ABSTRACT

Title of Thesis:	A Case for Simulation: An Evaluation of the Use of Player and Gazebo to Identify Key Factors Contributing to Success During the 2004 and 2005 DARPA Grand Challenge
Degree candidate:	Jason Clair Allen
Degree and Year:	Master of Science 2010
Thesis directed by:	Dr. David Hibler, Professor, Department of Physics, Computer Science, and Engineering

The foundation of this research is a work of critical scholarship. An analysis of objective evidence available through review of published records was performed. The analysis supports a conclusion that system integration was the fundamental problem of the Grand Challenge. During the analysis, key factors contributing to the success of teams participating in the 2004 and 2005 DARPA Grand Challenge were identified. The use of simulation as a tool which would have allowed teams to identify some of these key factors prior to the Grand Challenge was proposed. Simulations were designed to evaluate selected key factors using Player and Gazebo, free and open source software for robot and sensor applications. The use of simulation was effective, however successful simulation was only possible after many problems with the applications Player and Gazebo were resolved. The use of simulation as a tool to reduce cost, increase emphasis on artificial intelligence, and decrease emphasis on system integration in the development of autonomous vehicles is proposed.

A Case for Simulation: An Evaluation of the Use of Player and Gazebo to Identify Key Factors Contributing to Success During the 2004 and 2005 DARPA Grand Challenge

By

Jason Clair Allen

Thesis submitted to the Graduate Faculty of
Christopher Newport University in partial
fulfillment of the requirements
for the degree of
Master of Science
2010

Approved:

David Hibler, Chair ______________________________

David E. Game ______________________________

Lynn Lambert ______________________________

PREFACE

The foundation of this research is a work of critical scholarship published as Technical Report "A Case for Simulation: An Evaluation of the Use of Player and Gazebo to Identify Key Factors Contributing to Success During the 2004 and 2005 DARPA Grand Challenge (CNU Technical Report PCSE-2010)", herein referred to as "the Technical Report". The Technical Report provides relevant technical data, justification for conclusions, and resolution of discrepancies supporting this research.

DEDICATION

I would like to dedicate this research

to my wife, Lisa,

to my mother, Gloria, and

to the civilizations which flourished in the infancy of the Milky Way Galaxy, and whose stars died, giving the Earth her iron and silicon.

ACKNOWLEDGEMENTS

I would first like to thank the members of my thesis committee: Dr. David Hibler, Dr. David E. Game, and Dr. Lynn Lambert. I would especially like to thank Dr. David Hibler, my advisor and the chair of my thesis committee, for allowing me to complete and document the analysis supporting this research. I would like to thank Dr. Antonio Siochi for his timely assistance as Graduate Program Coordinator. Finally, I would like to thank Ms. Lyn Sawyer for the continuous assistance and support she has offered since I first decided to undertake my graduate education.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I. INTRODUCTION

The Defense Advanced Research Projects Agency (DARPA) established the Grand Challenge to “promote innovative technical approaches that will enable the autonomous operation of unmanned ground combat vehicles.” ([1] and [2], p. 4). DARPA described the Grand Challenge as a race during which autonomous vehicles would be required to “navigate from point to point in an intelligent manner so as to avoid or accommodate obstacles and other impediments to the completion of their missions.” ([1]) or, worded slightly differently, to “navigate from point to point in an intelligent manner to avoid or accommodate obstacles including nearby vehicles and other impediments.” ([2], p. 4).

No challenge vehicle successfully completed the 2004 Grand Challenge Event (GCE).

Following the 2004 GCE, DARPA reported ([3], pp. 1 - 2)[1]:

> **Rationale for Using Congressional Prize Authority for Autonomous Ground Vehicle Development**
>
> Following a series of studies, and influenced by a Congressional directive[1], DARPA determined that the first use of the Congressional prize authority would be in the area of autonomous ground vehicles with the following goals:
>
> - Increase the number of performers working on autonomous ground vehicle technologies.
> - Provide DoD access to new talent, new ideas, and

innovative technologies by motivating and enlisting innovators that would not normally work on a DoD problem.

- Accelerate autonomous ground vehicle technology development in the United States in the areas of sensors, navigation, control algorithms, vehicle systems, and systems integration.

