean
Q1	*out of 5*	2.88
Q2	"	3.04
Q3	"	3.08
Q4	"	3.6
Q5	"	3.04
Q6	"	2.76
Q7	"	3.04
Q8	"	2.64
Q9	"	2.84
Q10	"	3.24
Overall SUS	*out of 100*	75.1

Figure C.9: Phase 2 System Usability Scale Results

Controller E SUS:		mean
Q1	*out of 5*	3.04
Q2	"	2.96
Q3	"	3.32
Q4	"	3.4
Q5	"	3.16
Q6	"	2.92
Q7	"	3
Q8	"	3.12
Q9	"	3.2
Q10	"	2.88
Overall SUS	*out of 100*	77.2

Figure C.10: Phase 3 System Usability Scale Results

C.5 Post-Iteration Survey Results

Users took post-iteration surveys after each trial, primarily to gather information specific to the controller level. Questions were both free-text answer as well as ranked multiple choice. The survey was comprised of two sections:

1. The first asked questions regarding quality of training, ranked using the Likert scale defined in Table C.2.

2. The second asked users to rate their ability to conduct certain trial subtasks (related to either driving or camera manipulation), ranked using the Likert scale defined in Table C.3.

Table C.2: Post-Iteration Questionnaires: Training Likert Scale

More than adequate	4
Adequate	3
Less than adequate	2
Inadequate	1

Table C.3: Post-Iteration Questionnaires: Trial Subtask Skill Likert Scale

Extremely easy	5
Easy	4
Neither easy nor difficult	3
Difficult	2
Extremely difficult	1

C.5.1 Phase 1

Results are presented in two parts, training (Figure C.11) and maneuver (Figure C.12). Additionally, Figures C.13 and C.14 depict the "easiest" and "hardest" tasks with each controller type, as defined by users.

Phase 1--Post-Iteration Survey: Training

Question	Controller A *(mean)*	Controller B *(mean)*
Rate your training on the surveyed controller in each of the following areas:		
Length of formal training portion (with instructor)	3.36	3.28
General level of detail/instruction provided by instructor	3.48	3.56
Length/availability of hands-on practice	3.48	3.36
Understanding of tasks	3.52	3.52
Understanding the display	3.52	3.56
Understanding the controls	3.24	3.12
Driving the robot	3.04	3.04
Rate the overall quality of training:	3.44	3.4

Figure C.11: Post-Iteration Questionnaire: Phase 1 Results (Training)

Training comments (Controller A):

- Plenty of time to get at least semi-acquainted with the controls, especially given their ease of use.
- Easier to drive robot in straight paths.
- Well organized training and nice instructions.
- All aspects were well explained with more than adequate familiarization time to prepare for the trial.
- Joystick was more intuitive than the tilt-based controller.
- If there were arrows on the display that indicated orientation or a wider camera view it would help in navigation. Learn about how to drag the joystick left and right made an extraordinary difference in controlling the robot.
- Joystick seemed unresponsive.

Training comments (Controller B):

- The tilt-based control seems better than the joystick for steering and control while going fast. Similar to my problems with decreasing acceleration with the joystick, throttle was a little difficult with the tilt-based control. Going slow and medium speeds were easy, but going faster or a sufficient speed seemed more difficult and almost unobtainable at times (such as at the top of the ramp).
- The only thing that became a little bit of an issue was the understanding of the sensitivity of the tilt and what effect it would have on the robot.
- You did a great job!
- Plenty of time to practice controls and familiarize with objectives.
- I don't think that any more time would have improved performance.
- I would often try to re-engage control of the robot before resetting the phone's position.

Phase 1--Post-Iteration Survey: Driving

Question	Controller A *(mean)*	Controller B *(mean)*
Rate you rability to complete each of the following maneuver tasks for the surveyed controller:		
Move in the correct direction	2.44	3.52
Avoid obstacles	2.52	3.08
Identify any features that might have an adverse effect on the abiity of the robot to maneuver through the course	3.52	3.54
Anticipate whether the turn radius of the vehicle will allow a turn	3.28	3.16
Identify if you are on the course	3.64	3.63
Maintain control when driving at slowest speeds	3.48	3.68
Maintain control when driving at medium speeds	2.63	2.8
Maintain control when driving at fastest speeds	1.72	1.88
Return to the correct route after navigating around obstacles	3.38	3.16
tasks	2.96	3.2
Overall controller usability:	3	3.56

Figure C.12: Post-Iteration Questionnaire: Phase 1 Results (Driving)

Driving comments (Controller A):

