1-1-2003

Virtual reality aided vehicle teleoperation

Bryan Eugene Walter
Iowa State University

Virtual reality aided vehicle teleoperation

by

Bryan Eugene Walter

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Mechanical Engineering

Program of Study Committee:
Adrian Sannier (Major Professor)
James Bernard
Carolina Cruz-Neira
Jim Oliver

Iowa State University

Ames, Iowa

2003

Graduate College
Iowa State University

This is to certify that the master's thesis of

Bryan Eugene Walter

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Teleoperation, the operation of a vehicle from a distance, is a useful but difficult task. The single most complicating factor is lag, the communications delay that arises when signals must travel between the operator and the vehicle being controlled. This lag can be on the order of ten seconds if the signal must travel to a vehicle orbiting the Earth or even ten minutes if the signal's destination is Mars. Lag delays the execution of commands that the operator wishes to give to the vehicle. To complicate matters, it also delays the delivery of any information about the vehicle's state to the operator. For example, if a vehicle is heading towards an obstruction observed by a vehicle-mounted camera, the operator will not receive the signal until a number of seconds equal to the lag has passed. Even if the operator responds to the obstruction immediately upon receipt, any command intended to execute an evasive maneuver will be subject to the same delay. Thus, before any corrective measures can be taken, the vehicle continues to move towards a crash for a time approximately twice the lag. For an earth bound operator driving a vehicle on Mars, that would mean around twenty minutes of movement. The standard way to cope with these delays is to only allow the vehicle to move in small increments when it gets a command so that it sits idle for the majority of the double lag cycle before another operator input can be received and executed. While effective, this method of control is slow, frustrating, tedious and error prone.

This thesis shows that simulation methods in concert with a virtual environment can markedly improve upon an operator's ability to teleoperate a vehicle in certain applications. If the environment where the vehicle will be operating can be adequately geometrically modeled and is relatively static, an operator can control a simulated vehicle in a virtual version of the environment without experiencing any of the delays associated with lag. The results of the simulation are used to control the real vehicle from a distance. The simulated vehicle is based on a dynamics simulation of the real vehicle that is used to predict the position of the real vehicle resulting from the sequence of operator inputs. A test of this

approach using low cost, commodity components suggest that the effects of lag can be mitigated, providing a more continuous operating experience. Updates from the real vehicle are incorporated into the virtual world to provide cues to the operator about how closely the real vehicle is following the simulation. If a method is also included to minimize the distance between the real vehicle and the virtual one, virtual reality (VR) aided teleoperation can significantly improve vehicle control.

This thesis presents a VR aided teleoperation system that uses the wagon tongue dynamics method to control the distance between the real and simulated vehicle positions. This distance is also used to provide real-time feedback to the operator about the predicted degree of uncertainty in the vehicle position. An uncertainty box is drawn around the virtual vehicle that grows and shrinks in proportion to the distance between the path of the real vehicle and the path of the virtual vehicle. A test of this VR aided teleoperation system designed to compare its effectiveness with that of camera aided teleoperation and non-lagged direct control is also described. The results of this test show that VR aided teleoperation significantly outperforms camera-aided teleoperation, and is nearly as effective as direct, unlagged control.

MOTIVATION

A distant planet requires surface exploration, an unstable building needs to be assessed, people trapped in a mine need to be found. Remote controlled vehicles can accomplish these tasks and more through teleoperation. Teleoperation is the human mediated control of a robot from a remote location. The operator must complete any tasks without being able to directly observe the robot or its environment. Vehicle teleoperation, a specific type of teleoperation, is the remote control of a mobile robot or vehicle. In this thesis, the terms vehicle teleoperation and teleoperation will be used interchangeably. Teleoperation is common in applications where direct human involvement is dangerous, expensive, or impossible. These types of applications are frequently set in dynamic environments or require the robot to perform complex functions. Many of these complex tasks and dynamic environments would render autonomous robot design infeasible with current technology [1]. It is precisely these cases in which a human must be in the loop of control. This human mediation is what separates applications for teleoperated robots from those for purely autonomous robots.

Examples of Vehicle Teleoperation

Search and Rescue Robots

A dramatic example of a teleoperation occurred on September 11, 2001. A few hours after the terrorist attacks on the World Trade Center, the Navy, private companies, and the Center for Robot Assisted Search and Rescue (CRASAR) all sent teleoperated robots to the site to search for victims [2]. The CRASAR robots varied in size from a shoebox to a large suitcase and provided two-way communication, heat detection, and site visualization via color video camera feedback. These robots combed the rubble as commanded by their

remote human operators, and ultimately discovered the bodies of three victims. The robots accomplished this without exposing their operators to risk.

Chernobyl Explorer

Another example of teleoperation can be found in the aftermath of the disaster at Chernobyl in the Ukraine. In 1986, the nuclear reactor there suffered a catastrophic meltdown and to prevent further radiation exposure, the reactor was encased in a concrete sarcophagus. Since then, the sarcophagus has begun to develop cracks from weather and age, creating a potentially dangerous situation if the integrity of the sarcophagus were ever compromised. To complicate matters, the interior of the power plant is still radioactive enough to make direct human exploration dangerous. To avoid these risks, a teleoperated robot was sent into the plant interior to map out its current condition [3]. The information gained from this expedition has given scientists a more complete picture of what has happened inside the plant since its failure. The robot also took radiation data and photographed the site, providing valuable information to the scientists and engineers formulating plans for repairs to the sarcophagus.

Unmanned Aerial Vehicles

Another arena that has seen the successful application of teleoperated vehicles is aerial reconnaissance. The development of an inexpensive unmanned reconnaissance vehicle was accelerated by US Military operations in the 1980s in Grenada, Lebanon, and Libya [4]. In December 1985, the first unmanned aerial vehicles (UAVs) were commissioned by the military. An example of this model is shown in Figure 1. UAVs rely on a crew of four to eight ground-based operators who monitor the vehicle's position and

condition via video feeds from vehicle-mounted cameras and additional sensor data. The crew is responsible for piloting the UAV to its target safely through hostile territory, identifying the target, and capturing a good image of it. Of course, the crew performs these tasks in a room far away from the actual target, safely out of harm's way.

UAVs proved their worth in Operation Desert Shield/Storm in Kuwait and Iraq in the early '90s. The UAV was praised by Lt. Gen. Boomer of the Marine Corps Central Command Element Headquarters as "... the single most valuable intelligence collector" [4]. UAVs have continued to prove their value in the recent operations in Afghanistan and Iraq.

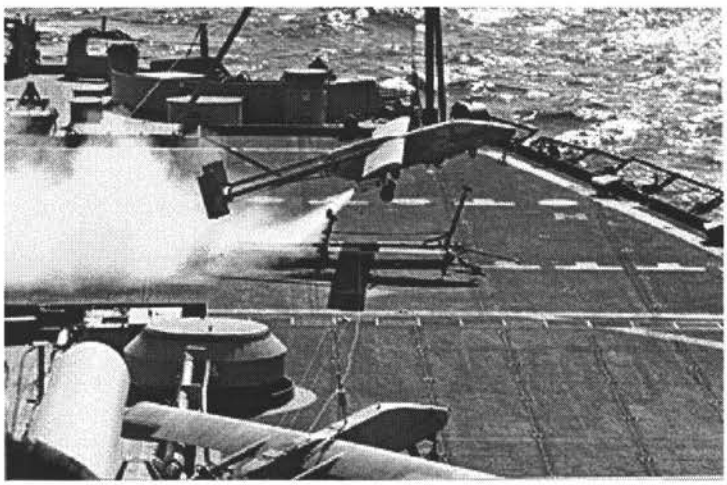

Figure 1: An Unmanned Aerial Vehicle

Planetary Rovers

NASA has also been a leader in the development and use of teleoperated vehicles. Perhaps the most well known is the Mars rover, Sojourner, shown in Figure 2. Sojourner was a six wheeled solar powered robot covered with scientific sensors. It was launched from Earth as part of the Pathfinder mission in December 1996 and arrived on Mars in July 1997. Its deployment plan called for it to stay within ten meters of its Lander and use its alpha-proton-X-ray spectrometer to determine the composition of rock on the Martian landscape

[5]. Because commands took anywhere from ten to fifteen minutes to reach Sojourner from Earth, it used sophisticated autonomy software. However, it still heeded human commands, making it an extreme example of teleoperation. Sojourner's mission lasted seven days, twelve times longer than planned, before communication was lost [5].

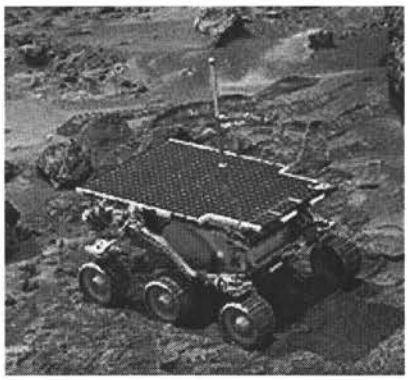

Figure 2: The Mars Rover Sojourner

The Challenge of Teleoperation

Despite the successes of teleoperation, it remains useful in only a handful of situations and it is a cumbersome method of control from an operator's perspective. Teleoperation is made difficult primarily by an unavoidable complication - lag.

Lag and the Ansible

Orson Scott Card was keenly aware of the problem of lag when he wrote the popular science fiction novel, *Ender's Game*. Teleoperation is at the heart of the story's plot as the main character, Ender, is responsible for teleoperating a fleet of remote starships in deep space. For any signal carrying one of his commands, physics dictates that it can travel no faster than the speed of light. Indeed, achieving this speed of signal propagation is difficult and is beyond current technology. For Ender, even the speed of light would not have been fast enough as he was responsible for giving orders to fleets of starships tens of light years

away. Even assuming signal propagation at the speed of light, his signals would have taken decades to reach the ships and any information from the fleet would likewise have taken decades to return back to Ender. Following these physical restrictions, Ender could have had only one back and forth conversation with the commander of the farthest fleet during his entire lifetime.

Card's answer to this dilemma was something he called the ansible, a device that uses an as yet undiscovered property of the universe to eliminate lags in communications [6]. With the ansible, Ender was able to reach out across the cosmos and talk to ship commanders as if they were on the other side of the table. Unfortunately, while science fiction writers have the luxury of inventing magical devices that bypass the laws of physics, remote applications in real life must directly address the challenges of lag.

Garden Hose Exercise

Anyone who has filled up a fish tank with a garden hose has dealt with the challenges of lag. In this exercise, it is temperature, not electronic signals that are lagged, but the principle is the same. A fish tank requires that the water be at a particular temperature before the fish may be placed in it. When one person mans the faucet at one end of the hose and the other guides the water into the tank while checking the water temperature, lag becomes a critical factor. Unless they have incredible luck, the water coming out of the hose initially will be either too hot or cold. Therefore, the person at the end of the hose informs the faucet operator of this. If the water is too cold, the faucet operator immediately turns the faucet in the hot direction, but it takes time for the hot water to reach the end of the hose. In the meantime, the temperature measurer continues to call for hotter water so the faucet operator continues to turn up the heat. Finally, the hot water reaches the thermometer and it will likely be too hot due to excessive operation of the faucet. The person at the end of the hose calls for colder water and the overshoot cycle begins anew.

Effect of Lag on Teleoperation

In many common applications relying on signal propagation lag is only an annoyance. For example, in satellite TV transmission, signal lag each way means that the picture shows up on viewers' screens some seconds after it was sent out. The effects of this signal lag can be made apparent by watching a program from a satellite feed alongside the same program received via an antenna or cable. This setup provides an amusing way to accurately predict the outcome of the game winning field goal with 100% accuracy! The effects of the lag on the TV feed are not important because the viewer does not rely on the timeliness of the signal to make critical decisions.

Lag is a severe problem in many vehicle teleoperation cases however. A two second lag means that if the vehicle should ever get into a position such that it will crash within two seconds, then it is lost, since any avoidance maneuver by the operator will take at least two seconds to reach the vehicle.

Consider a car with a steering wheel that delays driver inputs for two seconds before acting on them. Imagine that you are driving this car down a busy street when another car in the next lane suddenly swerves into your path. In a normal car, you could easily brake and steer away to avoid disaster. Unfortunately, with lagged controls, anything you do in response to the lane change will still result in a crash because no matter what you do you cannot change the car's behavior for the next two seconds.

This situation describes only half of the problem. In the scenario just described, the driver's inputs were lagged, but the information about the world surrounding the vehicle was not. Now imagine that you are controlling the car with the delayed steering wheel from your house using a remote control and a set of video cameras to see the world around the vehicle. When both the controls and video feeds are lagged by two seconds, the merging vehicle would hit your car before you even became aware that it had started to swerve into your lane!

Three characteristics determine how profound the effect of lag can be on a particular vehicle teleoperation application. These are the magnitude of the lag, the speed of the vehicle, and the predictability of the environment. If the lag is not of sufficient duration, its effect will be minimal. However, sufficient duration is different for each application and is directly tied to the vehicle speed and the dynamic nature of the environment. If a vehicle is traveling at a constant 0.1 meters per second, a one second delay results in a maximum positional error of 0.1 meters per command. In most applications, such an error is not a huge concern. This same lag though could easily present serious difficulties if the vehicle were traveling ten meters per second. However, if the environment is calm and predictable, an operation envelope of ten meters may be acceptable. Typically, knowledge of UAV position within ten meters is less critical, because reconnaissance aircraft do not frequently fly close to each other and there are not many other objects to hit at high altitude. Operation envelopes are far more critical for ground vehicles as they typically get much closer to each other and there are plenty of non-vehicle objects to hit.

Another difficulty presented by lag is that it is typically not constant. This means that if a set of commands is given at one-second intervals, they are not guaranteed to arrive at the vehicle one second apart. This characteristic is particularly troublesome because instead of simply delaying the vehicle's response, it alters it. For example, with a non-constant lag a vehicle that is told to travel forward for one second at 2 meters per second and then told to stop might instead travel forward for 1.2 seconds before receiving the command to stop. As a result, the operator would believe the vehicle had traveled two meters when it had actually traveled 2.4 meters. As commands are strung together, these inconsistencies can add up to large errors in position. These discrepancies can lead the operator to assume that they have succeeded in stopping their vehicle, when in fact they have sent it plunging into a pit. Figure 3 illustrates this situation.

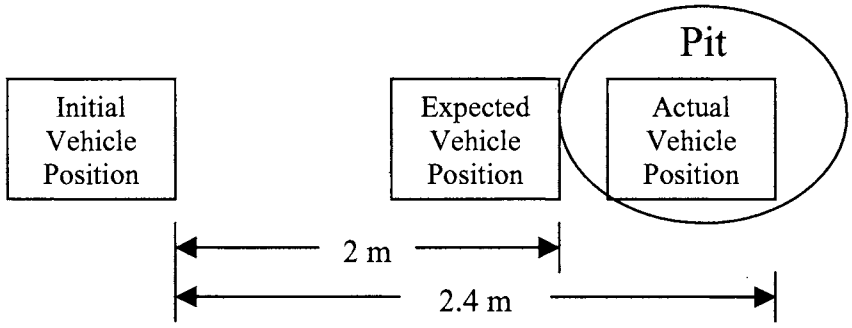

Figure 3: The Dangers of Non-Constant Lag

Challenge of Limited Field of View

Lag is not the only challenge presented to the operator of a teleoperated vehicle. All of the operators' knowledge about the environment must be mediated since they are remote from it. Typically, an operator views the vehicle's world through the limited field of view (FOV) of a video camera. Today's highest quality vehicle-mounted cameras provide at best a 120-degree arc, leaving a 240-degree blind arc around the vehicle [7]. Most cameras provide much less FOV than that, with a typical FOV being anywhere from 30 to 90 degrees [7]. Even if the camera can be rotated such that the whole FOV around the vehicle can be seen, only a portion of it can be seen at any one time. This limited FOV causes difficulties because it provides a much smaller visible area than humans are accustomed to seeing with their nearly 150 degree FOV. The limited FOV provided by the Predator UAV camera has prompted its operators to liken flying it to "flying through a straw" [8].