1 Congress expressed a clear interest in accelerating unmanned vehicle capabilities and, in fact, set a goal for the Department of Defense: The Fiscal Year 2001 National Defense Authorization Act states, "It shall be the goal of the Armed Forces to achieve the fielding of unmanned remotely controlled technology such that...by 2015, one-third of the operational ground combat vehicles of the Armed Forces are unmanned." Given the aggressive timeline in the directive, DARPA determined that organizing a prize authority event would be the quickest and most cost-effective approach to stimulate innovation and expand the research community in autonomous ground vehicle technologies.

Based on the results of the 2004 GCE, DARPA held a second Grand Challenge in 2005. On October 8, 2005, four teams participating in the 2005 GCE successfully completed the course. A fifth team completed the course the next day. DARPA awarded the prize of $2 million to Team 2005-16, the first team to complete the course.

The successful completion of the 2005 GCE was widely considered to be a significant achievement. DARPA stated: "The results prove conclusively that autonomous ground vehicles can travel long distances over difficult terrain at militarily relevant rates of speed." ([5]).

However, the actual goal of the Grand Challenge was concealed by the format of the Grand Challenge as a race. As noted by DARPA[1], the Fiscal Year 2001 National Defense Authorization Act states: "It shall be a goal of the Armed Forces to achieve the fielding of unmanned, remotely controlled technology such that...by 2015, one-third of the operational ground combat vehicles are unmanned." ([4], p. 46).

DARPA published rules prior to the 2004 and 2005 GCE. The rules reported a problem statement which was revised continuously prior to the 2004 GCE, through the 2004 GCE itself, to successful completion of the 2005 GCE. DARPA published clarifications to the rules, but also revised course length, maximum corrected time, and expectation of obstacle avoidance:

- DARPA revised the maximum corrected time of the 2004 GCE from greater than 10 hours to 10 hours on April 1, 2003, several weeks after the official start of the 2004 GCE on February 22, 2003.
- DARPA revised the proposed 2004 GCE course length continuously from the official start of the 2004 GCE on February 22, 2003 through the publication of revision "5 January 2004" of the 2004 GCE rules on January 5, 2004 from 300 miles to less than 210. The reported 2004 GCE course length was 142 miles.
- DARPA stated the 2005 GCE course length would not exceed 175 miles. The reported 2005 GCE course length was 131.6 miles.
- Prior to the 2004 QID or GCE, DARPA stated: "DARPA intends to clear the

Challenge Route of non-Challenge traffic and obstacles, but can not guarantee that there will be no non-Challenge traffic, obstacles, or humans on the Challenge Route... Sensing and processing designs must be able to avoid collisions with any obstacle, moving or static, that may exist on the route." ([1] and [6]). As a result, DARPA established an expectation that teams participating in the 2004 GCE would not encounter obstacles deliberately placed on the 2004 GCE course by DARPA to test and evaluate challenge vehicle obstacle detection and avoidance.

- Prior to the 2005 GCE, DARPA stated: "The vehicle must avoid collisions with any obstacle, moving or static, on the route. DARPA will place obstacles along the route to test obstacle avoidance capabilities." ([2], p. 22). DARPA later stated: "...vehicles were required to detect and avoid obstacles along the route..." ([7], p. 4) and "The 132-mile route contained a series of graduated challenges beginning with a dry lake bed, narrow cattle guard gates, narrow roads, tight turns, highway and railroad underpasses... Vehicles passed through tunnels and avoided more than 50 utility poles situated along the edge of the road." ([7], p. 9). Although both of these claims are true, DARPA did not report obstacles were placed along the route "to test obstacle avoidance capabilities".
- DARPA placed obstacles on both the 2004 Qualification, Inspection, and Demonstration (QID) and 2005 National Qualification Event (NQE) courses to evaluate challenge vehicle obstacle detection and avoidance capabilities.

In addition, a detailed course analysis indicates course difficulty changed between

2004 and 2005. For various reasons, the 2005 GCE course was not as difficult as the 2004 GCE course. For example:

- The 2005 GCE course was located on terrain with much less slope overall than the 2004 GCE course.
- The 2005 GCE RDDF eliminated extreme lateral boundary offset and course segment lengths similar to those defined by the 2004 GCE RDDF.
- The length of the 2005 GCE course was 131.6 miles, less than the 142-mile length of the 2004 GCE course. The maximum corrected time was not reduced to ensure an "average minimum speed of approximately 15 - 20 mph" ([3], p. 2) was achieved for either event.
- The 2005 GCE RDDF defined more waypoints than the 2004 GCE RDDF. As a result of this increase and the decrease in length of the 2005 GCE course compared to the 2004 GCE course the average distance between adjacent waypoints decreased and waypoint density increased.
- The 2005 GCE course was, overall, "smoother" than the 2004 GCE course. As a result of the increase in waypoint density, the 2005 GCE RDDF required more changes in bearing than the 2004 GCE RDDF, but these changes in bearing were less severe than those required by the 2004 GCE RDDF.
- The 2005 GCE RDDF defined forced deceleration lanes before significant terrain features.