- It seemed like there was a lot of dead space in the middle then it ramped up very quickly to full speed.
- The control was very sensitive with little gap between slow and FAST. Also it seemed that at times the control responded to input with a lag.
- Again, sensitivity became a small factor when trying to maneuver robot to point in direction of arrows
- the sensitivity of the controls was not linear, and felt a bit 'wonky'
- It is hard to use the controller with no use of iphone
- Identifying obstacles was difficult with respect to obstacles in peripheral field-of-view Most difficult aspect was the stop¡—¿full speed travel distance on the joystick. Otherwise, the theory of operation was sufficient to drive comfortably
- Could not maintain forward movement and change direction of turn. Occasionally it turned to opposite direction when I attempted to move forward and turn simultaneously.
- Video was laggy and straight forward was hard.
- It can become more accurate with practice.
- I had a lot of trouble determining how far to move the joystick to find a comfortable speed at which to travel and turn. It seemed as though sometimes the robot acted differently from how I was instructing it. Specifically, I would try to move forward and it would veer left, and if I tried moving at a slower speed it would try to reverse. Aside from that, the turns did just as I planned for them to. Because I had a hard time finding a good speed at which to travel/turn, I ended up just moving in quick spurts and correcting where I needed to.
- Very difficult to drive exactly straight.

Driving comments (Controller B):

- I really liked the controller and felt like it was easy to use, I just didn't feel like the robot responded very well to the controls.
- The range of motion gave a sense of less sensitivity. The operation of the robot felt more 'in control'
- much more difficult than the joystick
- I think the easiness to use the controller is the biggest merit. I do like it very much.
- Laggy video made small corrections hard, and I had a hard time turning in place with out going forward or backward
- Because of the limited view, I could not see all corners of the robot, allowing me to only be prepared for obstacles in my view. I don't think I did a center-point turn, only because I could not find the right angle at which to hold the controller, so I had to plan for a turn that may have moved the robot laterally into an obstacle.

Controller A Easiest/Hardest Tasks

Easiest:	**# of responses**
Seeing the course	3
Reverse	4
Turning in place	4
General movement	6
Learning movements of robot and joystick	5
Camera/video interface	3
Hardest:	
Throttle	3
Sensitivity	7
Moving straight forward	7
Seeing the course	1
Learning movements of robot and joystick	2
General movement	5

Figure C.13: Controller A: Easiest/Hardest Control Tasks

Controller B Easiest/Hardest Tasks

Easiest:	**# of responses**
General movement	6
Driving forward	6
Turning in place	2
Reversing	1
Controls are intuitive; easy to learn	6
User interface	1
Deadman switch	1
Seeing the course	2
Hardest:	
Turning in place	5
Controller "neutral," pose independence	3
Driving forward	5
Sensitivity	6
Tele-operation	2
Unfamiliarity	2

Figure C.14: Controller B: Easiest/Hardest Control Tasks

C.5.2 Phase 2

Results are presented in two parts, training (Figure C.15) and camera manipulation (Figure C.16). Additionally, Figures C.17 and C.18 depict the "easiest" and "hardest" tasks with each controller type, as defined by users.

Phase 2--Post-Iteration Survey: Training

Question	Controller C *(mean)*	Controller D *(mean)*
Rate your training on the surveyed controller in each of the following areas:		
Length of formal training portion (with instructor)	3.6	3.64
General level of detail/instruction provided by instructor	3.52	3.68
Length/availability of hands-on practice	3.72	3.72
Understanding of tasks	3.48	3.52
Understanding of task evaluation	3.48	3.44
Understanding the controller display	3.52	3.44
Operating the camera controls	3.44	3.56
Driving the robot	3.56	3.52
Rate the overall quality of training:	3.56	3.52

Figure C.15: Post-Iteration Questionnaire: Phase 2 Results (Training)

Training comments (Controller C):

- I had some difficulty panning the camera with smoothness rather than jumpy, jerky movements.
- I preferred the joystick over the tilt camera because the screen I was looking at did not have to move
- I didn't fully understand the rules on where the colored poms could be located until I had found the first one.
- More than enough training time –¿ most aspects more than adequate training time
- There was some camera lag when driving the robot
- The controls seemed more smooth this time, I don't know if they were tweaked or if it is just more practice.
- needed to ask about joystick irregularities
- The joystick was much easier than tilt for operating the camera – less intrinsic confusion when switching between modes of operation
- Perfectly explained, answered questions well

Training comments (Controller D):

- I had some difficulty adjusting to using the camera with the tilt feature and not moving the rest of my body.
- When I would tilt the camera, I felt like I had to rotate my head in the same direction as the tilt to get a good view of the screen
- A persistent snapshot button would have made it more obvious I needed to take a snapshot. I often did not notice that it had appeared.
- No lag. Both the driving and camera control seemed more responsive.