One common way to address the limited FOV problem is to mount multiple cameras in strategic locations on the vehicle to provide a full FOV at all times. The problem with this approach is that it creates the need for the operator to look at several different video feeds at once and create a consistent mental image of the world. This puts stress on the operator and takes mental capacity away from the task at hand. It also introduces a multifold increase in the bandwidth required to operate the vehicle, which would result in the requirement of more expensive, heavy or power consumptive communications equipment.

Operators with a limited FOV need to work harder to maintain an accurate mental image of the world around the vehicle. If there is significant lag, the operator must maintain this mental image for several seconds while waiting for new information. Furthermore, operators must remember what they can about the environment they have passed and is no longer visible. If the vehicle passes a rock on its left, the operator must remember that the rock is behind and to the left of the vehicle just in case a maneuver must be made in that direction. These distractions and mental tasks both inhibit the operator's ability to perform the relevant teleoperation task and increase the rate of operator fatigue. As a result, a team of operators is often needed to control one vehicle. In the case of a UAV, a crew of anywhere from four to eight is required depending on the difficulty level of the mission [9].

Related Research

Several research teams have tackled the challenge of teleoperation and devised systems to improve performance in various teleoperation tasks. Other teams have focused on making their systems as robust and simple as possible because their applications risk expensive equipment and human life. Some of this research is presented here in summary. Most of the systems cited employ advanced user interfaces and several use virtual reality (VR) to aid teleoperation. The Summary of Virtual Reality Section presents a brief discussion of VR for readers unfamiliar with it.

ARGOS

ARGOS, the Augmented Reality through Graphical Overlays on Stereovideo system was designed for teleoperation in environments that are unknown or rapidly changing [10]. It used augmented reality (AR) for the operator display. AR shares many qualities with VR, with one important difference. Instead of providing graphical tools to the user in a totally computer simulated world, AR paints graphics on top of a real world picture from a video

camera feed. The goal of ARGOS was to provide "... good visual, auditory, and perhaps even haptic cues ... so that the operator will have a sense of the remote environment" [10]. A camera mounted above and behind the robot on a pole provided a view of the environment. ARGOS used the robot for calibration because its dimensions are known and it was always present in the scene. Once calibrated, the system could overlay depth cues on top of the video feed to help operators perform complex spatial tasks, all while using a simple two-dimensional display on a computer monitor.

VEVI

The Virtual Environment Vehicle Interface, or VEVI, offered a better method to view the last known status of the vehicle than a camera feed can provide. It accomplished this by creating a virtual view of the vehicle's situation in the real environment. VEVI's virtual view presented the last known vehicle state in a virtual world, but unlike the system presented in this thesis, it did not simulate or predict vehicle positions [11]. The virtual display in VEVI eliminated the limitations caused by the incomplete field of view of remote cameras. VEVI was a flexible system that was not tied to a particular vehicle or environment. Its modular design allowed for its application in underwater exploration with the TROV, crater exploration with the walking robot DANTE II, satellite maintenance with Ranger, and planetary exploration with the Marsokhod rover [11]. A screenshot of VEVI in action with the Marsokhod planetary rover is shown in Figure 4.

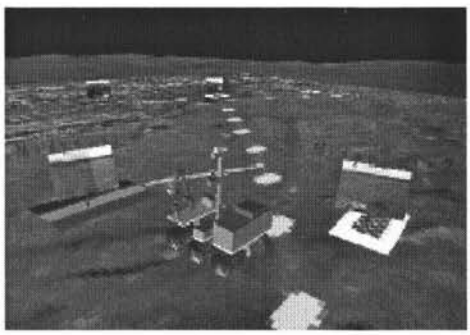

Figure 4: VEVI Interface

The Autonomy and Robotics Area of NASA used VEVI in both the Mars Pathfinder Mission and in a mission to map the interior of Chernobyl. A main lesson learned from these exercises was that "the ability to continually see all around the robot provided scientists with a more natural sense of position and orientation ... than is usually available through more traditional imaging systems" [3]. Additionally, they mentioned that "this capability ... substantially accelerated site exploration" [3].

Ranger TSX

The Ranger TSX Real-Time Visualization system was implemented for use with a shuttle-based robot arm interface [12]. The arm was used to repair satellites while in orbit. The system employed a typical virtual environment complete with a robot avatar, but it also had a novel feature in its predictive ghosting display. Whenever a user issued a command, the computer used a mathematical simulation of the arm to determine the effect of that command and then quickly displayed a translucent "ghost" version of the arm in the position likely to result. This was done to alert operators of the likely result of their action so that they could respond to mistakes before they happened. Without this capability, the operator would know the effect of their action only after it had potentially already resulted in consequences. By simulating the arm, TSX was able to provide the operator with real time error feedback. The inclusion of this error feedback feature increased operator performance in delicate tasks, and it also was a useful diagnostic tool. When the arm and program were calibrated and tested before launch, the ghost movements were aligned with the actual arm movements. As a result of this calibration, the error between the predicted position and the actual result of the command could be used as information in a real time diagnosis of the arm's operation.

Real time error feedback is an important part of the system implemented in this thesis and is described in a later section.

Teleoperation Interfaces

The Robotics Institute at Carnegie Mellon University tested two novel interfaces for vehicle teleoperation with researchers from the L'Ecole Polytechnique Federale de Lausanne in Switzerland [13]. One interface used gestural commands and the other used haptic feedback. Each interface was designed to improve the operator's performance through efficiency of design. A premium was placed on reducing the number of steps the operator must perform due to the potential effects of lag on each step. The first of these effort saving interfaces was called the GestureDriver. In this system, the operator was taught special gestures to perform with their arms and hands. Each gesture was mapped to a particular command for the robot. Image processing techniques were used on a video feed from a camera trained on the operator to ascertain the current gesture. They found that GestureDriver offered flexibility between users and allowed the user to move around between commands. However, having to remember all of the body positions for commands caused difficulty for some users and it became tiring after prolonged use.

The second interface, the HapticDriver, used a six degree of freedom (DOF) robotic arm as the input method to navigate a robot through a cluttered and confined space. This arm was equipped with force feedback to provide haptic response. This haptic feedback was manifested in the form of a push from the arm against the operator's hand to give the operator the illusion of resistance to the desired movement. The arm was constrained to a plane and an invisible barrier was modeled around the robot. A ring of proximity sensors on the robot informed the controlling software of the distances to the nearest objects. When the invisible barrier intersected an object, a repulsive force was generated in the arm. As the robot got closer, the force increased proportionately. HapticDriver was found to be an

effective interface in the navigation of cluttered environments and for performing docking maneuvers. It also improved obstacle detection and avoidance.

Robot-Centric Teleoperation Model

While developing their teleoperation interfaces, the Robotics Institute at Carnegie Mellon University also created a teleoperation system rooted in a novel philosophy. In this approach to vehicle teleoperation, the robot was made an equal collaborator with regard to its operation [14]. In this way, it could treat the human's commands as a source of noisy data and take its own view of its surroundings into account when it operated a command. In addition, if it found itself in a situation that it considered dangerous, it could alert the human operator by asking for help. This scheme implied a level of autonomy on the part of the robot; however, it was not completely autonomous. If the robot found itself in a situation in which it was unsure of the proper course of action, it needed not guess based on a decision metric. Rather, the robot could ask a human or group of humans for assistance first, guessing only if it got no response within a preset time limit or if the human input would send it into greater peril.

The main strength of this teleoperation philosophy was that the robot was capable of being a partner in its control, thereby providing a local perspective. Viewing the robot in this manner also aided multi-user collaboration since no one user was assumed to have absolute control [15]. In addition, the robot's ability to make some decisions was helpful if the lag in the system was significant. If the robot did not receive a signal for a long time, it could make its own decisions on how to avoid danger. Furthermore, the ability for the robot to treat the human's inputs as a noisy data stream addressed further difficulties posed by lag. Human operators issue commands based on decisions formed from old, lagged data, and if the situation had changed since then, carrying them out could have disastrous results. With the

robot's capability to ignore commands that would send it into peril, this system alleviated some of the problems caused by lag.

UAV Interface Research

Researchers at Wright-Patterson Air Force Base performed a study on how multi-sensory interfaces and advanced visualization techniques reduced a UAV operator's workload while improving their performance and situational awareness [16]. The work evaluated haptic feedback, head mounted displays, spatial audio, and virtual environments. The first part of the study evaluated how the use of visualization tools affected a UAV crew's ability to communicate target location. The second part studied the effectiveness of haptic cues at alerting the UAV operator of turbulence by applying resistance through a force feedback joystick. The first test resulted in crew performance improvement. The second test had mixed results as many test subjects complained that the force feedback magnitude on the joystick was set too high.

Teleoperation of Mining Robots

Mining companies have utilized teleoperated robots in search and rescue operations for more than ten years. Their method of control is relatively simple, relying on the well-established method of using a video feed from a camera and a two dimensional control panel displayed on a computer monitor. One such teleoperated robot, the Numbat, a search and rescue robot, has proven helpful in saving miner's lives [17]. In an application where advanced interfaces and teleoperation systems could be applied, they have not been applied because human lives were directly at risk. Liability and the unreliable nature of more sophisticated technologies have kept the cutting edge teleoperation technologies out of use in the mining industry. This example represents the opportunity available to new teleoperation methods if they can meet the robustness, simplicity and reliability requirements of industries

where human lives are at stake. However, any technology that would replace the current proven technology must resolve doubts about its reliability and usability.

Using Virtual Reality to Meet the Teleoperation Challenge

Summary of Virtual Reality

Many of the teleoperation systems described in the previous section employ virtual reality to improve operator performance, situational awareness and vehicle operation. Over the past decade, VR has blossomed from an immature technology to an accepted research and industrial tool. However, it has not yet reached the state where a standard definition has been forged. VR can be roughly defined as any computer simulated environment that immerses the user in a virtual world. Jerry Isdale, a leader in the VR field, defines it as "... a way for humans to visualize, manipulate and interact with computers and extremely complex data" [18]. He further defines it as "a computer mediated, 3D environment with viewer control over viewpoint" [18]. In engineering applications, virtual worlds are designed to mimic reality enough to provide a valid analysis tool with respect to what needs evaluation.

VR employs a wide range of devices to immerse application users. Common VR devices include head mounted displays and gloves, but they are not the only ones in widespread use. Other frequently used input devices are wands, tablet PCs and voice commands. Projection systems are popular display devices. In projection systems, a projector throws light onto a canvas screen in a fashion similar to the way it works in a movie theater. Projection systems can use from one to six screens and can be run by a single powerful computer or a cluster of commodity computers. A common projection system configuration is the Power Wall. This setup uses two or more screens placed side by side on a wall to create a wide display area. Another common configuration is the CAVE, which involves using three or more screens to create a room. Dr. Carolina Cruz Neira developed the CAVE for her PhD dissertation [19]. Most CAVEs have four screens, the left and right

walls, the front wall and the floor. However, a handful of six sided CAVEs have been built to provide full immersion. Typically, CAVEs employ stereo projection and polarized glasses to make the scene appear three-dimensional. Furthermore, while VR applications often deploy unique and expensive equipment, many definitions of VR would include applications that use a very common display and input device combination – the computer monitor, keyboard, and mouse.

Using VR to Aid Teleoperation

As a tool for teleoperation, virtual reality can improve the situational awareness of the operators, reduce the number of required operators, alleviate operator workload, and provide the needed information on one display surface. To use VR as an aid to teleoperation, the teleoperation task must be simulated. Two separate simulation subsystems must be designed to accomplish the overall task: the vehicle simulation, and the operational environment simulation. For ground vehicles, the simulation of the environment mostly involves creating a graphical and mathematical model representing the terrain surface that the vehicle will be traveling over.

VR improves the situational awareness of an operator by displaying a virtual version of the operational environment and allowing for an operator to select views around the virtual vehicle. In fact, the entire 360-degree arc around the vehicle can be displayed simultaneously if shown on the proper display device. In addition, unlike real cameras, the viewpoints in the virtual world can be easily changed programmatically during vehicle operation. This flexibility provides an operator with many different angles of the situation and is only possible because the virtual world is not limited by the placement of physical cameras. One popular view, easy in VR but difficult in the real world, is the view from behind and slightly above the vehicle, often referred to in the gaming world as a "chase

cam." With this view, both the vehicle and its immediate surroundings are visible to the operator.

The selectable views around the vehicle and the continuous display of the operational environment ease operator workload and lead to the reduction of the number of required operators. With the display of the virtual world, operators need not work as hard to maintain a mental image of the vehicle's surroundings. This reduction of workload allows operators to devote more of their mental energy toward performing the teleoperation task, making it easier to accomplish.

The operator workload can be further reduced if the vehicle is simulated along with the environment. Without a vehicle simulation, the common means of safe vehicular control is discontinuous and tedious. For example, when controlling a vehicle using a vehicle-mounted camera, the operator's ability to control the vehicle is affected by lag. Whenever the operator sends a command to the vehicle, it is delayed by a number of seconds equal to the lag. Any video resulting from that action is likewise delayed. Therefore, to safely pilot the vehicle, the operator must wait twice the lag in seconds between each command to see the result of each action. This is a slow, tedious and fatiguing method of control.

The inclusion of a vehicle simulation can be used to eliminate these difficulties. The operator can control the virtual version of the vehicle, which responds immediately to commands. The virtual world around the vehicle can similarly be updated instantaneously. If the remote vehicle can be made to follow the path of the simulated vehicle with a corrective algorithm, the effects of lag can be eliminated while still maintaining effective control. With VR, the operation of the vehicle is perceived as continuous despite the fact that the lag still causes the same delays in the execution of commands by the remote vehicle and the receipt of remote vehicle updates. The effects of lag are only experienced by the operator through the real time variance between the expected and actual vehicle states. This inevitable variation between the actual position of the real vehicle and the simulated world's

expectation of that vehicle's position is a result of approximations in the simulation models. This variation tends to grow with higher speeds and more severe maneuvers. Real time variance feedback alerts operators to increasing uncertainty in the actual position of the vehicle. With this information, operators can change their driving behavior, by reducing their speed and the severity of their steering, to allow the real vehicle position uncertainty to decrease. This real time feedback helps close the loop between the simulation and the real vehicle and helps reduce the error, or uncertainty zone, between the two.

VR also allows for the smooth integration of information from other sources such as radar, radiation level, temperature, or speed. This data can be fused into the simulated world to provide a single display that shows all the needed information. This fusion eliminates the need to divide attention between different displays. Indeed, these sources of information can be swapped in and out at the request of operators, allowing them to focus only on what is important.

To garner all of these advantages, the aforementioned simulation system with its vehicle and terrain subsystems must be developed. The terrain simulation is a graphical representation of the operational environment and is only as good as the quality and accuracy of the graphical model when compared to the real site. Indeed, generating realistic VR worlds requires *a priori* knowledge of the environment. This is only possible if the real world is relatively static and well known. In cases where *a priori* knowledge of the environment is not sufficiently complete, there is work being done in the field of near real-time terrain model rendering from 2D imagery. Researchers at the Computer Science Department at the University of Massachusetts have created a system that can convert satellite data into a three-dimensional graphics model in a matter of hours [20]. While this is not real time terrain generation, it does allow for virtual terrains to be created just before a teleoperated vehicle is deployed at a given site. With this ability, the virtual terrain is more recent, and therefore more useful, than a terrain modeled days, weeks or even months before.

As this work and other systems like it evolve into real time terrain generators, almost any type of terrain and operating environment will be able to be simulated and used for the purpose of VR aided teleoperation. In many cases, the terrain is sufficiently static to make teleoperation using VR practical. In the cases where it is not, work is being done to remedy the problem.