In summary, the evidence supports a conclusion that the 2005 GCE course was engineered and “groomed” to reduce its difficulty and increase the opportunity that at least one challenge vehicle would successfully complete the course.

The problem statement reported by DARPA was ostensibly one of autonomous navigation. Teams which participated in the 2004 and 2005 GCE were required to develop an autonomous vehicle with a controlling intelligence able to:

- distinguish the course from terrain that was unnavigable or was declared off-limits[2] based on detailed course information withheld until two hours prior to the event, and which identified the course and established speed limits which the controlling intelligence was required to observe, and
- navigate the course identified, while
- detecting and avoiding unintended obstacles encountered on the course,
- in less than ten hours at an average speed greater than 15 mph.

To “win”, teams were required to develop the autonomous vehicle which completed the course in the least “maximum corrected time”.

These problems were, to various degrees, solved prior to the Grand Challenge. As a result, based on a comprehensive review of published records, the author concluded the problem statement reported by DARPA concealed the fundamental problem of the Grand Challenge (“fundamental problem”), which was *system integration:*

- The Grand Challenge required teams to integrate sensor data intelligently, and integrate computer hardware with various sensors and the research platform on which they are mounted. Some teams which focused on the fundamental problem were *potentially disruptive*.

- The vast majority of failures during the 2004 and 2005 GCE were not failures of artificial intelligence, but system integration failures.

The Grand Challenge was therefore more a test of successful system integration than successful artificial intelligence, and the conditions of the Grand Challenge favored teams with greater experience and sponsorship.

As a result, in the broader context it is important to place the Grand Challenge in perspective, and determine what, exactly, the successful teams achieved in 2005.

In general, teams which participated in the Grand Challenge described the technical details and information concerning their approach to solving the fundamental problem in published records. Some team solutions demonstrate technical achievement in system integration. Analysis indicates that most teams spent a significant amount of money on their solutions to the problem, and that the total cost of team solutions represents an investment which exceeds what the Department of Defense may reasonably be expected to pay to procure them.

In addition, team challenge vehicles were mindless automatons incapable of true autonomous navigation, although some team challenge vehicles were capable of autonomous obstacle detection and avoidance and path detection while following a "bread crumb" trail.

As a result, team solutions were impractical solutions to the problem of autonomous navigation.

Although DARPA reported the Grand Challenge proved "...conclusively that autonomous ground vehicles can travel long distances over difficult terrain at militarily relevant rates of speed." ([5]), significant progress toward the goal of the Department of Defense to have one-third of operational ground combat vehicles unmanned by 2015 has not been made in the years since the 2005 GCE.

The author gained a greater appreciation for the difficulty teams participating in the 2004 and 2005 GCE must have faced while completing this research. However, the perspective of this research is that the Grand Challenge was a failure, despite the fact that prize money was awarded by DARPA, for the following reasons:

- the technical achievement was consistent with the state of the art,
- the development of basic algorithms and strategies for control of an autonomous vehicle was not the focus of the Grand Challenge,
- the cost of proposed solutions far exceeds what the Department of Defense may reasonably be expected to pay to procure them, and
- DARPA failed to structure the Grand Challenge to ensure long-term realization of its stated goals, and
- significant progress toward the actual goal has not been made in the years since the 2005 GCE.

The author asserts the use of simulation as a complement to the Grand Challenge would have provided teams participating in the Grand Challenge with a way to identify key factors contributing to success prior to field trials, increased focus on the development of basic strategies and algorithms to enhance the intelligence of autonomous vehicles, and provided a way to increase the competitiveness of the Grand Challenge by “leveling the playing field”, allowing teams with less experience or sponsorship to compete on a more even basis with teams with significant experience or sponsorship.