Phase 2--Post-Iteration Survey: Camera Manipulation

Question	Controller C *(mean)*	Controller D *(mean)*
Rate you rability to complete each of the following maneuver tasks for the surveyed controller:		
Identify any features that might have an adverse effect on the ability of the robot to maneuver through the course	3.6	4.04
Identify if you are on the course	3.8	4.12
Pan camera	4	4.12
Tilt camera	4.08	4.21
Identify which direction the camera is looking with respect to the robot	3.72	4
Scanning surroundings	4.04	3.96
Capturing still photograph	4.48	4.48
Overall controller usability:	4.125	4.125

Figure C.16: Post-Iteration Questionnaire: Phase 2 Results (Camera Control)

Camera control comments (Controller C):

- The camera moves a bit fast with the joystick. It was slightly more difficult to control with precision.
- Again it was difficult to know how far the sides of the robot were from an obstacle.
- lag in taking pictures; not present in other controller
- The joystick was a little too touchy to me, so I felt as if I was jumping around to the extreme ranges of the camera instead of slowly scanning the scene.
- i liked the joystick better than the tilt for the camera
- Interestingly, I found it much easier to scan the environment using the tilt controller. However, the orientation of the camera with respect to the robot was slightly less apparent.
- it is hard to pan camera by small amounts

- Fantastic control, the camera worked well.
- Had to understand how joystick works. it's a little different than expected

Camera control comments (Controller D):

- It was only difficult to remain still and only move the phone rather than my body. It is still a little difficult to visualize the sides of the robot when you are driving so that you know you are not going to run into something.
- The camera controls were great. The only less than perfect thing was the time laps I had between finding the object, pausing, taking a picture, and then returning. I'm not sure if that could become quicker, but it was great in general!
- The tilt camera was more difficult for me than the joystick-controlled camera.
- I found the ability to identify which direction the camera was looking wrt the robot to be a non-issue due to its automatic return-to-center when driving. Or it may have simply been so easy to keep track of that I never thought about it...
- hard to pan by small amounts
- I found myself trying to use a joystick for the camera control instead of tilting a couple of times. Overall I found the joystick version to be a bit more intuitive to use.

Controller C Easiest/Hardest Tasks

Easiest:	**# of responses**
Camera control	9
Scanning, boxes	4
Understanding controls	5
Driving	3
Still photograph	1
Returning camera to default	1
No response	2
Hardest:	
Camera control	6
Driving	6
Resolving camera heading from robot heading	1
Perceiving width of robot	1
Video/control lag	1
Sensitivity	4
Localization	1
Understanding task evaluation	1
Joystick	1
No response	3

Figure C.17: Controller C: Easiest/Hardest Control Tasks

Controller D Easiest/Hardest Tasks

Easiest:	**# of responses**
Camera control	7
Panning, tilting	4
Still photographs	3
Understanding the controls	2
Driving	8
No response	1
Hardest:	
Camera control	7
Driving	7
Scanning	4
Resolving camera heading from robot heading	1
Understanding tasks	1
Still photographs	1
Separating functions (drive and camera)	1
Localization	1
No response	2

Figure C.18: Controller D: Easiest/Hardest Control Tasks

C.5.3 Phase 3

All Phase 3 results were collected post-*experiment*, since only one timed trial was conducted. A portion of that survey asked the same training and manipulation questions asked in Phases 1 and 2's post-iteration surveys; therefore, those results are reported here: training (Figure C.19) and driving/camera control (Figure C.20). Additionally, Figure C.21 depicts the two most important customizable options used to create Controller E, as defined by users.

Phase 3--Post-Iteration Survey: Training

Question	Controller E *(mean)*
Rate your training on the customizable controller in each of the following areas:	
Length of formal training portion (with instructor)	3.56
General level of detail/instruction provided by instructor	3.52
Length/availability of hands-on practice	3.8
Understanding of tasks	3.6
Understanding the controller display	3.48
Operating the camera controls	3.52
Driving the robot	3.44
Manipulating the camera	3.52
Setting up controller via settings pane	3.4
Rate the overall quality of training:	

Figure C.19: Post-Iteration Questionnaire: Phase 3 Results (Training)

Training comments (Controller E):

- I was a little unsure about the controller turning controls?
- Good training, no complaints
- Training was fine, plenty of time to try out various settings.
- already knew most of it, so very simple
- The on-screen descriptions were not perfect, but instructor training makes up for it.
- The customizable interface was very intuitive.

- I enjoyed being able to customize the features.
- The instructions were concise, and being able to choose my controls and their layout made it personalized and easier for me to use.
- Settings pane would be hard to find if you had no idea where to look.