The vehicle simulation is somewhat easier to create. A vehicle simulation is a mathematical model that predicts, with reasonable fidelity, the response of the real vehicle to a specific set of inputs. Creating this simulation involves studying the dynamic response of the vehicle to its inputs and matching the mathematical model as closely as possible to the behavior of the real vehicle.

With the simulation system in place, virtual reality can be used in teleoperation applications to improve situational awareness, reduce the number of required operators, alleviate operator workload, and display various sources of information.

GENERAL SYSTEM MODEL

System Components

A new vehicle teleoperation system was created using the methods described in the previous sections for using VR to aid teleoperation. Figure 5 shows the overall layout of the system and its components in general terms, with each box representing a different hardware component.

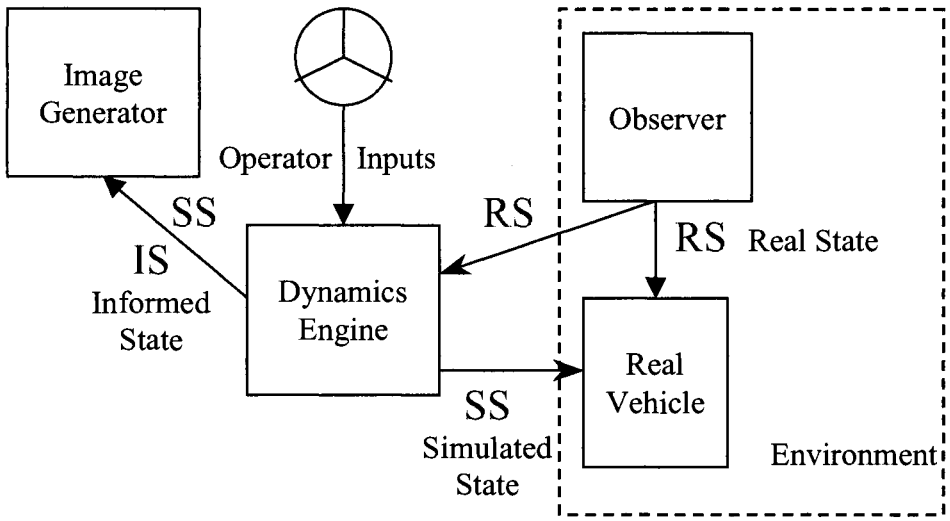

Figure 5: General System Model

The three vehicle states shown in Figure 5 are the simulated state (SS), the real vehicle state (RS) and the informed state (IS). A vehicle's state encompasses its position, orientation, speed and rotational speed. The SS is the best prediction of the state of the vehicle after the execution of a set of inputs. The RS is the actual vehicle state resulting from the execution of these inputs. The IS is the virtual world's best guess at the current state of the real vehicle using lagged state updates from the real vehicle. This state will differ from the current RS and SS because the simulation cannot match the behavior of the real vehicle

perfectly. The SS and IS are generated from the dynamics engine and the RS results from movements of the real vehicle and is calculated by the observer.

The computer at the center of Figure 5 is the platform for the dynamics engine. This engine is responsible for taking operator inputs and using them to predict the response of the vehicle to those inputs. The dynamics engine employs a mathematical model of the real vehicle to create a SS of the vehicle. The SS is used to position the simulated vehicle within the virtual environment rendered by the image generator. The simulated vehicle is displayed to the operator as a proxy for the real vehicle. The operator controls the simulated vehicle directly, without lag. The goal is to use the simulated vehicle to direct the position of the teleoperated real vehicle. Without a system in place to modify the inputs used by the real vehicle to keep it following the intended path, both the simulated vehicle and real vehicle execute the same user inputs. However, even if the simulation is a perfect match for the real vehicle, their respective behaviors could still differ due to lag. In practice, no simulation matches the behavior of a vehicle perfectly, so an input modifying method is needed to address the inevitable separations that will arise between the RS and the SS.

In order to implement an input modifying corrective strategy, the position of the real vehicle must be known. This position is obtained by the observer, which can be a camera performing optical tracking, a GPS device, a wireless internet card performing triangulation, or some similar locating method. Clearly, the observer must be located in the environment where the real vehicle is being operated. The observer determines the real state of the vehicle and sends it directly to both the real vehicle and the dynamics engine.

Because of the lag in the signal propagation of the RS when sent to the dynamics engine, the dynamics engine effectively receives a RS "from the past." Ideally, the dynamics engine would like to compare the RS to the latest SS to give the operator real time feedback regarding the separation between the real vehicle and the simulated vehicle. However, since the dynamics engine cannot know the instantaneous position of the real vehicle, it must

instead be predicted, based on the age of the latest received RS and the history of user inputs from that time to the current time. To accomplish this task, the dynamics engine stores the history of each SS and then uses this history and the latest known RS to develop an estimate of the current real vehicle position. The dynamics engine determines inputs for each dynamics time step using the same mathematical model used to create the SS. This prediction of the present real vehicle state is the informed state (IS). The IS is used to provide the operator with feedback on how far the real vehicle is deviating from the desired path.

Once the dynamics engine has calculated the latest SS and IS, it sends these states to the image generator for display. The image generator renders the virtual world, complete with a virtual version of the remote environment and the real vehicle. It also uses the distance between the IS and SS to generate real time uncertainty feedback.

System Model in Action

Table 1 shows how information and time flows through the complete system when each component communicates with the others. The requisite information for each component's calculation is shown in functional format in Equations 1-3 on the next page.

Table 1: Time Flow of System Model

Time	SS	SS History	IS	SS History Needed by IS	<<>> <<>>	RS	SS
0	SS_0	SS_0	No RS Available	-	<<>>	RS_0	No SS
L	SS_L	$SS_{[0,L]}$	No RS Available	-	<<>>	RS_L	SS_0
L+1	SS_{L+1}	$SS_{[0,L+1]}$	No RS Available	-	<<>>	RS_{L+1}	SS_1
2L+1	SS_{2L+1}	$SS_{[0,2L+1]}$	IS_{2L+1}	$SS_{[1,2L+1]}$	<<>>	RS_{2L+1}	SS_{L+1}
3L+1	SS_{3L+1}	$SS_{[0,3L+1]}$	IS_{3L+1}	$SS_{[L+1,3L+1]}$	<<>>	RS_{3L+1}	SS_{2L+1}
T	SS_t	$SS_{[0,t]}$	IS_t	$SS_{[t-2L,t]}$	<<>>	RS_t	SS_{t-L}

L = lag delay for one direction of communication
$[t_1,t_2]$ = time interval between t_1 and t_2

Equations 1 through 3 show the variables upon which each calculated vehicle state depends. The SS is a function of the previous SS and the user inputs. The RS is a function of the previous RS and the latest SS. As a result, no RS is generated until a SS has arrived at the real vehicle. The IS is a function of the latest RS and the SS history. In Equation 1, f_s is the mathematical model of the vehicle simulation. This function is also used in Equation 3 to generate the IS. The function f_r in Equation 2 represents the actual dynamic response of the vehicle. Note that f_r is a natural phenomenon and not a mathematical function defined in the system. Rather, the system tries to approximate the results of f_r with f_s. In the equations below, L is the lag delay for one direction of communication.

$SS_t = SS_0$ for $t = 0$
$SS_t = f_s(SS_{t-1}, \text{user inputs})$ for $t \geq 1$ *Equation 1*

$RS_t = RS_0$ for $t \leq L$
$RS_t = f_r(SS_{t-L}, RS_{t-1})$ for $t > L$ *Equation 2*

$IS_t =$ not defined for $t \leq 2L$
$IS_t = f_s(RS_{t-L}, SS_{[t-2L,t]})$ for $t > 2L$ *Equation 3*

Time progresses from zero at the start of the vehicle teleoperation, to an arbitrary time t and each time step is represented by a change of 1 unit. Each column in Table 1 represents data that is either generated or dispatched at every time step. All of the columns to the right of the "<<>>" in Table 1 are data items calculated at the remote environment and those to the left are not. At time zero, the operator provides the first inputs to the dynamics engine and the first SS is created. This SS is then sent to the real vehicle. The real vehicle will use this SS to generate its command inputs. However, until L time units have passed, the real vehicle will not move because it has not yet received any commands. As a result, RS_0 through RS_{L-1} are all the same state. At time L, the first SS (SS_0) arrives at the real vehicle from the dynamics engine and the real vehicle moves and generates a RS. This RS is

determined by the observer one step later and sent to the dynamics engine and the real vehicle at time L+1. The dynamics engine does not receive this first RS (RS_L) from the observer until time 2L+1. Until this first datum is received, no IS can be created. However, at time 2L+1, the dynamics engine calculates the IS as a function of the SS history from time 1 to 2L+1 and the RS it just received. The SS history for 2L time units applies because the latest RS was actually calculated at time L+1. The RS calculated at time L+1 used a SS sent at time 1 for inputs. As a result, the RS_{L+1} would need to be compared with the SS_1. Thus, the dynamics engine will always require 2L time units worth of SS to bring the RS to the current simulation time for comparison with the current SS. For all time steps greater than 2L+1, the system will generate all three states.

METHODOLOGY

The general system described in the previous section is a model, an idealization. Certain methodologies must be adopted in order to turn the model into a real system that can be tested and validated. The following four parts of this section consider the most important of these methods; a method to address the challenges of lag, a method to alleviate the errors caused by lag, a method to minimize the differences between the simulated vehicle and real vehicle, and a method to maximize field of view and operator presence. The development of each these methods and how they fit into the overall VR aided teleoperation system is discussed in this section.

Method to Address Lag

The vehicle simulation is the main tool used to address the effects of lag by providing an interface to control the lagged vehicle. The simulation helps to limit the lag because the driver operates the vehicle in the unlagged virtual world, providing continuous control. The real vehicle then determines which inputs it must use to follow the path laid out by the operator. Using the simulation to indirectly pilot the vehicle removes the fatiguing pauses caused by lagged control. The dynamics model must be capable of using operator inputs to predict the vehicle's behavior to the terrain that it is currently traveling over.

A yaw plane dynamics model is used to simplify the dynamics because the test area is a flat surface. This means that all vertical forces are ignored. As such, dynamics models such as these cannot accurately support uphill or downhill slopes. This is not a limitation of the methodology; a complete 3D dynamics model can be implemented, but a yaw plane model is sufficient to test the validity of the model.

The dynamics engine predicts the response of the vehicle to user inputs. The dynamics equations in this system are not as sophisticated as those found in commercial dynamics software. Because the test system uses a toy tank (which does not have the

dynamics complexities of a real tank), the dynamics engine uses kinematic equations of motion along with reasonable simplifying assumptions. With a dynamics model in place, positions can be simulated in the time between the receipts of RS updates to provide an approximate location for the vehicle at all times.

Method to Minimize State Differences

No mathematical simulation model, no matter how sophisticated, will be able to predict the response of a real vehicle perfectly. The basic foundation for vehicle dynamic simulation was laid in the 1950s, and while vehicle simulation has progressed greatly since then, the mathematical models still rely on simplifying approximations. These approximations result in differences between the real vehicle behavior and the predicted behavior. Over time, these accumulated differences can lead to a large discrepancy between the expected and actual vehicle position. As a result, a method must be adopted that can accommodate these differences and correct them. A number of alternatives exist. The naïve way to accomplish this task is to tell the real vehicle to go where the simulated vehicle is currently, resulting of course, in a massive overshooting of position. A more sophisticated approach would use a proportional integral differential (PID) to follow the desired path of the simulated vehicle. An even more advanced scheme would employ a genetic algorithm to determine the optimal inputs required to follow the desired path.

The method used in this thesis was originally developed by Kroll and Roland of CALSPAN in 1970 and is called the wagon tongue method [21]. One vehicle is designated as the leader and one as the follower. The method presumes that states are known for each vehicle, and that the lead vehicle is following the desired path. The method determines what inputs must be given to the follower vehicle to get it to align with the desired state. This situation is illustrated in Figure 6.

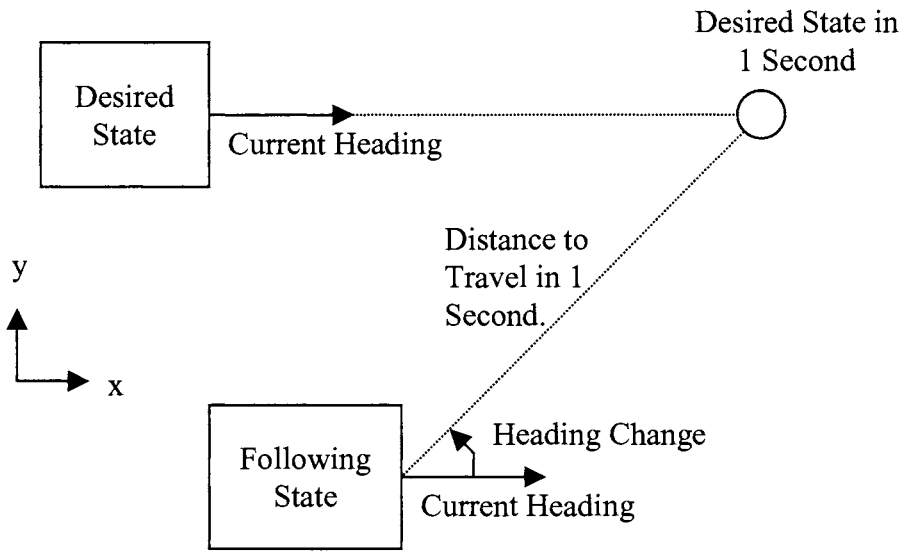

Figure 6: The Wagon Tongue Method

The wagon tongue method takes the desired state and uses its velocity, heading and acceleration to determine where it will be in the future if there are no changes to those state variables. The amount of time into the future, called the wagon tongue constant (w), is a parameter of the method assigned during software development. Figure 6 is drawn with a w of one second. W has to be at least three times as long as the time between vehicle movement commands (t_c) and frequently requires adjustment for best results. The reason that w must be at least three times longer than t_c is to account for overshooting the desired position. If w is less than or equal to t_c, then the vehicle will try to reach that spot immediately. The likely result is that it will not be heading in the right direction when it arrives and so it will not be positioned well for the next command. As w becomes larger than t_c, the sharpness of the required heading change will decrease and the follower will approach the leader's path asymptotically. This results in a smoother, more accurate path for the follower. However, if t_c is set too large, the follower will not get to the desired location quickly enough and will respond too sluggishly to positional discrepancies.

Once the desired future state is calculated, the distance between that state and the following state can be calculated. The goal is to get the following vehicle to the future desired state in a number of seconds equal to the wagon tongue constant. The distance between the follower and desired future state, divided by the wagon tongue constant, yields the constant speed the following vehicle must attain. The vector between the following state and the desired future state can be used in conjunction with the heading of the follower to determine the necessary heading change to reach the desired location. With these two pieces of information, the required inputs to the dynamics simulation can be calculated. Of course, in general, the following vehicle never arrives at the desired position because a new desired state arrives before w seconds passes and the method is applied anew. The repeated application of the method generates a series of desired positions that result in the two vehicles approaching one another asymptotically.

In the system developed for this thesis, the leader in the wagon tongue method is the SS generated by the simulated vehicle. The follower is the RS generated by the real vehicle and determined by the observer. The wagon tongue method is used by the real vehicle to determine the inputs it should execute each time step. The wagon tongue method is also used by the dynamics engine to determine the IS. The details of the IS generation process are covered in the next section.

A major limitation of the wagon tongue method is that all real vehicles have a maximum speed, but the wagon tongue method may require that a vehicle go faster than its maximum speed in order to reach the desired position. If this happens repeatedly, the real vehicle and simulated vehicle will diverge. The uncertainty box, which provides real time feedback to the operator about the distance between the real vehicle and simulated vehicle positions, is designed to help alleviate this problem. The image generator renders a blue wire frame cube around the simulated vehicle and the cube grows uniformly as the uncertainty distance increases. If the operator is trained to slow down when the cube gets too large, that

will eventually reduce the speeds demanded by the wagon tongue method below the threshold of the real vehicle's maximum speed, allowing it to catch up.