CHAPTER II. THE CASE FOR SIMULATION

II.A. Rationale for the use of simulation

II.A.1. Number of potential participants

Perhaps the best rationale for the use of simulation may be provided by reviewing the interest with which the announcement of the Grand Challenge was greeted by potential participants ([8] and [3]):

DARPA received 106 applications for the 2004 GCE. Eighty-six teams submitted technical proposals by the deadline established by DARPA. Of the 86 technical proposals received, 45 teams proposed autonomous vehicles of interest to the DOD[3].

However, it did not appear as if all 45 teams would have vehicles ready in time to participate in the 2004 GCE. DARPA evaluated the technical proposals for 19 teams as "completely acceptable", and selected these teams for advancement to the next phase of the Grand Challenge. DARPA evaluated the technical proposals for an additional 26 teams as "possibly acceptable" and established a site visit process to determine the final teams[4].

On December 19, 2004, DARPA announced 25 teams from around the United States were selected to participate in the next phase of the Grand Challenge: Qualification, Inspection, and Demonstration (QID). The QID was used to determine the final 20 participants for the Grand Challenge. The 25 teams that passed the technical proposal review process were invited to the QID to take place March 8 through 12, 2004. Twenty-one teams participated. The QID comprised several distinct activities: a safety and technical inspection of the team challenge vehicle[5]; a separate practice area; and a demonstration course which was approximately 1.4-mile long that the vehicle was required to traverse.

The demonstration course allowed DARPA to evaluate the ability of each challenge vehicle to sense a series of static and moveable obstacles representative[6] of those that might be found on the actual 2004 GCE course, and navigate a course described by a series of adjacent waypoints. Each vehicle was ranked according to its overall time to complete the course, and point deductions[7] were taken for impacting obstacles, exceeding established speed limits, or deviating from the established course.

Over a five day period, eight teams completed the 2004 QID course, nine teams partially completed the course, two teams terminated within the starting chute area, and two teams officially withdrew. On March 12, 2004 DARPA announced 15 of 21 teams which participated in the 2004 QID qualified for the 2004 GCE.

In summary, only 15 of 106 applicants were allowed to participate in the 2004 GCE. No challenge vehicle which qualified was able to complete the 2004 GCE. To achieve this result, DARPA effectively eliminated 91 of 106 potential applicants, removing incentive those teams may have had to participate in the Grand Challenge and provide access to “new talent, new ideas, and innovative technologies” or develop autonomous ground vehicle technologies in the areas of “sensors, navigation, control algorithms, vehicle systems, and systems integration”.

If the purpose of the Grand Challenge was that stated by DARPA (see Chapter I.), the Grand Challenge was either a marginal success or an abject failure, depending on perspective. From one perspective, the Grand Challenge was a marginal success because

DARPA was able to achieve some of its goals, although to a limited extent. The author considers it likely:

- the number of individuals, groups, or organizations working on autonomous ground vehicle technologies increased during the years before and immediately after the Grand Challenge,
- the Grand Challenge motivated individuals that would not normally work on a "DOD problem", and
- the Grand Challenge resulted in some development of autonomous ground vehicle technologies in the areas of sensors, navigation, control algorithms, vehicle systems, and systems integration.

The author is aware of no evidence which directly supports or refutes these assertions. For example, the author is aware of no survey of the robotics community before and after the Grand Challenge which supports an assertion that the number of individuals, groups, or organizations working on autonomous ground vehicle technologies increased during the years before and immediately after the Grand Challenge, and has since decreased. However, the author considers it unreasonable to assert, given the published record, that some progress has not been made, in particular in the development of autonomous ground vehicle technologies. The difference, to the author, lies in the intention and meaning of words such as "accelerate", or the duration of time during which DARPA expected the Grand Challenge to provide the DOD with access to "new talent, new ideas, and innovative technologies".

From another perspective the Grand Challenge was an abject failure. Although four teams successfully completed the 2005 GCE, the three teams with the best times, including the winner, Team 2005-16, were representatives of a single academic institution in all but name: Carnegie Mellon University. Team 2005-16 did not participate in the 2004 GCE. The Team 2005-16 team leader was a faculty member at Carnegie Mellon University when the Grand Challenge was officially announced on February 22, 2003 and transferred to Stanford University in July, 2003 approximately eight months before the 2004 GCE took place on March 13, 2004.