Phase 3--Post-Iteration Survey: Driving & Camera Control

Question	Controller E *(mean)*
Rate your ability to complete each of the following tasks for the surveyed controller:	
Move in the correct direction	3.52
Avoid obstacles	3.6
Identify any features that might have an adverse effect on the ability of the robot to maneuver through the course	3.8
Anticipate whether the turn radius of the vehicle will allow a turn	3.36
Identify if you are on the course	3.92
Maintain control when driving at slowest speeds	4.16
Maintain control when driving at medium speeds	3.46
Maintain control when driving at fastest speeds	2.6
around obstacles	3.8
Overall ability to perform driving tasks	3.68
Pan camera	4
Tilt camera	4.2
Identify which direction the camer is looking with respect to the robot	4.24
Scanning surroundings	4.2
Capturing still photograph	4.54
Overall ability to perform camera manipulation tasks	4.2
Overall controller usability:	4.04

Figure C.20: Post-Iteration Questionnaire: Phase 3 Results (Camera & Driving)

Driving/Camera control comments (Controller E):

- Sometimes it didn't recognize when I was moving the iphone to move the camera.
- Upon using the hard button for the camera, some weird effects were found whereby the camera was panning in the opposite direction and sometimes not responding at all.
- I like the fact that sensitivities can now be changed to fit an individual's preferences.
- this is taking into account the multiple times i've been able to practice with this system
- Sometimes when driving the robot seemed to veer left when I was trying to go straight forward. Some of this could have been operator error but I could not seem to ever fully adapt to it drifting left
- the hardest thing was anticipating the lag
- "I felt moving the camera in ""real time"" made it more difficult. If I could just push a button that would make the camera automatically scan in a predetermined way all the possible angles it could scan with respect to my position, it would be easier."
- I thought the camera was mounted a little differently today, because I was not able to see the front end of the robot. This played into my ability to detect how far my nose or tail was from surrounding objects. Also, I chose the camera joystick because it is quicker to scan with less physical requirements of the user, but it still seemed a little jumpy, even as I played with its size.

Controller E Most Important Customizations:

First:	**# of responses**
Senstivity	13
Mode	5
Joystick size	2
Tilt responsiveness	3
Control location	1
No response	1
Second:	
Senstivity	11
Mode	3
Joystick size	
Tilt responsiveness	8
Control location	1
No response	2

Figure C.21: Controller E: Most Important Customizable Control Options

C.6 Post-Experiment Survey Results

C.6.1 Phase 1

Users took post-experiment surveys at the conclusion of each phase, primarily to gather information specific to user preferences. Questions were both free-text answer as well as multiple choice (e.g. Controller A vs. Controller B).

Phase 1--Post-Experiment Survey

Question	Controller A *(as % of total)*	Controller B *(as % of total)*	
Which controller did you prefer overall?	36%	64%	
Which controller did you prefer when performing the following tasks?			
Navigating around obstacles:	44%	56%	
Driving straight along course:	24%	76%	
Driving through a turn:	36%	64%	
Viewing video feedback:	32%	68%	
Negotiating the robot through confined areas:	48%	52%	
Holding controller with one hand:	68%	32%	
Holding controller with both hands:	16%	84%	
	One-handed	**Two-handed**	**No preference**
When considering the advantages and disadvantages of one-handed vs. two-handed controller operation, which do you prefer and why?	12%	76%	12%
During one-handed operation, do you prefer the controller in landscape or portrait mode?	36%	44%	20%

Figure C.22: Post-Experiment Questionnaire: Phase 1 Results

Overall comments regarding user preference:

- Overall, I preferred the tilt-based controller due to the easiness of maneuverability at higher speeds; I did, however, find the faster speeds a little more difficult to obtain with the tilt-based controller compared to the virtual joystick. The finger(?) touch method was a little inconvenient with the tilt-based in that I quickly experienced discomfort in the arm and wrist.
- I thought the joystick was a little more accurate but I have always preferred tilt based in the past. I even felt myself trying to tilt when using the virtual joystick since that's what I'm so used to.
- Tilt-based controller was the most natural interface for me. I felt that the virtual joystick was not response enough so that when I wanted to move the tank slowly, I invariable had to move the joystick so far in one direction that the tank moved in lurches.

- (tilt) easier to use
- (tilt) Due to feeling more 'in control' from the feeling of less sensitivity
- Tilt-based was easier to understand sensitivity of controller but harder to move at higher speeds
- the tilt based one seemed to have inconsistent turn speeds
- I'd say that a combination of the two would be great. tilt is excellent for going forward and back. joystick is good for turning
- (tilt) This controller is much easier to control the robot.
- Tilt-based was more intuitive; less focus has to be rationed toward rule-recall
- If the R/C tank had had a more smooth control system, the disparity between these would probably be lower
- Both felt jerky but I think that it was a limitation of the robot and not the controller. Tilt based seemed smoother and more intuitive.
- I preferred the joystick although it seemed somewhat more erratic. The tilt based input was consistent but not as easy to use.
- The tilt controller was easier to drive with while going straight and making small corrections. The joystick made turning in place easier.
- The green/red light can be used to guide the robot toward an object. I did not realize this until late in the experiment.
- (tilt) Because this method is so similar to driving, it was much easier for me to adapt to.
- (tilt) It was more predictable. It was easier to go forward and to make small adjustments in direction.