Method to Generate the IS

Using a vehicle simulation with the wagon tongue method provides a continuum of states that do not diverge indefinitely, resolving one of the issues associated with lag. Using a vehicle simulation does not, however, address the issue that every RS is itself lagged and therefore, old when received by the dynamics engine. Each RS must be brought forward to the current simulation time to be useful in calculating the IS, which is needed to provide the operator with real time updates of the real vehicle's success in following the path generated by the simulated vehicle.

There are two methods to determine the IS. One method, called dead reckoning, uses the last known heading, velocity and acceleration of the vehicle to continually predict where the vehicle is until a new vehicle state arrives. The problem with this method is that the vehicle cannot change direction or acceleration during the time that it is being dead reckoned. The second method involves storing all of the simulated vehicle states. Using this method, the lagged vehicle positions can be brought to the present time through a process similar to dead reckoning called informed reckoning. The main difference between the two methods is that instead of keeping the heading and acceleration constant, informed reckoning permits these values to be changed based on the results of the wagon tongue method between the RS and SS at each time step. In this way, the inputs to the simulated intermediate states generated while integrating the IS can be determined using the same method that the real vehicle applies when it receives a new SS. Ensuring that the same vehicle input generation method is used makes it more likely that the predicted IS will match the actual current RS.

The following example compares the use of dead reckoning with informed reckoning in a system with a one second lag. The times used in the example represent the number of

seconds that have passed since the beginning of the teleoperation task. Figure 7 illustrates this example.

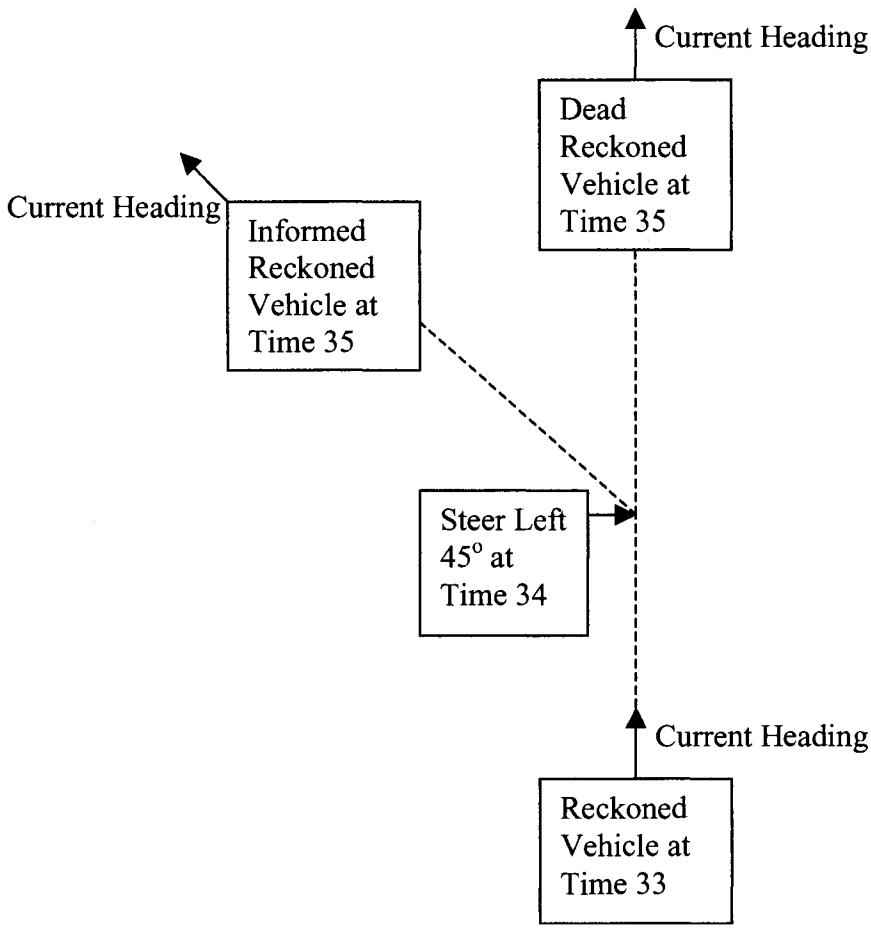

Figure 7: Dead Reckoning vs. Informed Reckoning

When the dynamics engine receives a RS at time 35, it is actually a RS from time 34. Furthermore, the user command to generate the SS that the RS used was from time 33. Therefore, while the user desires the command to occur at time 33, its results do not return until time 35. Suppose further that at time 34, the user entered a steer of 45 degrees and decelerated the vehicle so that its velocity was reduced by 0.5 ft/s. If dead reckoning is employed to generate the IS at time 35, the IS will be 0.5 feet farther forward than it should

be and it will be heading in the wrong direction by 45 degrees. Instead, if informed reckoning is used to generate the IS at time 35, the heading and speed changes at time 34 will be taken into account. Using informed reckoning, only the differences between the dynamics simulation and the real vehicle dynamics will cause a discrepancy between the IS and the RS.

Method to Improve Operator Presence

To address field of view and operator presence, a fully textured model of the setting is rendered to provide an immersive experience. In addition, a graphical model of the real vehicle is also included in the virtual environment (VE). These models are designed to give the operator the illusion of being at the remote site and to give the operator a fuller sense of the activity around the vehicle as well as the nearby terrain features. This characteristic of the VE increases the operator's sense of presence.

The VE is designed to be displayed on diverse types of devices. For example, it can be run on a computer monitor or a CAVE display system. The VE looks like a low grade automotive racing video game when displayed on a computer monitor. Despite its graphical shortcomings, the VE gives the operator a clear idea of what the terrain around the vehicle looks like, what the potential obstacles are, and where the vehicle is. The VE is particularly effective in the CAVE, as that display device allows for a 360-degree arc of visibility around the vehicle, enhancing the operator's situational awareness.

To make the VE as useful as possible, several virtual camera viewpoints are incorporated to present the virtual environment. These viewpoints are designed to increase the FOV of the operator and to provide the flexibility to allow operators to choose the particular FOV that is most useful for their current task. With the virtual camera, the operator can switch between viewpoints to see different perspectives around the vehicle. In addition, a viewpoint inside the vehicle is available to provide the illusion of driving from the vehicle's cockpit. However, a cockpit view limits the field of view unless it is being shown

on a multiple wall display device like a CAVE. Even with this type of display, operators must turn their heads to see other viewing arcs of the vehicle than the forward arc. Multiple preset virtual camera viewing positions around the vehicle are provided to address this limitation. By cycling through these views, the operator can get a forward view, rear view, side view, or chase view. The chase view is common in video games and often proves to be the most useful view. Because operators look from above and behind the vehicle, they get a complete view of the vehicle's immediate surroundings on one display surface. With a CAVE, the visible surrounding area increases substantially.

SYSTEM PERFORMANCE TESTING

A prototype version of the general model was implemented using the methods described in the previous section to validate the system model and methodologies. Figure 8 illustrates the components used in the performance testing prototype system.

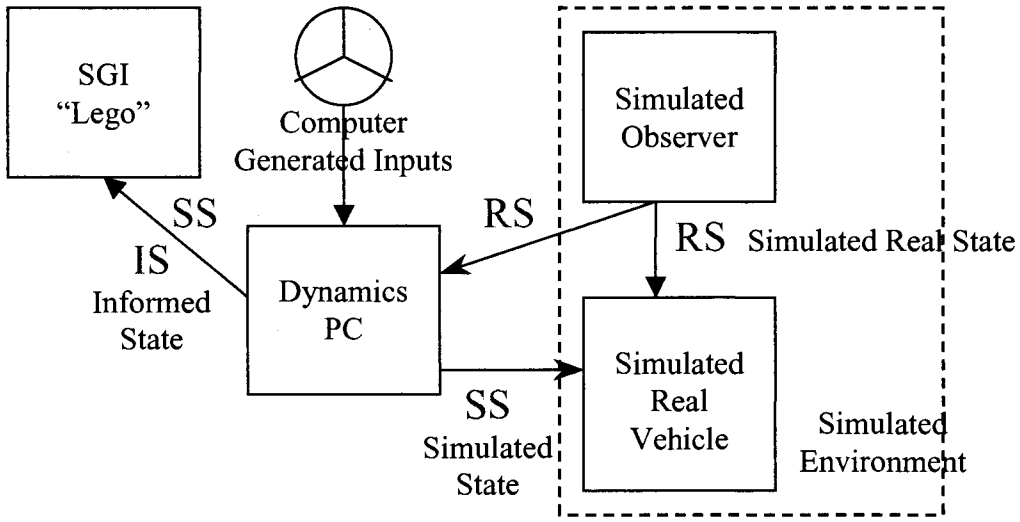

Figure 8: Simplified Prototype System

To minimize the number of hardware components involved in the performance testing, parts of the complete system were simulated. This decision allowed for the development of the more complex hardware components to be decoupled from the software testing. Specifically, both the remote control tank and the observer were simulated for testing purposes. As a result, the prototype system did not have any components at the remote environment, simplifying software development. Furthermore, this system involved no human operator because the simulated vehicle and the simulated real vehicle were given inputs based on a predetermined path. Minimizing human involvement and reducing system complexity allowed for the simulation software to be isolated and for the development of a useful test harness. This test harness allowed for the efficient improvement and performance

testing of the vehicle simulation code. For these code improvements or results of any tests of this prototype to be meaningful, the simulated components of the system had to reflect the behavior of the actual hardware they replaced. The sections following this one describe in detail how this behavior matching was accomplished.

The vehicle simulation code was designed with flexibility as a prime requirement. The code base, like VEVI, supported the addition of different vehicles and dynamics models. This design flexibility proved useful when the original remote vehicle, a remote controlled (RC) Jeep, was replaced with the RC tank. Because the tank dynamics used the same object interface as the Jeep dynamics, they were easily interchangeable. All of the vehicle simulation code was written in this modular fashion using object oriented C++ in a Windows XP environment. The image generator code was written using C++ for an IRIX environment. Most of the code developed for the prototype system was also used in the complete system.

In the prototype setup, the dynamics engine and image generator from the complete system were coupled to simulations of the observer, the real vehicle, and the input generator, to drive the dynamics in expected ways. The accuracy of their performance was then recorded. The "real vehicle" here was a simulation of the remote controlled toy tank to be used in the real system.

Test Course

To test the dynamics code in the dynamics engine, a course was designed based on the Consumer Union test course illustrated in Figure 9 [22]. Each "X" in Figure 9 represents a cone. This path is a combination of two Consumer Union lane change maneuvers, one in each direction, with one directly following the other. To successfully traverse the course, a vehicle must steer through the cones on the right and then through the cones in the middle. Finally, the vehicle must turn to the left and then straighten out to make it through the final

two cone gates. This is a modification of the course that Consumer Union uses to determine the safety of automobiles for sale in the USA. The course is designed to provide a test of a common maneuver, a lane change.

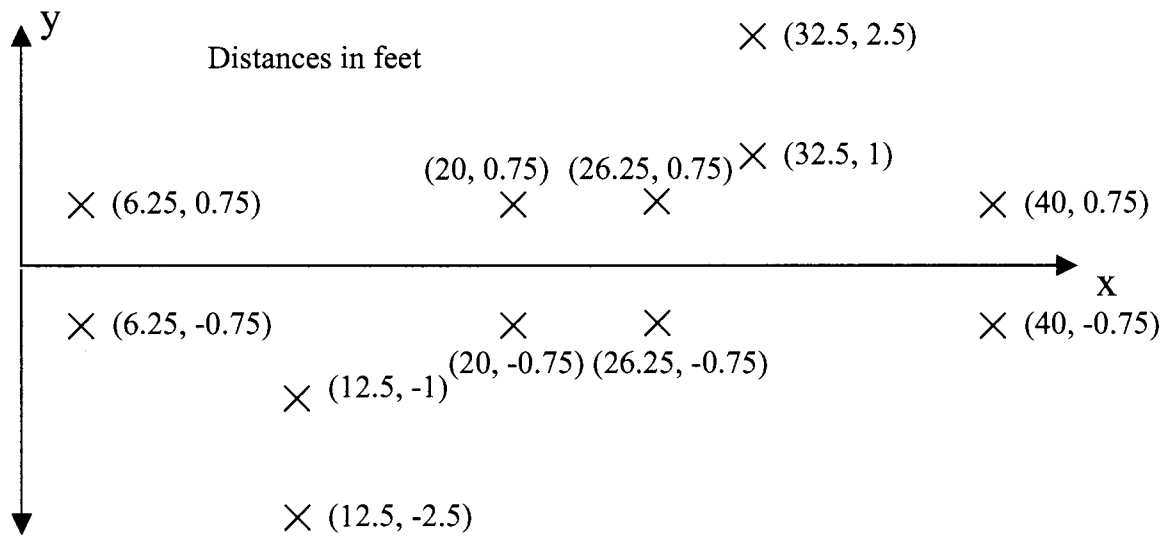

Figure 9: Consumer Union Test Course Layout

The prototype system must accomplish this double lane change to be able to accomplish more severe maneuvers. If the prototype piloted the vehicle through the course, it should be able to control the vehicle in most teleoperation tasks. To prove that it could send the tank through the course, a driver was simulated by using the tank dynamics model in the dynamics engine with calculated inputs in the form of those received from a Microsoft Sidewinder. These inputs, coupled with the simulation, were used to send the simulated vehicle through the virtual course, which in turn sent the real vehicle through the real course.

A graphics model of a test track complete with the Consumer Union course was created using Multigen Creator. The test track was a flat concrete lot with virtual cones placed at the proper locations and scale for the test course. A graphics model of the tank was created using 3DStudioMax. This model acted as the simulated vehicle in the virtual world. Figure 10 shows the virtual world rendered by the image generator on a computer monitor.

The virtual tank is making its run through one of the cone gates of the virtual Consumer Union test course.

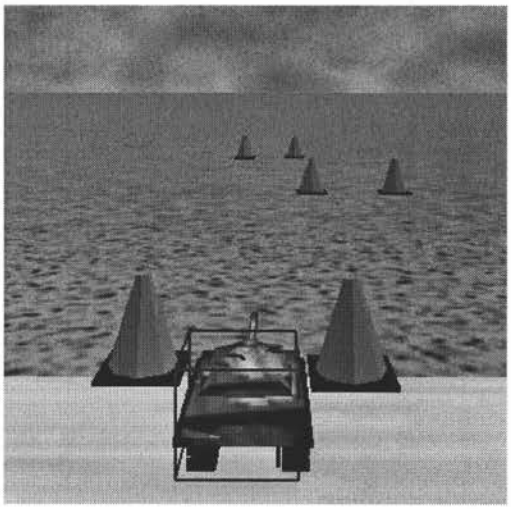

Figure 10: Screenshot of the Virtual Environment

The Image Generator

The image generator was based on two software platforms, VRJuggler and IRIS Performer. Performer provided a scene graph to organize the objects in the graphical scene. VRJuggler, an open source VR application development toolkit developed at Iowa State University, simplified the development of our VR application by providing object-oriented abstractions of a complete set of VR display and input devices. Applications written with VRJuggler can then be easily ported, without rewrite, to any platform or VR display device that VRJuggler supports. According to the VRJuggler website, "VRJuggler provides virtual reality (VR) software developers with a suite of application programming interfaces (APIs) that abstract, and hence simplify, all interface aspects of their program including the display surfaces, object tracking, selection and navigation, graphics rendering engines, and

graphical user interfaces" [23]. The prototype system used VRJuggler to display the virtual world on a computer monitor. For the complete system, the same image generation code was used to display the virtual world on VRAC's C6, a six-sided CAVE.