The Grand Challenge *might* have been a very close competition and a significant number of the teams participating in the Grand Challenge *might* have successfully completed the 2004 or 2005 GCE course, demonstrating proficiency in the skills required to develop an autonomous vehicle. As a result, the DOD *might* have gained increased or lasting access to "new talent, new ideas, and innovative technologies" or DARPA *might* have accelerated the development of autonomous ground vehicle technologies in the areas of "sensors, navigation, control algorithms, vehicle systems, and systems integration" to a greater extent.

By restricting the number of participants to the few teams with experience or sponsorship which were able to field a research platform, DARPA virtually guaranteed the eventual outcome of the 2004 and 2005 GCE: the only team which successfully completed the 2005 GCE and which was not closely tied to Carnegie Mellon University was Team 2005-06, which placed fourth during the 2005 GCE, and emerged as the only disruptive team which participated in either the 2004 or 2005 GCE.

II.A.2. Risk of rollover

Objective evidence supports a conclusion the 2005 GCE course was engineered and “groomed” to be less difficult than the 2004 GCE course to reduce the risk of rollover. Although some teams were aware of the risk of rollover, the author has not encountered an alternate detailed route analysis which indicates a team was aware of the extent to which the 2005 GCE course was engineered by DARPA. Although the author was unable to determine the total cost of team challenge vehicles, published records report costs from $35,000 to in excess of $3 million. As a result, the potential impact due to rollover was significant.

II.A.3. Stopping distance and field-of-view limitations

Review of team technical proposals supports a conclusion the teams had difficulty visualizing the interaction of their challenge vehicles with the environment, with potentially significant consequences, such as challenge vehicles traveling at speeds exceeding their stopping distance or an inability to adequately detect obstacles during a turn or on sloped terrain.

The use of simulation is proposed specifically to address these deficiencies. The use of simulation would:

- support autonomous vehicle development without requiring the sponsor of such research to engineer a course able to be completed by research platforms consistent with the state of the art,
- encourage participation in autonomous vehicle development by individuals and

institutions not having the resources required to develop a research platform,

- minimize the risk of rollover to research platforms until the necessary technologies were developed to enable the controlling intelligence to adequately evaluate the risk,
- allow teams to visualize the interaction of the research platform with the environment,
- provide teams with a way to identify some key factors which does not require procurement of a research platform or sensors to perform test and evaluation,
- increase focus on the development of basic algorithms and strategies,
- provide a way to increase competitiveness by “leveling the playing field”, and
- provide a tool which would help ensure long-term realization of DARPA's stated goals.

Overall, the use of simulation would allow teams to focus on the basic algorithms for using environment and geolocation sensors, and place the focus of autonomous vehicle development on artificial intelligence, not system integration, and “level the playing field” between teams with more experience and those with less experience.

II.B. Selection of the simulation environment

The most common approach to integrating hardware and software in use by teams which participated in the 2004 and 2005 GCE may be described as a “mixed” or “composite” architecture, where disparate, distributed elements were integrated using client-server relationships. These elements can be reproduced through the use of simulation. Therefore, one of the most important considerations for developing an architecture for simulation of an autonomous vehicle was the simulation environment itself. The author developed a list of requirements and desired features of the simulation environment, the first and most important of which was that it be free for academic use, with a preference for Free and Open Source Software (FOSS). Commercial software was not evaluated. Other requirements and desired features of the simulation environment included (in no particular order):

- Cross-platform availability.
- A graphical user interface using OpenGL.
- High-fidelity, rigid-body three-dimensional (3D) physics simulation, including collision detection and 6 degrees of freedom.
- Support for popular image formats and cameras.
- Terrain rendering.
- An active user community and developer base.

A review of available FOSS alternatives revealed the Player Project satisfied the author's requirements, with some caveats. In addition, the Player Project provided other desirable features, such as the ability to use XML files to configure the simulation, and could be extended by the author. As a result, the author selected the Player Project, specifically the applications Player and Gazebo, to complete this research.

II.C. DARPA evaluation of the use of simulation

The author is unaware of any published record that reports DARPA, following the

2004 or 2005 GCE, concluded that high-fidelity simulation was necessary, desirable, or even useful. From one perspective, this is certainly true. By engineering the 2005 GCE course, DARPA was able to create conditions which made it possible for several teams to successfully complete the 2005 GCE. In addition, a number of key factors contributed to team success (herein referred to as "key factors contributing to success" or "key factors").