C.6.2 Phase 2

Phase 2--Post-Experiment Survey

Question	Controller C *(as % of total)*	Controller D *(as % of total)*
Which controller did you prefer overall?	56%	44%
on a scale from very unimportant to very important	***mean***	
This task required you to manipulate the camera independent of driving the robot, meaning you could not do both at the same time. How important do you feel SIMULTANEOUS control would be--ability to control the camera while driving?	2.80	
How large a role did auditory feedback (e.g. being able to hear the robot's motors engage) play in your experience with the robot?	3.72	
How important is the need to customize the camera's neutral position as it refers to the position the camera is in for driving?	3.24	
When using the camera, it could either remain pointed where the user last positioned it before releasing the controls OR automatically return to driving neutral when the camera switch/joystick is released. How important is the need to choose which of these behaviors a user prefers?	3.24	
Would you prefer visual feedback indicating when the camera is approaching its far left, right, top, and bottom limits? This would likely be done via colored lights along the edge of the display for which a limit is being approached. Indicate the level of importance of such an indicator.	3	
Would you prefer a choice of the type and/or location of deadman switch for tilt-input activation either for camera control or driving?		
Camera Switch: Location on Screen	2.84	
Camera Switch: Type (hard/physical button vs. touchscreen button)	2.64	
Driving Switch: Location on Screen	2.96	
Driving Switch: Type (hard/physical button vs. touchscreen button)	2.8	
How important is it for the user to be able to set the sensitivity of their controls?	4.52	
How important is it to be able to choose the type of control used for each channel. For instance, choosing between joystick and tilt controls for driving and camera, to make your own combination based on your own preferences?	4.28	

Figure C.23: Post-Experiment Questionnaire: Phase 2 Results

Overall comments regarding user preference:

- The joystick camera was easier for me to control because I felt like I was moving a "head" around to look at things, so to speak. That is, compared to the tilt-based camera, which would seem like I had to use the whole "body" to move the field of view.
- Felt joystick was more accurate and repeatable
- Joystick usage was more intuitive and didn't cause me to move more than necessary
- I did not have to turn my head with the joystick camera
- much simpler/more intuitive to use
- I was physically moving a lot more with the tilt based camera but was more relaxed and confident

- (tilt) It is much easier to control the camera.
- My largest frustration with the tilt-based controller when used for motion was the combination of high-sensitivity and absence of boundaries. This frequently caused dramatic over-steering. However, the boundary of motion in-place when using the camera improved usability immensely. Unfortunately, this is unlikely to be applicable to locomotive control. =[
- Switching between tilting for driving and tilting for looking kept me in a constant state of minor confusion; reduced confidence. I was far more inclined to use the camera while moving between objectives when it was on the joystick.
- The joystick was more intuitive, but the tilt-based was easier to control.
- The tilt controls were much smoother than the joystick camera controls.
- (joystick) Well, I really like both of them, but because I did well in the first one. that is not mean the second one is bad. I believe both of the applications are good.
- the joystick camera is very hard to maneuver smoothly
- I enjoyed both options, but simply because the joystick was really touchy and jolted all over the place, I'd prefer a softer and smoother scanning method.
- Even though the joystick was easier, the tilt method allowed me to easily identify the robot's direction vs the camera's direction.
- (joystick) For me it was easier to pan and tilt the camera with this one.
- The tilt-based camera was easier for me to control.

Describe your scanning strategy:

- I did not use the camera at all for driving tasks. I tended to scan where I had a feeling the ball would be, mostly moving from upwards left to right and then downwards.It was imperative for me to position the car correctly to scan most effectively.
- I only used the camera pan/tilt for searching for balls, not for driving. I would go into a box then scan left to right within the camera's range, then rotate based on layout of room and continue.
- I imagined I was a mom hiding a toy from a child and looked in the locations I would have placed the toys first.
- I started scanning 180 from the direction I entered the box starting low and going high
- Moved in a box, went one direction, then up, then over, then back down. To ensure I didn't miss anything I then did a second more random look around before repositioning the robot.
- Since I knew which direction the robot would be going after I completed the task, I tried to maneuver the robot the other direction so that I would end up pointing in the correct direction.
- i never used the camera scan for driving; never had to
- I used the camera for driving tasks occasionally but not often. I used the camera to know my location on the map. My scanning depended on the environment. In a room with a lot of space on the floor, I started scanning low and then went hight. For a room with high walls and very little floor space, I scanned high then low. Whether I went left to right or right to left was entirely random