Simulated Components

Simulated Real Vehicle

The real remote controlled tank was not used in the prototype system, eliminating sources of error and increasing test repeatability. In its place was a software simulated real vehicle, based on the dynamics model in the dynamics engine being tested. A glance at Equations 1 and 2 shows that this implied that f_r was equal to f_s, meaning that the dynamics model was a perfect simulation of the behavior of the real vehicle. Since such a perfect match is not realistic, the track speed inputs to the dynamics model for the simulated tank were modified to introduce discrepancies. A multiplier ranging between 50% and 200% modified the magnitude of the track speed inputs. This input behavior represented a situation in which the operator asked the tank to go one speed and it actually went another.

To introduce more positional discrepancies, the track speed inputs were additionally modified randomly using a Gaussian error distribution, resulting in a ±2% change at one standard deviation. This randomization was only applied to the real vehicle inputs when the track speed inputs to the SS changed. This additional randomness represented the slightly unpredictable behavior that results from track acceleration. Applying the additional randomization in this fashion allowed the simulated real vehicle to keep a constant heading and speed when the simulated vehicle also kept a constant heading and speed. Note that due to the modification of the track speed inputs, the headings and speeds between the simulated vehicle and real vehicle still differed.

Simulated Observer

A simulated observer sent the simulated RS to the dynamics engine. Instead of a camera or GPS tracking the RC tank and providing a real time stream of vehicle states, the simulated observer was simply a buffer that held each calculated RS. When the specified lag time had passed, it alerted the RealVehicleListener object in the dynamics engine to the presence of a new RS. In this way, the RS generation and delivery were all accounted for in the simulation. As a result, the dynamics engine could not discern that the observer was simulated.

Simulated Lag

Artificial lag was generated because naturally occurring lag was not of sufficient duration. Further, simulated lag allowed for precision control of the lag duration. The lag was kept at a constant delay for the entire run. The rationale behind using constant lag was that synchronization code on the real vehicle held SS commands long enough to execute them in such a way that the time between commands was unchanged from when they were issued. A fellow graduate student, Jared Knutzon, developed the synchronization code on the real vehicle [24]. Employing the synchronization code eliminated the effect of variations in lag, making the constant lag effects the only pertinent ones. The creation of a constant artificial lag was accomplished by buffering communication packets until they reached a desired age.

The Dynamics Engine

The dynamics engine was the same for both the prototype system and the complete system. It was responsible for:

- accepting user inputs,
- converting those inputs into dynamics inputs,
- giving these dynamics inputs to a mathematical model of the vehicle to create the SS,
- receiving updates of the last known RS,
- calculating the IS, and
- sending the latest IS and SS to the image generator for display.

A Hewlett Packard PC with a 950 MHz Athlon processor and 256 M RAM was used to perform these tasks. The dynamics engine and the image generator were on the same Local Area Network, a 100Mb Ethernet link.

Dynamics Math Model

As shown by Equations 4-8, the dynamics model employed for the tank was a simple yaw plane model that used the sum of moments of the track speeds to determine the vehicle state. The kinematics diagram used to generate the equations is shown in Figure 11.

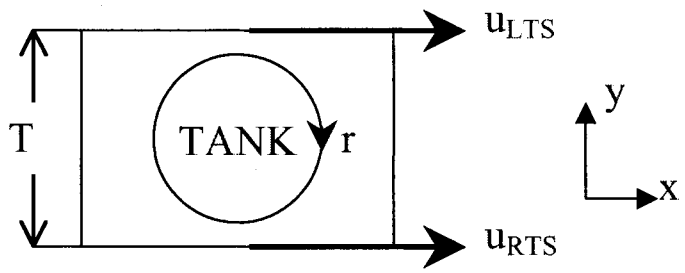

Figure 11: Tank Kinematics Diagram

$$u = \frac{u_{LTS} + u_{RTS}}{2} \qquad \textit{Equation } 4$$

$$r = \frac{u_{LTS} - u_{RTS}}{T} \qquad \textit{Equation } 5$$

$$\frac{dx}{dt} = u \cdot \cos\psi \qquad \textit{Equation } 6$$

$$\frac{dy}{dt} = u \cdot \sin\psi \qquad \textit{Equation } 7$$

$$\frac{d\psi}{dt} = r \qquad \textit{Equation } 8$$

Equation 4 shows that the vehicle speed is simply the average of the track speeds, with u_{LTS} and u_{RTS} representing the left and right track speeds respectively. The velocity is in the direction of the heading, ψ. The yaw rate, r, calculated by Equation 5, is related to the width, or track, of the vehicle, T. Equations 6 and 7 were integrated to yield a position, while Equation 8 was integrated to get the heading of the vehicle. The numerical integration method was Runge Kutta with a time step of 0.05 seconds. For flexibility, the integrator and dynamics equations were written such that the integrator could use different sets of dynamics equations. Therefore, the dynamics for an automobile could be used by the system without requiring sweeping changes to the code. This would allow the system to incorporate a Chevrolet Corvette or a Bradley fighting vehicle in place of an RC tank.

Dynamics Input Generation

The left and right track speeds were the inputs to the dynamics model described above and used in Equations 4 and 5. However, the interaction device used to generate operator inputs in the complete system was a Microsoft Sidewinder steering wheel and pedals set. Because the prototype system was simulating user inputs with a simulated driver, it generated them in a similar fashion to how they were issued in the complete system with the Sidewinder. Taking in inputs in this manner allowed for the same tested code base to be

used in the complete system. Unfortunately, the Sidewinder gave inputs as changes in desired speed and heading and not as track speeds.

To convert steer and speed values into track speeds, the inputs were represented as percentages off the maximum allowed by the input device. For example, if the operator steered all the way to the left, a steer input of -100 was generated, while if the steer were all the way to the right, it generated a steer input of 100. Likewise, pressing the throttle pedal halfway resulted in a speed input of 50. These inputs needed to be converted into track speeds to be made into dynamics inputs. For this to be possible, the maximum steer inputs and speed inputs had to be mapped into track speed values. These mapping factors were chosen so that an input of 100 steer and 100 speed did not require one track to go faster than the real vehicle allowed. Note that this meant that the maximum mapped speed was less than the maximum speed attainable by the real vehicle.

A programming object called the TankJoystick performed this function. It encapsulated the code necessary to get data from the steering wheel and pedals set. The TankJoystick then used Equations 9-10 to convert these inputs into track speeds. It was also capable of performing the reverse conversion, which was useful in generating the simulated RS and IS. Equations 11-12 show how the reverse conversion was accomplished.

$$u_{LTS} = \frac{u_{T\max}\left(speed - \frac{steer}{2}\right)}{100} \qquad \textit{Equation } 9$$

$$u_{RTS} = \frac{u_{T\max}\left(speed + \frac{steer}{2}\right)}{100} \qquad \textit{Equation } 10$$

$$steer = \frac{100(u_{LTS} - u_{RTS})}{\delta_{\max}} \qquad \textit{Equation } 11$$

$$speed = \frac{100\left(\frac{u_{LTS} + u_{RTS}}{2}\right)}{\delta_{\max}} \qquad \textit{Equation } 12$$

The u_{Tmax} in Equations 9 and 10 is the maximum track speed that either track could achieve. The δ_{max} in Equations 11 and 12 represents the difference between the maximum and minimum track speeds allowed and is the mapping factor described above. Because the TankJoystick encapsulated all of these conversions, the dynamics engine only needed to ask for the operator's desired track speeds and all of the conversions were transparent to it.

Receipt of Real Vehicle Updates

Another task of the dynamics engine, receiving each RS, was accomplished by an object called the DynamicsCorrector. The DynamicsCorrector was responsible for getting real tank positions, and then using them to predict the position of the real tank at the current time using informed reckoning. Once an estimate of the real tank's current position was available, the DynamicsCorrector computed the IS and the uncertainty error between the RS and the SS. To perform the informed reckoning, the DynamicsCorrector spawned a thread called the integrating thread. The integrating thread's only responsibility was to perform informed reckoning so if there was no new RS available, it sat idle. In order for the DynamicsCorrector to feed the integrating thread new RSs, it also created the RealVehicleListener, which ran in its own thread, the listening thread. The RealVehicleListener listened for a new RS from the observer. Whenever it received a new vehicle state, it stored it and notified the integrating thread that a new RS was available for reckoning.

Whenever the integration thread was notified about a new RS, it immediately compared the current simulation time with the time stamped in the RS. It then integrated from the time stamped in the RS to the current simulation time by generating commands at each time step. The commands were generated using the SS history and the wagon tongue method. Specifically, at each time step of the informed reckoning, the integration thread got the SS for that time from the history and compared that SS with the current position of the

vehicle it was integrating from the RS. The integrating thread then performed the wagon tongue method on these two states to determine what track speed inputs to feed to the dynamics model for the next time step. In this application of the wagon tongue method, the intermediate state was the follower and the SS was the leader. During this integration, the integrating thread timed itself on how long it had taken. Then, as the end of the integration neared, it added on the time elapsed during the integration to bring the IS as close to the actual current simulation time as possible. During the integrating thread's integration of a particular RS, any new RSs were ignored until it was done integrating the one it was working on.

Once the integrating thread was done creating the IS from the RS, it notified the primordial thread of the DynamicsCorrector that a new IS was available for comparison with the latest SS. The DynamicsCorrector then calculated the distance between the IS and the SS, yielding the error or uncertainty between the two states. Additionally, if the file generation option was selected, the IS, SS, and the uncertainty distance were recorded in a file for later playback or analysis. To get the newly calculated IS and the uncertainty distance to the image generator, the dynamics engine employed another object, the GraphicsClient. Each time step, it sent the SS, the latest IS, and the uncertainty distance to the image generator via a TCP/IP connection. The Image Generator used the uncertainty distance to draw the blue cube around the simulated vehicle to provide a visual cue to the operator about the uncertainty between the real and simulated vehicles.

System Performance Testing Results

The prototype system described previously was used to test the dynamics engine and the overall system design. The test cases involved three system variables: the lag delay, the mean track speed modification in the simulated real vehicle, and the presence or absence of the wagon tongue method. The one-way lag varied between three values: one, five, and ten

seconds. Five different track speed multipliers were chosen: 50%, 80%, 100%, 120%, and 200%. A track speed multiplier of 80% meant that when the real vehicle was given a track speed input of 1 ft/s, it actually moved the track at 0.8 ft/s. Factoring in the inclusion or exclusion of the wagon tongue method, thirty test runs were completed. The mean and standard deviation of the uncertainty distance during the entire run through the Consumer Union course was calculated for each run.

Figure 12 shows the mean uncertainty distance for all the runs that did not use the wagon tongue method, while Figure 13 shows it for all those that did. Likewise, Figure 14 shows the standard deviation of the mean uncertainty for all the runs that did not use the wagon tongue method, and Figure 15 shows it for all those that did.