However, the fundamental problem of the Grand Challenge was system integration, not autonomous navigation or artificial intelligence, and the cost of fielding a research platform was prohibitive - out of reach for most individuals and even most academic institutions without corporate sponsorship. Although the author was unable to determine the total cost of team challenge vehicles, available evidence supports a conclusion that team challenge vehicles represented a considerable investment in terms of time and material resources.

Via the "Team Resources" section of the archived Grand Challenge 2004 website ([11]), DARPA hosted an "Outside Resources/Links" link to technical resources such as the Carnegie Mellon Navigation Toolkit (CarMeN) and many other libraries, applications, and utilities written to solve portions of the autonomous vehicle development problem. DARPA did not, however, include the the Player Project on the list of technical resources. None of the technical resources to which DARPA referred provided a simulation environment similar to the Player Project.

II.D. Team evaluation of the use of simulation

Via 2005 GCE Standard Question (SQ) 2.5.1[8] DARPA requested teams: "Describe the testing strategy to ensure vehicle readiness for DGC, including a discussion of component reliability, and any efforts made to simulate the DGC environment." Sixteen of 48 teams which participated in the 2004 QID or GCE or 2005 GCE referred to the use of simulation. Six of 48 specifically referred to the Player Project or to a simulation environment similar to the Player Project.

- Team 2005-02

Team 2005-02 stated: "To support bench testing, a simple vehicle simulator component was devised that sends out position- and velocity-related JAUS messages as if the vehicle were moving through an RDDF corridor." ([12], pp. 616 - 617).

- Team 2005-04

Team 2005-04 stated: "Portions of the software were tested on different simulation and emulation environments. Two specific simulation environments were developed for testing obstacle avoidance. One was a simple, flexible 2-D package for initial testing. The second was based on the Player/Gazebo environment and with the 3-D developments made, could actually include terrain configurations from real data." ([13], p. 6).

Team 2005-04 later referred to the use of simulation ([14]), but not specifically to the Player Project.

- Team 2005-05

Team 2005-05 stated: "[The challenge vehicle controlling intelligence] could be driven by real-time sensor data, by a simple simulator, or from previously recorded log data. The simulator was invaluable for debugging the high-level behaviors of the planner, but its models were not accurate enough to tune the low-level controllers. The replay mode allowed us to debug the ladar obstacle filters and the state estimators in a repeatable way, without having to drive the vehicle over and over." ([15], p. 531).

- Team 2005-09

Team 2005-09 referred to the use of simulation as part of their autonomous vehicle development process throughout their technical proposal ([16]), but did not refer to a specific simulation environment. Team 2005-09 did, however, refer to the use of simulation to "fit" the vehicle's performance in simulation to real-world performance: "The behavior of [the challenge vehicle] during the test would then (1) drive refinements to the simulator to more accurately reflect the demonstrations and (2) lead to new improvements in the software." ([16], p. 6).

Team 2005-09 later stated: "When a problem was found or a new phenomenon identified, it was first modeled in the simulation environment. With a simulation of the problem or new phenomenon in hand, the body of operational code was adjusted to deal with it. Once proven in simulation, the robot was field tested to evaluate the changes, and improvements were fed back to the model. A result of the model-build-test approach was that the model grew in fidelity and became a lasting repository of project experience." ([17], p. 835).

- Team 2005-11

Team 2005-11 stated: "[Challenge vehicle] testing included both physical and software-only simulation runs." and "Multiple simulation runs, particularly obstacle avoidance scenarios, were executed prior to field testing." ([18], p. 9). The author does not consider this reference to "simulation" to be a reference to a simulation environment similar to the Player Project.

- Teams 2005-13 and 2005-14

Teams 2005-13 and 2005-14 stated: "In addition to these system tests, [the challenge vehicle] has tested for software endurance via simulation..." and "Planned tests include end-to-end race day simulations..." ([19], p. 15, and [20], p. 15). The author does not consider this reference to "simulation" to be a reference to a simulation environment similar to the Player Project.

Teams 2005-13 and 2005-14 later referred to testing in simulation of control routines developed using Simulink ([21], p. 471).

- Team 2005-15

Team 2005-15 stated: "...we have simulation modules that allow for testing of all other modules, with the exception of the data acquisition modules." ([22], p. 6) and "In the lab environment, we use the GAZEBO toolkit to perform system and vehicle simulations." ([22], p. 11).