- I start to scan left to right from high to low places.
- While navigating, I did not take advantage of the ability to pan and tilt the camera. It remained static for the entire duration. However, upon reaching the "viewing box," I believe I scanned from high-to-low and left-to-right. Moving the camera while driving could be more efficient, but with my driving skill I would not have been able to effectively use them at the same time. If I were more experienced with driving then I would label it as "Important."
- Auditory feedback played an important roll in letting me determine approximately "how on" the motors were before they started moving...Used when trying to move slowly drive to the boxes, stop, scan low to high and then left to right, rotate the robot in the boxes, scan again until the targets are in sight.
- I drove in short to medium bursts, seldom if ever using the tilting or panning of the camera. Once I got to a scanning area I would scan around, hitting corners first, then looking in the middle. After a thorough search I would reposition the robot and scan some more.
- Upon entering the scanning-box, I would immediately scan what I could see, moving from low to high, then work counterclockwise until I found the ball. Orienting it to the map, I usually took second to find how close each object was to the course markers.
- The camera's default position was adequate for most of the driving. I used it once to see how I was stuck on a wall and I was able to look directly at the wall since I could tell which side it was on when I hit it.
- Rarely scanned with the camera when driving the robot. It is fairly intuitive to drive and understand where the tracks and front of the robot were. When scanning I usually looked up before left or right. With the tilt controlled camera I had more trouble determining where the end of the camera range was
- I did not really have any strategy planned for moving the camera around, i just picked a location and looked around
- scan from initial robot position left to right, top to bottom. Next rotate the robot in the direction required to make the next turn 120 degrees, scan, and so on. The first time i scanned, it was pretty arbitrary just to get used to the controller in a real life situation.
- i tended to scan low left to right then high right to left
- I tried to drive in straight lines to my destination and make all turns at 90 degrees whenever possible. I did try to scan left to right. I used the camera whenever I was lodged against a wall or identifying an object.
- I used the camera while driving only when I realized I was not moving in the direction I was hoping. This allowed me to see that I was stuck on a corner or slightly against a wall. For scanning, I started at the robot's eye level and moved up, because this initial level showed everything I needed at the lowest point. I didn't notice if I moved left to right or right to left, but I generally started in the center and worked my way to the maximum height, and then began scanning.
- I did not move the camera when not scanning. I would start scanning in the direction I was facing, then move the robot at about 60 degree intervals to the right. This allowed me to recheck some of my previously scanned area.
- I didn't use the camera for driving, only for scanning. I remembered enough from phase one to navigate the course without need to use the camera for scanning. Once in the scan box I'd scan as far as I could, re-orient and then scan again until I found the object.

- I didn't use the camera's mobility except for the scanning portion. I can't think about moving two things at once. When I was ready to scan, I usually scan the entire field of view available at the position where I stopped the robot, hoping to get lucky. If not, I would try to turn the robot 120 degrees or so to get the adjacent field of view. I repeated this until I found the target.

C.6.3 Phase 3

Phase 3 post-experiment results were already presented, primarily, in the section on Post-Iteration surveys. The only remaining questions asked users to identify options that they *couldn't* customize, but wanted to; also, what changes they anticipate making when adapting for gloved control (an exercise executed at the end of Phase 3 trials).

Were there any options that you would like to have customized but could not?

- Yes, I would prefer to have two joysticks, one on each side. Left side controls camera, right side controls driving. Preferably, you could use them at the same time, but even not, I'd prefer to have them be different places on the screen instead of a switch.
- The ability to limit the joystick when moving the camera. For a mission that only requires you to search an area, it is unnecessary for me to be able to view directly down at the top surface of the robot.
- sensitivity of the turn speed; it was so sensitive i had to come to a complete stop before every turn
- cruise control
- A non-linear mapping from the joystick position to acceleration may be useful. One minor grievance was the inability to scale obstacles, such as the entry ramp, when at lower accelerations. Similarly, navigation was dramatically easier with the acceleration at lower settings, but the artificial cap on acceleration was irritating during the long stretches of open track.
- Turning as tilt and throttle as joystick (motors only run when throttle allows; no turning even at full tilt if throttle is down)
- Tuning the balance between the motors.
- to move the camera and leave it in position while driving
- I would like the camera pan to be automatic.
- Where the camera is actually mounted, and it's original line of sight. This only comes to mind because I noticed a slight difference in the driving view from previous tests. If camera and robot motion could happen simultaneously, this would not be an issue, but if I were able to choose what my driving view was I may stop less often to check my surroundings.
- I think there might have been too many customizations available. Most would not make sense to the normal user without introduction.

If you were told you must wear gloves designed for use with touchscreens while controlling your robot, would that impact the controller settings you chose? If so, what do you think you would do differently?