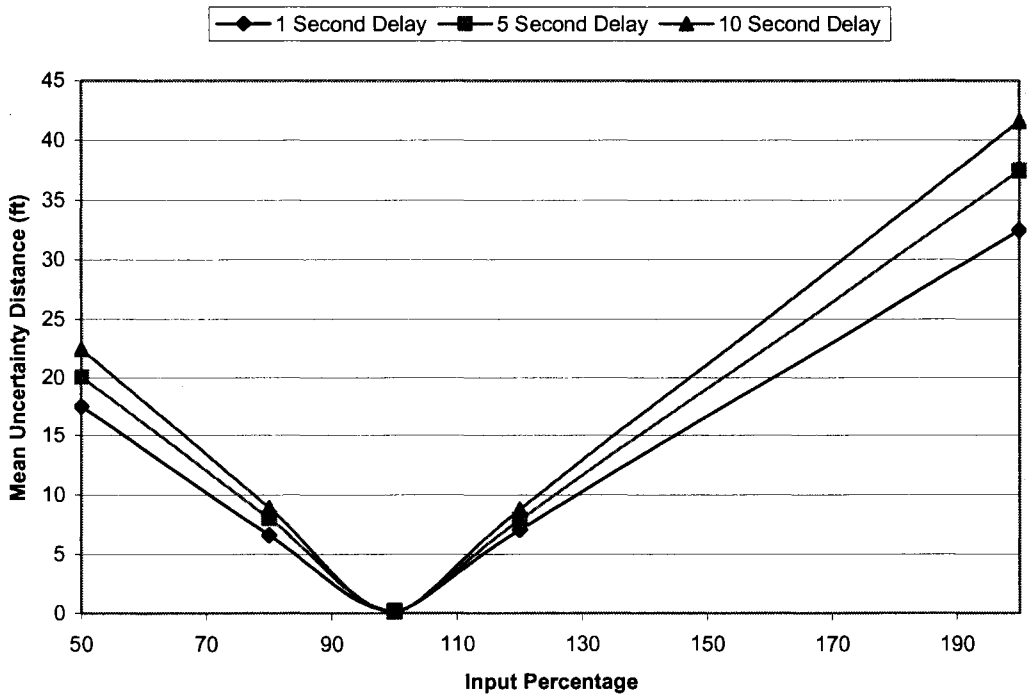

Figure 12: Mean Uncertainty Distance With No Correction Method

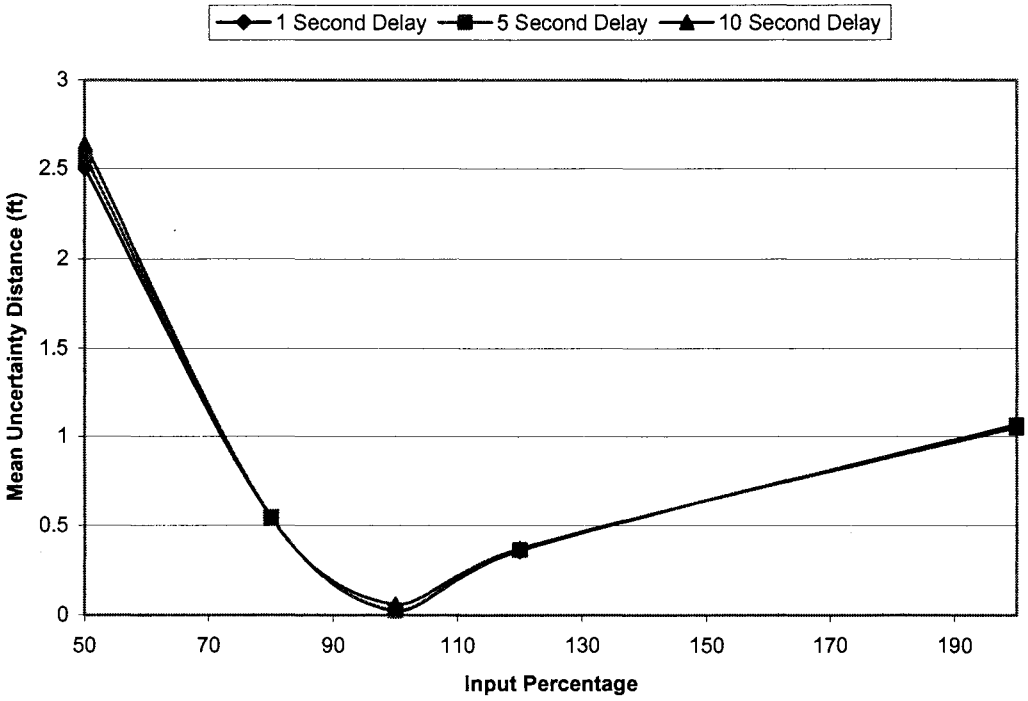

Figure 13: Mean Uncertainty Distance With the Wagon Tongue Correction Method

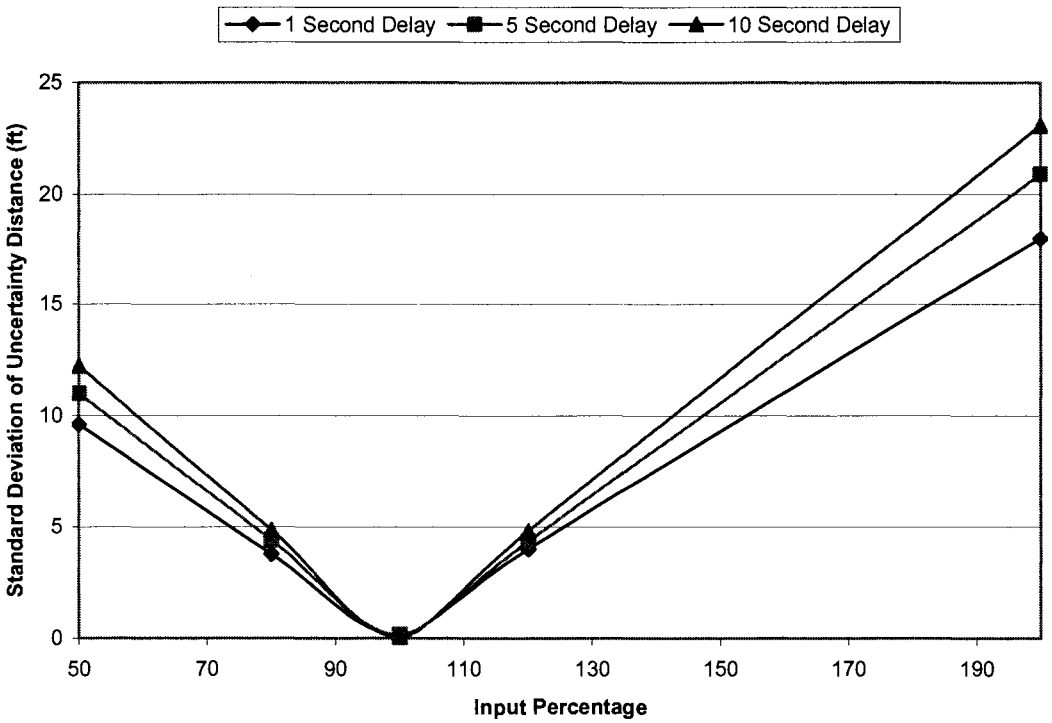

Figure 14: Standard Deviation of Uncertainty Distance With No Correction Method

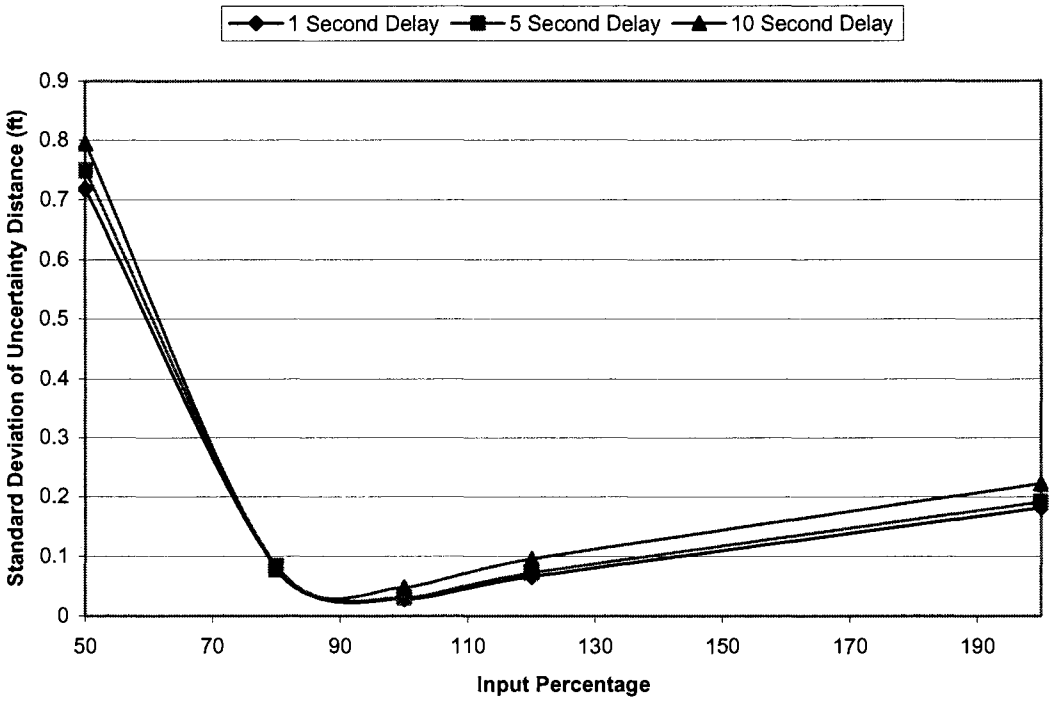

Figure 15: Standard Deviation of Uncertainty Distance With the Wagon Tongue Correction Method

System Performance Testing Conclusions

The results shown in Figures 12-15 illustrate that, when a command correction method was employed, the system model was sound, suggesting that the dynamics engine described above would be suitable for use in the complete system. In all of the tests, the real vehicle was able to follow the path laid down by the simulated vehicle within an envelope. This envelope changed size based on the parameters of the test, the most significant of these being the inclusion or exclusion of the wagon tongue method.

The wagon tongue correction method improved the system performance dramatically. The inclusion of the wagon tongue corrector reduced the maximum uncertainty distance from over 42 feet to just 2.7 feet. In fact, if the inputs were modified to be above 75% and below 190%, the uncertainty distance was less than a foot, even with a lag of ten seconds. This result is important because this range falls into the domain of solid dynamics models. In

many applications, creating a model that can approximate the behavior within this range is possible.

The standard deviations of this mean took on large values that exceeded 20 feet in the runs that did not employ the wagon tongue method. An uncertainty distance was calculated between the SS and the RS each time step during the entire run and these uncertainties were then averaged to create the mean uncertainty. In the runs that did not use the wagon tongue method, the uncertainty at the beginning of the run was small and it grew continuously. As a result, a large number of the uncertainty distances were well above or below the mean, indicating that the mean was not a very good indication of the uncertainty at any one time. This happened because there was nothing to keep the SS and RS from diverging farther and farther apart over time.

The deviation in the runs with the wagon tongue correction method was less than 0.8 feet for all runs, indicating that the mean uncertainty was close to the actual uncertainty at any given time. This was because the corrective measures of the wagon tongue method worked to minimize the divergence between the SS and RS.

An interesting observation can be made by looking at Figures 12 and 13. Without the wagon tongue correction method, the maximum uncertainty occurred with 200% inputs and the uncertainty for the high input multipliers was larger than those for the smaller ones. With the wagon tongue, the opposite occurs. This behavior can be explained for both cases. In the case of the runs without the wagon tongue, the uncertainty increased to a larger value as the input multiplier increased because the tank moved farther each time step. Since there were no corrective measures, the tank sped away from where it should be faster than it did when the track speeds were slowed by the input multiplier. In the case of the runs with the wagon tongue method, the highest uncertainty occurred at the low input multipliers because as the real vehicle lagged behind the simulated vehicle, the wagon tongue method demanded that it speed up. Before long, the speed demands of the wagon tongue method became more than

the real vehicle could provide. Therefore, as it fell behind and the uncertainty grew, it had to catch up slowly because there was no human to watch the uncertainty box grow and slow the vehicle down in response. Since the simulated vehicle slowed down on the return from a cone gate back toward the centerline, the real vehicle used that time to catch up with the simulated vehicle.

Figure 16 shows the vehicle path for both the corrected and non-corrected vehicles, along with the desired path of the simulated vehicle using 50% inputs and a five second delay. Notice that the corrected vehicle's path is clearly lagging behind the desired path due to the wagon tongue method's demands for speeds that cannot be supplied by the real vehicle.

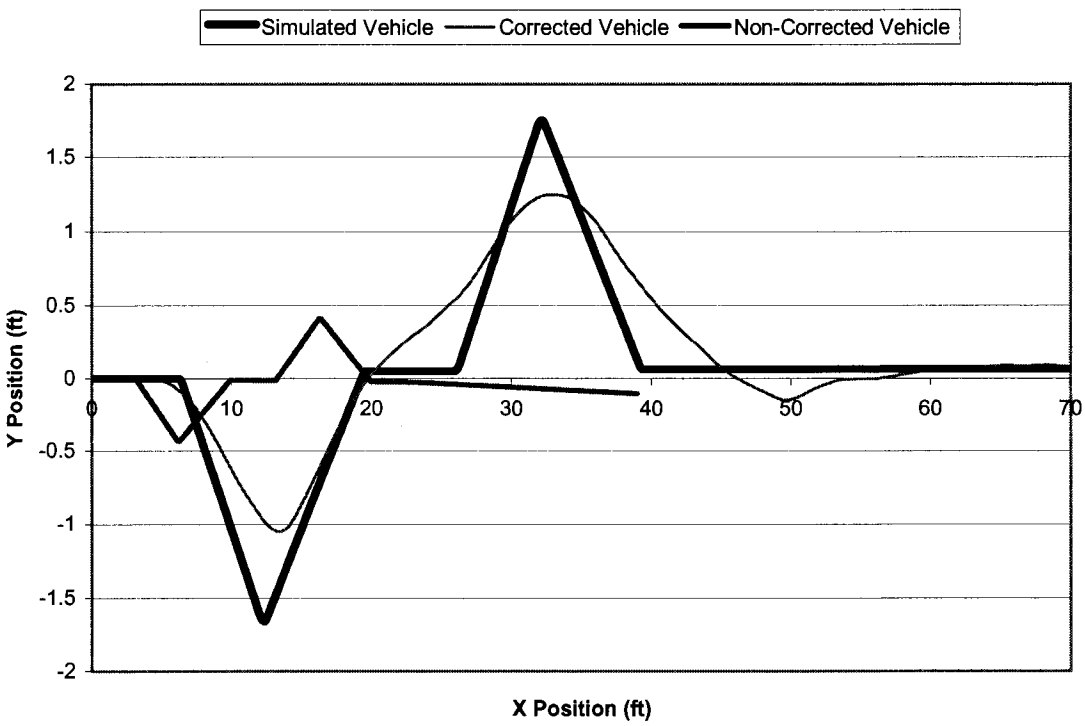

Figure 16: Vehicle Path With 50% Input and a 5 Second Delay

In every case where the inputs were perturbed, the wagon tongue method improved the match between the simulated vehicle and the real vehicle. This result can be clearly seen

in Figure 17 which depicts the vehicle path at 200% inputs with a ten second lag delay. The non-corrected vehicle angled upward instead of going straight at the end because of the ±2% random variation of the inputs whenever the simulated vehicle changed direction. While the corrected vehicle could accommodate this randomness, the non-corrected one had no way of getting headed in the right direction. Promising results like those shown in Figure 17 promote confidence in the general system model and the methods employed in implementing it.

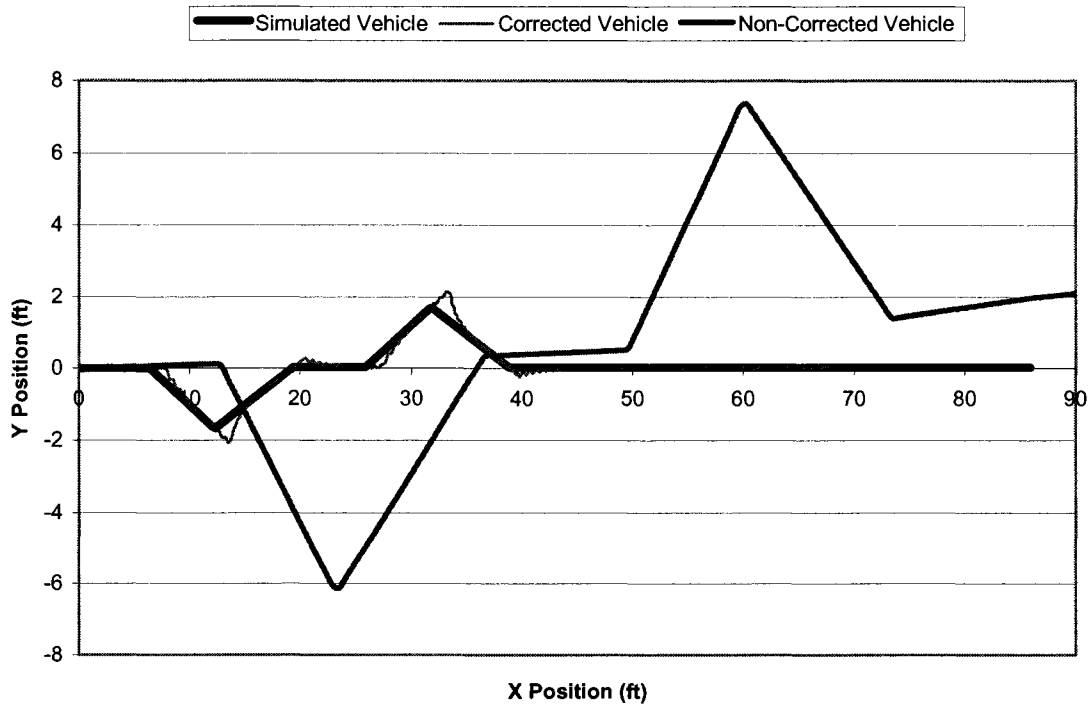

Figure 17: Vehicle Path With 200% Input and a 10 Second Delay

COMPLETE SYSTEM TEST

With the system model and methods tested, the dynamics engine and image generator were then connected to hardware versions of the observer and real vehicle to perform a complete system test. Figure 18 shows the components of the system in terms of the system model. Each piece is described individually. A test of the system and its results are also described.

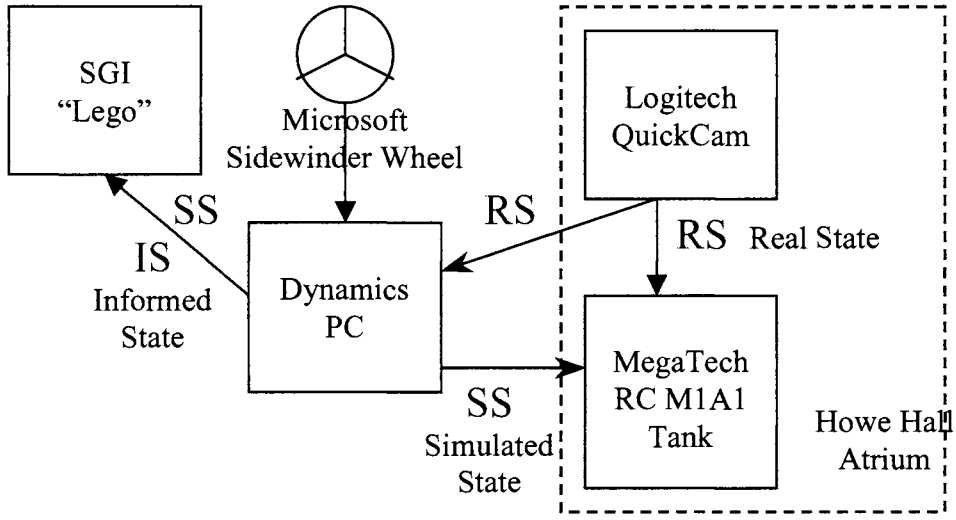

Figure 18: Complete System

The Real Vehicle

The vehicle chosen for this application was a remote controlled MegaTech 1:20 scale M1A1 tank [25]. Figure 19 shows the MegaTech M1A1 tank with the red and blue tracking squares attached.