Team 2005-15 later stated: "With the use of the Gazebo simulator ... and tools for playing back recorded vehicle data, much of the debugging and development could be carried out on individual laptops; so development work could continue when the vehicle was not available." ([23], p. 582).

- Team 2005-17

Team 2005-17 stated: "A vehicle simulator is included in [the challenge vehicle] software suite. The simulator provides a test environment that emulates the physical environment in which the vehicle operates. Daily builds of the software are tested against a collection of test cases gathered from the real world. Developers perform unit level testing of changes to the software using the combination of the vehicle simulator and visualization tools included in the software suite.” ([24], p. 10).

Team 2005-17 later stated ([25], p. 563):

> [The challenge vehicle's simulator] is a physics-based simulator developed using the Open Dynamics Engine physics engine. Along with simulating the vehicle dynamics and terrain, [the simulator] also simulates all the onboard sensors. It populates the same [queues] with data in the same format as the sensor drivers. It also reads vehicle control commands from [queues] and interprets them to have the desired effect on the simulated vehicle.
>
> While [the simulator] is a physics-based simulator, such as Stage ... and Gazebo ... it has two interesting differences. First, [the simulator] does not provide any visual/graphical interface. The visualization of the world and the vehicle state is provided by the Visualizer module, discussed later. Second, [the simulator] also generates a clock, albeit a simulated one, using the [queues].

Team 2005-17 later stated: “By maintaining a system-wide simulated time, [the

Team 2005-17 simulator] is able to create a higher fidelity simulation than that provided by Stage and Gazebo. The computation in the entire system can be stopped by stopping the clock; and its speed can be altered by slowing down or speeding up the clock. This also makes it feasible to run the application in a single-step mode, executing one cycle of all programs at a time, thereby significantly improving testing and debugging." ([25], p. 563).

Team 2005-17 also stated: "Yet, testing in the current generation of simulation environments, such as [the Team 2005-17 simulator], Stage[,] ... and Gazebo ... is quite limited. While these environments are good for doing integration testing, their simulation abilities are quite limited in providing information about how the vehicle may perform in the real world, such as, in different terrains and weather conditions." ([25], p. 577).

- Team 2005-18

Team 2005-18 stated: "Two simulation environments are also used: a dynamic model of the vehicle motion (including traction) that is used for testing without sensory input and a Gazebo simulation environment." ([26], p. 9). Via a footnote on the same page, Team 2005-18 stated: "The Gazebo simulation environment was used relatively lightly due to the team's decision to focus on desert testing."

- Team 2005-19

Team 2005-19 referred to the use of simulation as part of their autonomous vehicle development process throughout the team technical proposal ([27]), but did not refer to a specific simulation environment. Team 2005-19 later referred to "simulated or logged data", "numerical simulation", and "a simulated course" ([28]), but did not refer to a specific simulation environment.

- Team 2005-20

Team 2005-20 stated: "[Team 2005-20] attempted to implement an open source robotic simulation environment to assist in the evaluation of the code prior to running on the robot. This proved to be ineffective since the overhead of the open source package swamped the limited computational resources available for real-time operation. Therefore, the real-time code had to be redone outside the open source environment. The final solution was to develop a simulator utilizing the Team ENSCO developed real-time code. The simulator estimates where the vehicle position would be based on the commands sent instead of reading its position from a GPS device, but is otherwise identical to the software on the robot." ([29], p. 15).

- Team 2005-21

Team 2005-21 stated: "Modeling and simulation of the [challenge vehicle] was done using ADAMS to determine vehicle performance over various size obstacles and to evaluate steering response at various vehicle speeds." and "Rockwell also developed a simulation environment that included all of the vehicle dynamics. This simulation was used to test the vehicle control interface, real-time path planner and behavior control. Similar to on the vehicle, a series of waypoint could be executed while avoiding planned obstacles. The 2004 race path was executed several times in this simulation environment to determine if the vehicle could navigate the entire path." ([30], p. 13).

Team 2005-21 later stated: "A full vehicle model of the truck was created in Advanced Dynamic Analysis of Mechanical Systems (ADAMS) by assembling subsystem models of suspensions, steering, chassis, and tires. A typical NATO Reference Mobility Model (NRMM) obstacle course with over 70 different obstacles of different sizes and shapes was used to evaluate the underbody clearance... The results of this simulation gave an idea about the truck's capability to maneuver through different obstacles at low speeds." ([31], p. 695).