- No. (*8 responses*)
- If the gloves worked well, I wouldn't change anything. If they didn't, I would probably use a hard button with tilt for driving.
- I would probably use the tilt function instead since the gloves wouldn't allow me as fine a control with a joystick.
- probably not. I preferred slow and slightly sensitive settings. The slower the settings, the more control I felt I had on the robot.
- Perhaps; it would take a little getting used to, since the 'touch' would be a bit off
- No I would not change the settings. I primarily used the tilt and hence cannot imagine that the gloves will affect my control
- the controller should be easy to control. I will not choose the joystick controller. I will choose the other one.
- Not likely. Neither of the available controller schemes are tactile. I suspect wearing gloves would have little to no impact on one's driving performance.
- yes, it would affect the sensitivity
- Yes, hard button for everything. I would probably still use the dual joystick controller, but I would consider a tilt-based controller if it meant that I would no longer need gloves and the gloves were uncomfortable.
- Not really. I would prefer not to use gloves because i feel it would add a degree of separation from the controls, but overall it wouldn't affect my ability to operate the robot.
- It most likely would but how it would depend on how the gloves affected the soft control button.
- It depends on the type of gloves. If they were designed for use with a touchscreen, I may not have to do anything different.
- I've had no experience with gloves that allow touchscreen action, but I could imagine that with a little testing and practice I would still be able to use the touch screen control settings I've chosen. If the gloves were difficult for me, I would switch to hard button controls.
- I would not use virtual joysticks at all.
- No, I think I could still operate the joystick and camera controls with gloves that are designed for use with touchscreens. Hard to say for sure without trying it out, though.

C.7 Other Results

The final results presented are a by user account of the custom options selected (Figure C.24) and their preferences from Phase 1 to Phase 3 (Figure C.25).

	Default	15	25	20	21	6	4	3	17	7	9	22
Ungloved												
accel_pref	**0.4**	0.286	0.100	0.4	0.247	0.100	0.247	0.353	0.4	0.4	0.4	0.347
camera_button_type	**1**	1	1	1	1	1	1	1	1	1	1	1
camera_mode	**1**	1	1	1	2	2	1	1	1	1	1	2
camera_switch_location	**1**	1	1	1	6	1	3	1	1	1	1	6
driving_button_type	**1**	1	1	1	1	1	1	1	1	1	1	1
driving_mode	**1**	1	1	1	1	1	1	2	1	2	1	1
driving_switch_location	**3**	3	3	3	3	3	6	3	3	3	3	3
joystick_scale	**1**				1	1.27		1.5		1		0.768
responsiveness_straight	**0.45**	0.278	0.290	0.361	0.45	0.327	0.386	0.493	0.45	0.45	0.45	0.420
responsiveness_turn	**0.45**	0.290	0.211	0.366	0.390	0.290	0.302	0.505	0.383	0.45	0.45	0.448
# of settings changed		3	3	2	4	5	5	5	1	1	0	6
Gloved												
accel_pref		0.286	0.340	0.4	0.247	0.100	0.247	0.353	0.4	0.4	0.4	0.347
camera_button_type		1	1	1	1	1	1	1	1	1	1	1
camera_mode		1	1	1	2	2	1	1	1	1	1	2
camera_switch_location		1	1	1	6	1	1	1	1	1	1	6
driving_button_type		1	1	1	1	1	1	1	1	1	1	1
driving_mode		1	1	1	1	1	1	2	1	2	1	1
driving_switch_location		3	3	3	3	3	3	3	3	3	3	3
joystick_scale					1	1.266		1.5		1		0.883
responsiveness_straight		0.2777	0.192	0.361	0.45	0.327	0.386	0.493	0.45	0.45	0.45	0.420
responsiveness_turn		0.290	0.131	0.366	0.390	0.290	0.302	0.505	0.356	0.45	0.45	0.448
# of settings changed from ungloved configuration		0	3	0	0	0	1	0	1	0	0	1

	Default	13	24	23	1	11	19	10	18	14	16	8
Ungloved												
accel_pref	**0.4**	0.453	0.338	0.353	0.404	0.307	0.166	0.233	0.100	0.4	0.150	0.302
camera_button_type	**1**	1	1	1	1	1	1	1	1	1	1	1
camera_mode	**1**	2	1	1	1	1	1	2	1	1	1	1
camera_switch_location	**1**	1	1	1	1	1	1	1	1	1	1	1
driving_button_type	**1**	1	1	1	1	1	1	1	1	1	1	1
driving_mode	**1**	2	2	2	1	2	1	2	1	2	1	1
driving_switch_location	**3**	3	3	6	3	3	3	3	3	3	3	3
joystick_scale	**1**	1.08	1.24	1.022		1.5		1.207		0.950		
responsiveness_straight	**0.45**		0.45	0.354	0.438	0.230	0.165		0.344	0.45	0.466	0.127
responsiveness_turn	**0.45**		0.45	0.600	0.426	0.198	0.209		0.336	0.45	0.377	0.130
# of settings changed		4	3	6	4	5	3	4	3	2	3	3
Gloved												
accel_pref		0.453	0.338	0.397	0.404	0.307	0.166	0.284	0.100	0.4	0.150	0.302
camera_button_type		1	1	1	1	1	1	1	1	1	1	1
camera_mode		2	1	1	1	1	1	2	1	1	1	1
camera_switch_location		1	1	1	1	1	1	1	1	1	1	1
driving_button_type		1	1	1	1	1	1	1	1	1	1	1
driving_mode		2	2	1	1	2	1	2	1	1	1	1
driving_switch_location		3	3	6	3	3	3	3	3	3	3	3
joystick_scale		1.453	1.241	1.022		1.5		1.207		0.950		
responsiveness_straight			0.45	0.354	0.438	0.230	0.165		0.344	0.45	0.466	0.127
responsiveness_turn			0.45	0.600	0.426	0.198	0.209		0.336	0.45	0.377	0.130
# of settings changed from ungloved configuration		1	0	1	0	0	0	1	0	1	0	0