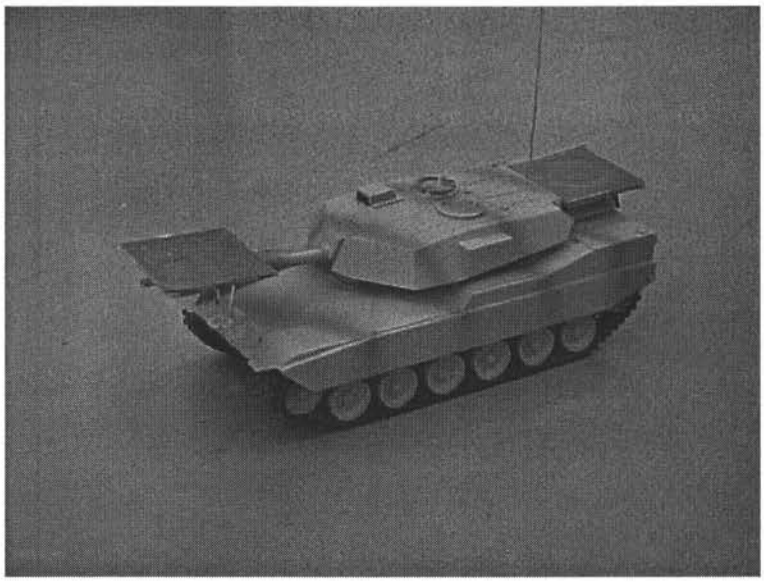

Figure 19: The Real Vehicle

The tank's stock controller has two joystick controls, one for speed and one for steering. The tank has proportional inputs meaning that the farther the speed control stick is pressed forward, the faster its tracks go. Likewise, the vehicle has proportional steering. This characteristic of the tank is highly desirable because digital remote controlled vehicles that can provide only "all or nothing" responses to inputs are more difficult to simulate using vehicle dynamics equations. Digital control vehicles have very different dynamic responses to inputs when compared to their real world analogs. A proportional input vehicle matches its real world analog more closely and thus can be simulated using equations assuming a continuum of input values.

The tank's remote control was modified to allow for direct communication between it and a computer parallel port. WinIO [26] was used to manipulate individual pins of the parallel port. These pins were accessed by C++ code written by Jared Knutzon to send the desired tank inputs over the parallel port to the modified controller. The tank inputs were represented by the percentage off maximum supported for both the speed and steering. For example, an input of 50 called for the tank to be driven forward at half speed. This input

representation was similar to how inputs were modified as they came from the steering wheel. Each "percentage off" the maximum input value was then converted into an 8-bit number. These 8-bit numbers were sent to digital potentiometers to change the voltage applied to the controller. This change in voltage caused the speed or steering to respond as though the computer had "pulled" or "pushed" a thumb stick on the remote control. The specifics of the electronic circuit and other modifications made to the remote control are beyond the scope of this thesis and can be found in [24]. A Dell Inspiron 8200 laptop interfaced with the controller. It had a 2.4 GHz Intel Pentium 4 processor with 256 M RAM and ran Windows XP.

The User Input Device

A Microsoft Sidewinder steering wheel and pedals set designed for use with computer games was the user input device. While a steering wheel did not match up well with the input devices found on real tanks, it did match up well with the real vehicle controller interface. Figure 20 shows the Sidewinder set used for this test.

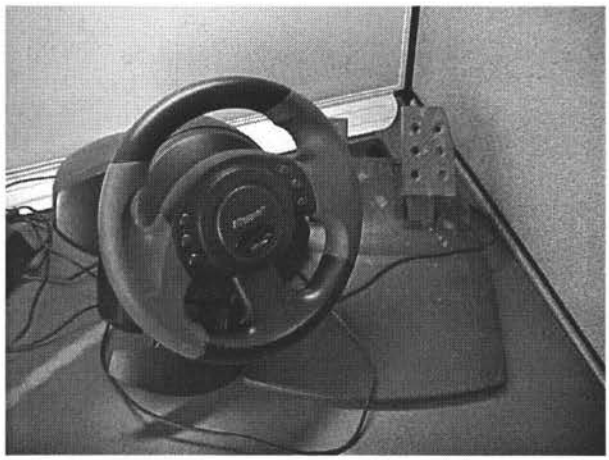

Figure 20: The User Input Device

The Dynamics Engine

The dynamics engine was the same one used for the simplified prototype system described in the System Performance Test section. For the full system test, the dynamics engine was run on a Dell Dimension 4550 PC with a 3.1 GHz Intel Pentium 4 processor and 256 MB of RAM running Windows XP.

The Environment

The teleoperation environment was the atrium of Howe Hall at Iowa State University. This area is a large three story high room with a concrete floor. A 30-foot by 22-foot area of the floor was utilized as the operational area for the test. It is an ideal location for the test because it is devoid of potential collision hazards, has a convenient place to mount the camera to offer a view of the entire test area, and has a wireless local area network. Figure 21 shows the Howe Hall Atrium. Figure 22 shows the virtual version of the Howe Hall Atrium.

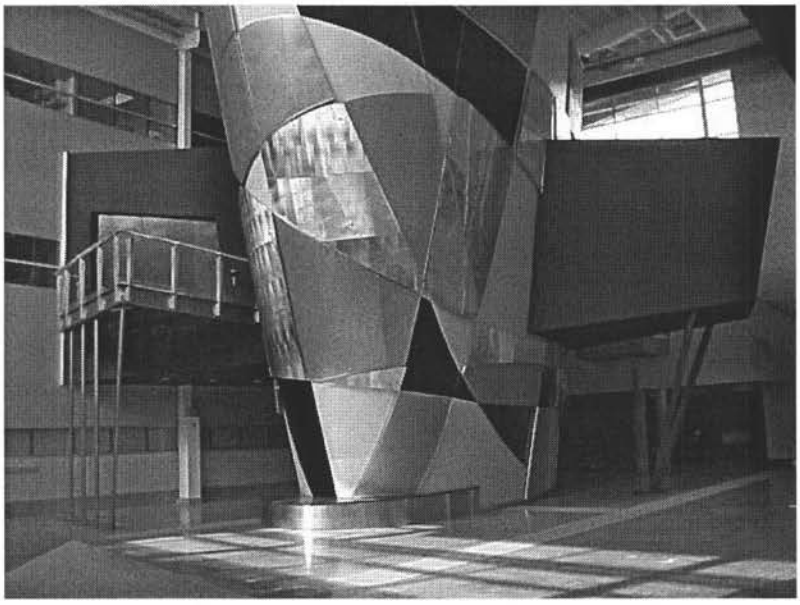

Figure 21: The Test Environment

Figure 22: The Virtual Environment

The Image Generator and Display Device

The display device used for the test was the Virtual Reality Applications Center's C6. The C6 is a six-walled CAVE display device with each wall consisting of a 10'x 10' stereoscopic screen [27]. A wireless Ascension Technologies tracking system allows the environment to be tailored to the user's location without tethering the user to the computer. To save space, the images from the six projectors are bounced off mirrors, which accounts for the shape of the housing for the C6, the large silver and black structure dominating Figure 21. Figure 23 shows a schematic of how the C6 works. The computer that generates the images displayed by the C6 is an SGI Onyx2 InfiniteReality2 Monster known as "Lego." Lego has "six InfiniteReality graphic displays, 24 R12000 processors, 12 gigabytes of memory, and access to large disk I/O and gigabit Ethernet networking" [27].

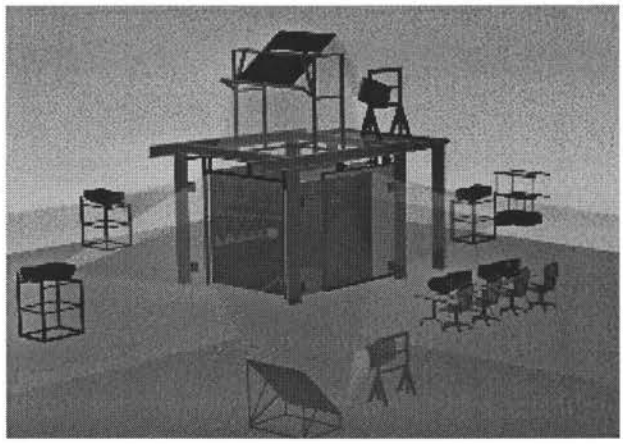

Figure 23: The C6 Display Device

The Observer

The observer for the complete system was a Logitech QuickCam Pro [28] placed approximately 35 feet above the atrium floor. For the tracking system to provide the location of the tank, it had to pick the tank out of individual video frames captured by the camera. To accomplish this, two squares made of cardboard covered with electrical tape were mounted on the tank. One square, mounted on the front of the tank, was covered with blue tape. The other square was mounted on the back of the tank and was covered with red tape. The tracking system searched each camera image for the locations of the blue and red squares and then used the pixel location of the locus of each color to determine the ranges of the tank's position in the atrium. The conversion from pixel location to world location was simplified because the tank was constrained to move in a plane. The heading of the vehicle was determined by calculating the angle between the x-axis and the vector formed from the red to blue square locations. Once the heading was determined, the position was calculated by starting from the red square location and adding half the distance between the blue and red squares along the direction of the heading. The linear and angular velocities were calculated by taking the difference between the last position or heading and the current one and dividing

that difference by the time since the last state was observed. This state information was then sent to the vehicle and the dynamics engine. The details of how the image was processed for the tracking system are beyond the scope of this thesis and can be found in [24].

The same Dell Inspiron 8200 laptop that ran the WinIO interface code for the tank remote control ran the image processing code for the observer. The application running the optical tracking communicated via a wireless TCP/IP connection with the remote controller application to get the time stamp of the RS just obtained by the observer. This behavior was important for the generation of the IS. When a SS came from the dynamics engine, it came with a time associated with it. This time was one dynamics time step beyond when the operator entered inputs. Once a new SS was received, the wagon tongue method was used to determine the inputs sent to the tank's controller. Shortly thereafter, the camera calculated the RS resulting from that set of inputs. It then took the time stamp for the position it just calculated and added it to the RS it was about to send to the dynamics engine. This time was the same as the time stamped in the SS. When the dynamics engine got the time stamped RS, it compared that state with the SS from that time in its calculation of the IS.

System Test Design

A test was devised to determine the effectiveness of the VR system in aiding the operator to teleoperate the vehicle. This test had the same objectives as the test described in the System Performance Testing section. Furthermore, in the complete system test, the user was required to pilot the RC tank through a modified version of the Consumer Union course used in the prototype system. This course is shown in Figure 9 in its original setup. The commonality between the prototype and complete system tests showed that the complete system test was an extension of the prototype system test. It simply replaced all of the simulated components with the real ones.

Test Course

The test course was modified from its original configuration shown in Figure 9 by removing the last cone gate, starting the tank inside the first gate instead of 6.25 feet before it, and increasing the width of the gates by one foot. The first two modifications were needed to fit the course inside the trackable area of the atrium. The last modification provided slightly more lenient targets to accommodate the imprecise control of the toy tank. The course was setup on the Howe Hall atrium floor with cones made of soda cans covered with white paper. The success of the operator was determined by how long it took to pilot the tank through the gates as well as how many gates were successfully navigated. Runs employing artificial lag delays of one, five and ten seconds were tested. Lags were simulated, since the user and dynamics engine were connected to the same high bandwidth network and naturally occurring lags approached only a tenth of a second at the high end. The relative effectiveness of the VR aided teleoperation system was also determined by performing test runs using direct unlagged control and teleoperation via a vehicle-mounted camera with the same test course.

Direct Control

Direct control was tested to gain a baseline of the best performance possible. Direct control was accomplished by placing the operator and the steering wheel in the atrium so that the operator could see the tank directly. Because direct control is unlagged control, this test could not be done at varying levels of lag. This type of control represents the easiest method for the operator and it is what all teleoperation systems attempt to approximate and mimic.

Camera Aided Teleoperation

The vehicle-mounted camera method represents the most common method for teleoperation. While the camera aided method employed in this thesis was simplistic

compared to other, more advanced camera aided teleoperation systems, it suffered from the same difficulties and had the same basic characteristics of more sophisticated methods. This means that the simple method employed here can still provide useful results for comparisons. These results were generated from runs at the three lag levels and represent the minimum values the VR system must outperform to be an improvement. A ZTV Electron wireless camera was mounted on the tank for the video feed test runs. Figure 24 shows this very small camera. Its dimensions are 0.79" x 0.79" x 0.55" and it provides a color image with a 900 MHz signal [29]. Figure 25 shows this camera mounted on the tank. A simple application was written to buffer the images coming from the camera using an open source library called OpenCV [30]. This program was needed to provide the effect of the artificial lag. It took as its input the amount of time to hold the image before displaying it. This image delay, along with the buffering of operator inputs, accurately simulated the effect of lag on the system.

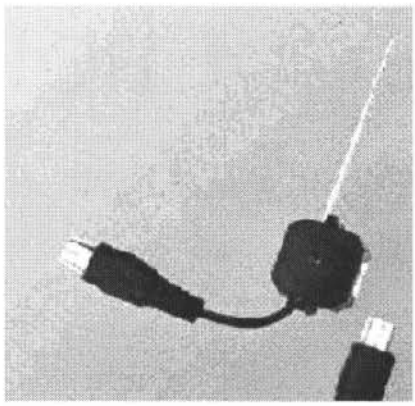

Figure 24: Vehicle-Mounted Camera

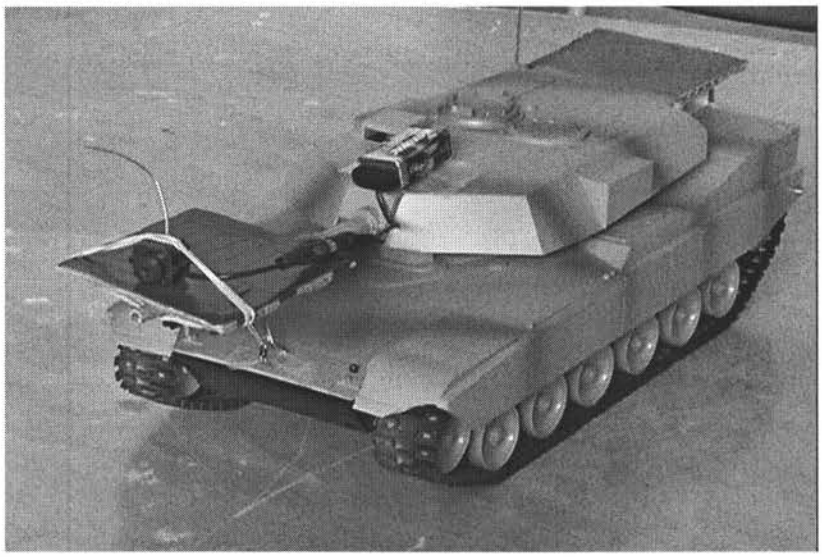

Figure 25: Camera Mounted on the Tank

Matrix of Test Runs

With all of the system components in place, seven test cases were performed to determine the effectiveness of the VR system. These cases were: direct control; camera control with one, five, and ten second delays; and VR control with one, five, and ten second delays. Three runs of each type were planned to buffer against outlier results. In each of the test cases, the user inputs were collected and sent to the tank every 0.75 seconds.

Results

Direct Control

The best-case scenario for vehicle operation is direct control. After approximately fifteen minutes of practicing piloting the tank through the cone gates with direct control, three test runs were performed. The results of these runs are shown in Table 2. With direct control, the tank made it through all of the gates for each run. The average time for a run through the course was 26.0 seconds. As the driver for these test runs, the author experienced the smoothness of direct control as it provided immediate feedback and

response. The direct control results acted as a baseline for the other two methods, VR aided teleoperation and video camera aided teleoperation.

Table 2: Results for Direct Control

	Elapsed Time (sec)	**Gates Navigated**
	25.6	5
	26.1	5
	26.3	5
Averages:	**26.0**	**5.00**

Camera Aided Teleoperation

The next test runs evaluated the most common method for vehicle teleoperation, camera aided teleoperation. In these tests, a ZTV Electron wireless camera was mounted to the cannon on the front of the tank, providing a 90-degree arc of visibility. After fifteen minutes of practice, the author executed the test runs shown in Table 3. Three sets of runs were performed with increasing lag times. In the first set, three test runs through the cones were completed with a one second delay. In the second set, two runs through the cones were completed with a five second delay. In the third set, two runs were completed with a ten second communication delay. The run time was measured as the elapsed time between the first movement of the tank and the moment when the tank passed completely through the last gate.

Qualitatively, piloting the tank through the gates using the delayed camera feed proved far more difficult than using direct control. Unlike direct control, the tank could only be controlled in a "stop and start" fashion, and control became markedly more difficult as the lag increased. Even in the best case of a one second delay, it took an average of 101.1 seconds to pilot the tank through the test course, almost 4 times longer than with direct control. In addition, the second cone gate was missed in the second run.

Table 3: Results for Video Camera Aided Teleoperation

Lag Delay	Elapsed Time (sec)	Gates Navigated
1 Second	145.3	5
	68.2	4
	89.7	5
Averages:	**101.1**	**4.67**
5 Seconds	307.7	4
	407.7	5
Averages:	**357.7**	**4.50**
10 Seconds	633.1	4
	533.8	5
Averages:	**583.5**	**4.50**

A quick glance at the data in Table 3 shows that the deviation of the times from the average can be large. The reason for this variation was that control via this method is difficult, rendering the repetition of the same path through the cones nearly impossible. In the case of the one-second runs, any action made by the operator took one second to reach the vehicle and then it took another second for the result of that action to show up on the camera feed. As a result, the only effective method of control was to take one action at a time and then wait for the results to return. For example, the operator would accelerate the tank and then wait double the lag time (two, ten or twenty seconds depending on the case) before observing the results of their action and then taking another one. This made for a frustrating and slow process, especially in the runs with a ten second lag. This is illustrated by the increase in the average elapsed times from 101.1 to 357.7 to 583.5 seconds as the lag delay increased from one to five to ten seconds.

In each test case, at least one cone gate was missed. This resulted from the loss of situational awareness that comes with a restricted field of view. As the author sent the tank through a cone gate, it was necessary to estimate whether the tank had gone far enough through the gate to avoid hitting a cone if it subsequently turned left or right. In three test runs, the tank was believed to be sent far enough forward but after a few seconds, the driver

learned that a cone had been knocked down. The limited information provided to the driver by the camera feed made such predictions difficult. The missed gates illustrate the dangers of losing situational awareness around the vehicle.

VR Aided Teleoperation

The final test runs evaluated the VR aided teleoperation method developed in this thesis. Once again, the author served as the driver and the time elapsed was measured as the time from when the tank first moved to when it passed through the last gate. The author practiced with the system for approximately fifteen minutes before making any of the test runs. In the test runs, the author was immersed in a virtual atrium in the C6 and drove a simulated tank through the test course. Qualitatively, the experience felt similar to the direct control runs. The results of these runs are shown in Table 4.

Table 4: Results for VR Aided Teleoperation

Lag Delay	Elapsed Time (sec)	Gates Navigated
1 Second	30.2	5
	31.4	5
	36.0	4
Averages:	**32.5**	**4.67**
5 Seconds	41.9	5
	31.1	5
	31.0	5
Averages:	**34.7**	**5.00**
10 Seconds	31.2	5
	30.2	5
	31.7	4
Averages:	**31.0**	**4.67**

The results show that the average elapsed time was between 31 and 35 seconds for all of the cases. This range of average times was 5 to 9 seconds longer than that for direct

control, which was only an increase of 19% to 35%. VR aided teleoperation took noticeably less time when compared with the camera aided teleoperation. In the case of one second lag, VR aided teleoperation was 68.6 seconds faster. To put that in perspective, the VR aided teleoperation runs at 1 second lag took 25% longer than direct control while camera aided teleoperation runs took 289% longer than direct control. VR aided teleoperation was faster than camera aided teleoperation in the five and ten second lag cases by 323.0 and 552.5 seconds respectively.

The fact that the average elapsed time of all the VR aided teleoperation runs fell in the same 4 second range implied that the VR aided method was independent of the magnitude of the lag delay. This result was expected due to the synchronization code, the simulation, and wagon tongue corrective algorithm. These system components made the behavior of the tank independent of the lag. The synchronization code insured that the commands were executed within the correct time interval. The simulation, in concert with the wagon tongue method, generated the correct commands to keep the tank on the desired path. None of the system components was affected by the magnitude of the lag. What was affected by lag magnitude was the uncertainty box feedback in the virtual world. In the case of the ten second lag runs, the first positional data from the tank reached the image generator twenty seconds after the time that the user entered the commands via the steering wheel. Since these tests only lasted about 31 seconds, the error box was only available for the last 11 seconds of the run. In all of the runs, the box never expanded more than a few percent of the tank size because the wagon tongue method kept the tank within one foot or less during the entire run.

Conclusions

This thesis evaluated the effectiveness of a VR aided teleoperation system at providing an improved operator interface to control a teleoperated vehicle. The results of the

tests of the VR aided system, camera aided system, and direct control system showed that the VR aided teleoperation system shared many of the desirable qualities of direct control. This conclusion is meaningful because direct control is the goal of all teleoperation systems. Both direct control and VR aided teleoperation were independent of lag and the average time elapsed for the runs were close. The VR aided teleoperation system did miss a few gates, but this shortcoming could be addressed in future versions.

The results also clearly showed that VR aided teleoperation was superior to camera aided teleoperation. Despite the fact that the number of gates missed was about the same between the two methods, the time elapsed during the runs was far less for the VR aided teleoperation system. In the case of a ten second delay, the difference between the average time elapsed between the two methods was almost nine minutes – the camera aided method took more than 18 times longer to navigate the course than the VR Aided method. This large time difference was due to the difficulty of piloting the vehicle using camera aided teleoperation. With VR aided teleoperation, driving the vehicle through the course was similar to driving with unlagged direct control; the only difference being that one used a virtual world while the other used the real world. With camera aided teleoperation, the operator relied on a lagged camera feed to move the vehicle. This characteristic of the system made it slow, frustrating and fatiguing. Conversely, the VR aided system used an unlagged virtual version of the vehicle as the interface to the user, so the lag and its effects were invisible to the user.

With small changes to the system, the accuracy of the VR method could be improved. The cone gates were missed in the VR aided teleoperation primarily due to differences between f_s and f_r, the imprecise control of the vehicle, and inaccuracies in the tracking system. The differences between the simulation and the real vehicle can never be eliminated as mentioned in previous sections, but they can be reduced. A more sophisticated dynamics model of the RC tank that took into account the bending of the tracks and the slip along the

floor would enhance the match between f_s and f_r and therefore have reduced the likelihood of missing a gate.

The problems stemming from the imprecise vehicle control could have been alleviated by purchasing a more sophisticated vehicle. As mentioned earlier, the real vehicle used in this thesis was a modified children's toy. The speeds of its tracks, and therefore, its behavior could only be approximated because there was no direct access to these speeds through the remote control. The remote only supported steering or changing speed, and it did not support both actions simultaneously. While this behavior was mimicked in the simulation, the inaccuracies of the control contributed to the error between the simulated and real positions.

The inaccuracies stemming from the tracking system resulted from the size of the tracking squares and the difference between the calibrated locations in the camera's scene of the atrium and the actual positions of these locations. This meant that the distance between two points of interest on the atrium floor, as calculated by the camera tracker, was slightly different than their actual distance. Furthermore, each colored tracking square on the tank was three inches on a side, bounding the minimum resolution of the tank position. These discrepancies introduced errors in the positions reported by the tracker that could easily be improved in a future version of the system.

However, despite all these flaws - the simplistic dynamics model, the imprecise, non-repeatable vehicle control and the inaccurate tracker - the VR aided teleoperation system clearly outperformed typical teleoperation and very nearly matched the feeling and usability of direct control. From the perspective of the operator, the VR aided teleoperation system was as responsive as direct control, and there was no apparent effect of lag. Furthermore, the operator in the VR aided teleoperation system suffered less fatigue as the vehicle control was continuous instead of stop and go, and the field of view around the vehicle was larger than offered by vehicle mounted cameras.

The clear improvement netted by employing the VR aided teleoperation system over the typical teleoperation systems provides the impetus for this further development. Based on our experience, we feel that VR aided teleoperation is a promising direction for vehicle teleoperation that, with continued development, could be made part of a robust system suitable for broad application.

FUTURE WORK

The test results show that VR aided teleoperation is an improvement over camera aided teleoperation for basic maneuvers in a relatively static environment. We believe that this method would perform even better if applied to a real vehicle, with more precise controls. Ideally, if another tracked vehicle were used, it should be capable of receiving control for each track separately. Beyond tanks, other types of vehicles, such as cars, airplanes, or boats, would be teleoperated with the VR aided system. While each of these vehicles has different dynamics behaviors, we believe that our method could be applied equally well to all of these vehicle types. An analysis comparing the results of these four types of vehicles would illustrate the wide applicability of VR aided teleoperation.

If two vehicles were controlled in the same space, the issues of using the VR aided teleoperation system for remote collaboration could be explored. This collaboration task could be cast in the mold of a teleoperated race of the two vehicles around the test area. Each driver would see a virtual representation of their vehicle and the other vehicle driving around a virtual racetrack environment. The introduction of dynamic elements into the scene is an important step in the development of this method.

The system would be improved further by using a more sophisticated vehicle dynamics model. The inclusion of a fully 3D dynamics model would allow for the simulation of vehicles traveling over realistic terrain such as hills and valleys. Higher fidelity vehicle dynamics models would improve the match between the simulation and the real vehicle and would therefore improve the overall system performance. Creating this type of dynamics model would require more rigorous testing of the vehicle in order to determine the appropriate parameters for the simulation.

Another way to improve the system performance would be to use a more sophisticated correction algorithm. Although the wagon tongue did produce good results, it may prove to be insufficient with more extreme maneuvers. In addition, it was developed in

1970 and several useful path following technologies have been developed since then. The most intriguing one to the author is the use of a genetic algorithm to follow the path. With this model, the vehicle could be trained on how best to follow the path laid down by the simulated vehicle.

A final, but important, task would be to include methods to accommodate changes in the environment around the vehicle. This would be accomplished by mounting various sensors on the vehicle and using the information from these sensors to modify the virtual world as quickly as possible. Additionally, nearly real time virtual terrain generation would be included to reduce the static nature of the current virtual world used in the thesis. Before this system can be widely and commercially deployed, the capability to account for variations in environment would need to be developed and included. The promising results of this thesis provide impetus to tackle this work.

REFERENCES

1. Thorpe, Charles et. al., "Vehicle Teleoperation Interfaces," The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 2001.
2. "Remote-controlled Robots Search World Trade Center Rubble," JOM, copyright by the Minerals, Metals & Materials Society, vol. 53(12), p. 4-7, 2001.
3. Nguyen, Laurent et. al., "Virtual Reality Interfaces for Visualization and Control of Remote Vehicles," Autonomy and Robotics Area at NASA Ames Research Center, Moffet Field, CA, 2000.
4. "Pioneer UAV History Page," Pioneer UAV Website, www.aaicorp.com/pui/history.htm, visited June 17, 2003.
5. "Mars Pathfinder Rover," NASA NSSDC Master Catalog: Spacecraft, http://nssdc.gsfc.nasa.gov/database/MasterCatalog?sc=MESURPR, visited June 2, 2003.
6. Card, Orson Scott, *Ender's Game,* TOR, New York, NY, 1991.
7. "Gates Underwater Products Website," www.gateshousings.com, visited June 6, 2003.
8. Grant, Rebecca, "Reach-Forward," Air Force, Journal of the Air Force Association, vol. 85 no. 10, October 2002.
9. Walters, Brett et. al., "Modeling the Effects of Crew Size and Crew Fatigue on the Control of Tactical Unmanned Aerial Vehicles (TUAVS)," Micro Analysis and Design, Inc., Boulder, CO, 2000.
10. Rastogi, Anu et. al., "Virtual Telerobotic Control," University of Toronto Ergonomics in Teleoperation and Control Lab, Toronto, ON, 1993.
11. Piguet, Laurent et. al., "The Virtual Vehicle Interface: a dynamic, distributed and flexible virtual environment," Intelligent Mechanisms Group at NASA Ames Research Center, Moffet Field, CA, 1996.
12. Lane, J. C. et. al., "Advanced Operator Interface Design for Complex Space Telerobots," University of Maryland Space Systems Laboratory, College Park, MD, 2001.
13. Fong, Terrance et. al., "Novel Interfaces for Remote Driving: Gesture, Haptic and PDA," The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 2001.
14. Fong, Terrance et. al., "Multi-Robot Remote Driving with Collaborative Control," The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 2002.
15. Fong, Terrance et. al., "Collaborative Control: A Robot-Centric Model for Vehicle Teleoperation," The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 1998.
16. Draper, Mark H., "Multi-Sensory and Visualization Techniques Supporting the Control of Unmanned Air Vehicles," Air Force Research Laboratory, Wright-Patterson AFB, OH, 2000.
17. Hainsworth D.W., "Teleoperation User Interfaces for Mining Robotics," CSIRO Exploration and Mining, Brisbane, Australia, 1999.

18. "Jerry Isdale's What is VR Page," http://vr.isdale.com/WhatIsVR/noframes/WhatIsVR4.1-A.html, visited June 21, 2003.
19. Cruz-Neira, Carolina, "Virtual Reality Based on Multiple Projection Screens: the CAVE and its Applications to Computational Science and Engineering," University of Illinois at Chicago, 1995.
20. Collins, Robert T. et. al., "Site Model Acquisition and Extension from Aerial Images," University of Massachusetts Department of Computer Science, Amherst, MA.
21. Kroll, C. V.; Roland, R. D., Jr., "A preview-predictor model of driver behavior in emergency situations," Interim technical report, Cornell Aeronautical Laboratory, Inc., Buffalo, N.Y., 102 p, Sponsor: Bureau of Public Roads, Traffic Systems Research Division, Washington, D.C., Report No. CAL VJ-2251-V-6. UMTRI-15397, 1970.
22. Bernard, James et. al., "Evaluation of Vehicle/Driver Performance Using Genetic Algorithms," Iowa Center for Emerging Technology, Iowa State University, SAE Paper 980227, 1998.
23. "VRJuggler Website," www.vrjuggler.org, visited June 9, 2003.
24. Knutzon, Jared, "Tracking and Control Design for a Virtual Reality Teleoperation System," Computer and Electrical Engineering Department, Iowa State University at Ames, August 21, 2003.
25. "MegaTech Website: M1A1 Desert Storm," www.megatech.com/product.php?ID=7904, visited January 27, 2003.
26. "Digital Mars Website: WinIO," http://www.digitalmars.com/rtl/winio.html, visited May 2, 2003.
27. "Iowa State University Virtual Reality Applications Center Website," www.vrac.iastate.edu, visited June 9, 2003.
28. "Logitech Website: Logitech QuickCam Pro 4000," http://www.logitech.com/index.cfm?page=products/productlist&crid=20&countryid=19&languageid=1, visited April 2, 2003.
29. "The Spy Store Website: WVCS-1C Micro-Miniature Wireless Pinhole Camera," http://www.thespystore.com/wirelessvideo.htm, visited June 17, 2003.
30. "Intel Research Website: OpenCV," http://www.intel.com/research/mrl/research/opencv, visited May 4, 2003.

ACKNOWLEDGEMENTS

I would like to thank my wife, Jessica Walter, for helping me with revising this thesis and providing ongoing encouragement in the face of relentless editorial corrections. Special thanks go to my friend and research partner, Jared Knutzon, for his effort in designing his parts of the system and helping me perform the overall test runs. I appreciate the guidance of my committee members, specifically Dr. Adrian Sannier and Dr. Jim Oliver for their advice on the general system design, and Dr. James Bernard for his knowledge of vehicle dynamics. I would like to thank Kevin Teske and Glen Galvin for technical support and guidance in the making the video of the test runs. Thanks also to Rachael Norgaard for creating the models of the atrium and the tank. I would like to acknowledge Dr. Carolina Cruz-Neira and the VRJuggler team for the libraries used in this thesis and Chad Spencer for his assistance in creating the OpenCV image delaying code. Thanks too to Mark Knight for his contributions to the image generation code and Chad Austin for his help in tracking down a particularly nasty bug in my code. Finally, I would like to express my gratitude to my family for their lifelong support.