	Default	12	5	2	AVERAGES	# of times changed
Ungloved						
accel_pref	**0.4**	0.261	0.295	0.275	0.293	21
camera_button_type	**1**	1	2	1		1
camera_mode	**1**	1	1	1		5
camera_switch_location	**1**	1	1	1		3
driving_button_type	**1**	1	1	1		0
driving_mode	**1**	1	2	2		10
driving_switch_location	**3**	3	3	3		2
joystick_scale	**1**		1.5	1.272	1.178	12 of 14
responsiveness_straight	**0.45**	0.536	0.332	0.535	0.382	18 of 23
responsiveness_turn	**0.45**	0.503	0.530	0.45	0.380	19 of 23
# of settings changed		3	6	4	3.520	
Gloved						
accel_pref		0.261	0.295	0.275	0.306	2
camera_button_type		1	2	1		0
camera_mode		1	1	1		0
camera_switch_location		1	1	1		1
driving_button_type		1	1	1		0
driving_mode		1	1	2		3
driving_switch_location		3	3	3		1
joystick_scale			1.5	1.272	1.215	2
responsiveness_straight		0.528	0.297	0.535	0.376	3
responsiveness_turn		0.520	0.378	0.45	0.370	4
# of settings changed from ungloved configuration		2	3	0	0.600	

Figure C.24: User-Customized Settings

Preferences (all Phases)

parID	Phase 1	Phase 2	Phase 3	
	Drive Preference	*Camera Preference*	*Drive Mode*	*Camera Mode*
1	tilt	joystick	tilt	tilt
2	joystick	joystick	joystick	tilt
3	tilt	joystick	joystick	tilt
5	tilt	joystick	joystick	tilt
6	tilt	joystick	tilt	joystick
7	joystick	joystick	joystick	tilt
10	tilt	joystick	joystick	joystick
11	tilt	joystick	joystick	tilt
12	joystick	joystick	tilt	tilt
13	joystick	joystick	joystick	joystick
17	tilt	joystick	tilt	tilt
19	tilt	joystick	tilt	tilt
21	tilt	joystick	tilt	joystick
24	joystick	joystick	joystick	tilt
4	tilt	tilt	tilt	tilt
8	joystick	tilt	tilt	tilt
9	tilt	tilt	tilt	tilt
14	joystick	tilt	joystick	tilt
15	tilt	tilt	tilt	tilt
16	joystick	tilt	tilt	tilt
18	joystick	tilt	tilt	tilt
20	tilt	tilt	tilt	tilt
22	tilt	tilt	tilt	joystick
23	tilt	tilt	joystick	tilt
25	tilt	tilt	tilt	tilt
Joystick:	**9**	**14**	**10**	**5**
Tilt:	**16**	**11**	**15**	**20**

Figure C.25: Preference Comparisons (by User), Phase 1-3

Appendix D

DVD

This section describes the digital folders and files available on the enclosed DVD. Questions regarding its content may be referred to the author at the permanent e-mail address: `amw91682@gmail.com`.

D.1 WiRC

The folder labeled WiRC houses the entire XCode project (requiring XCode for build settings and the compiler) as well as .h and .m header and implementation files able to be opened in any C viewer or text editor. The application architecture is described in detail in Chapter 3, and should be referenced where code comments alone do not suffice. The WiRC manuals, custom protocol definitions, and C-based desktop client are also included.

Folder list:

1. *AttitudeTest*. The code written for motion handling and quaternion manipulation.

2. *Control+IASK*. The **default** controller application.

3. *InAppSettingsKit*. The open source project used to handle the customizable

settings interface.

4. *Joystick v1.* The basic joystick control interface, using cocos2d.

5. *WiRC_iOS_SDK_original.* The software development kit provided by Dension, upon which the attitude aware controller was built.

D.2 Experiment Design

The folder labeled Experiment Design includes pertinent Institutional Review Board (*IRB*) documents as well as survey instruments, cataloged by phase (*Survey Instruments*).

D.3 Experiment Results

This folder includes the raw data presented in Appendix C, in spreadsheet form.

D.4 Videos

This folder holds a couple of sample videos displaying the robot and controller in action. *Ph1Tr7.mov* is a sample of the video feedback available from a timed trial.