Team 2005-21 participated in the 2004 GCE as Team 2004-23. Team 2004-23 was the only team which participated in the 2004 GCE to refer specifically to the use of a "simulation environment"[9]. Team 2004-23 stated: "A simulation model of the Challenge Vehicle has been developed and the software modules are being tested on the simulation environment." ([34], p. 11).

- Team 2005-22

Team 2005-22 stated: " A vehicle simulator program was also designed to test conditions and situations that would be difficult, if not impossible, for [the challenge vehicle] to encounter in Blacksburg. This program creates a virtual map and sensor data that is relayed to the actual pieces of software that control the vehicle. This simulator, along with information about [the challenge vehicle's] vehicle dynamics, tested the algorithms in a virtual space before ever placing them on the vehicle. It also allowed for testing during conditions where it would normally not be possible, such as at night or times when [*sic*]" ([35], pp. 12 - 13).

Teams 2005-22 and 2005-23 did not later refer to the use of simulation ([36]).

- Team 2005-23

Team 2005-23 stated: "A vehicle simulator program was also designed to test conditions and situations that would be difficult, if not impossible, for [the challenge vehicle] to encounter in Blacksburg. This program creates a virtual map and sensor data that is relayed to the actual pieces of software that control the vehicle. This simulator, along with information about [the challenge vehicle's] vehicle dynamics, tested the algorithms in a virtual space before ever placing them on the vehicle. It also allowed for testing during conditions where it would normally not be possible, such as at night or during heavy rain." ([37], p. 6).

Teams 2005-22 and 2005-23 did not later refer to the use of simulation ([36]).

II.E. Limits on the use of simulation

Although the approach discussed herein was implemented using Player and Gazebo, it is important to recognize limits imposed by the use of simulation. Several teams referred to specific limits on the use simulation:

- Models only approximate real world behaviors

Team 2005-05 stated: "The simulator was invaluable for debugging the high-level behaviors of the planner, but its models were not accurate enough to tune the low-level controllers." ([15], p. 531).

Team 2005-17 stated: "Yet, testing in the current generation of simulation environments, such as [the Team 2005-17 simulator], Stage[,] ... and Gazebo ... is quite limited. While these environments are good for doing integration testing, their simulation abilities are quite limited in providing information about how the vehicle may perform in the real world, such as, in different terrains and weather conditions." ([25], p. 577).

Team 2005-18 stated: "The Gazebo simulation environment was used relatively lightly due to the team’s decision to focus on desert testing.” ([26], p. 9). Although Team 2005-18 did not state their decision to focus on desert testing was driven by a limitation of Player and Gazebo, the team implied Gazebo did not represent desert terrain with sufficient fidelity for testing.

Based on the author's experience with Player and Gazebo, the extent to which models, including simulated worlds, terrain, and obstacles, approximate the real world or real world behaviors is more dependent on the accuracy of the model and availability of computing resources than on the simulation environment. The author notes teams participating in the 2004 and 2005 GCE may have had neither the time nor incentive to develop accurate models, but considers poor fidelity evidence of a resource allocation decision or a consequence of limited computing resources. The author does not consider sufficient evidence is available to conclude poor fidelity is due to an inherent limit on the use of simulation.

- The use of simulation is computationally intensive

Team 2005-20 stated: “[Team 2005-20] attempted to implement an open source robotic simulation environment to assist in the evaluation of the code prior to running on the robot. This proved to be ineffective since the overhead of the open source package swamped the limited computational resources available for real-time operation.” ([29], p. 15).

The author concluded an increase in processing power available to the challenge vehicle controlling intelligence between the 2004 and 2005 GCE was a key factor. The author asserts an increase in processing power may have addressed the limitation identified by Team 2005-20.

- Real time versus "simulated time" simulation

Team 2005-17 stated: "By maintaining a system-wide simulated time, [the Team 2005-17 simulator] is able to create a higher fidelity simulation than that provided by Stage and Gazebo. The computation in the entire system can be stopped by stopping the clock; and its speed can be altered by slowing down or speeding up the clock. This also makes it feasible to run the application in a single-step mode, executing one cycle of all programs at a time, thereby significantly improving testing and debugging." ([25], p. 563).

Although the author considers this a feature of the Team 2005-17 simulator, it does identify a limitation inherent in Player and Gazebo: the simulation can be paused or slowed by throttling the simulation time step, but not stopped without exiting the simulation environment, and neither Stage nor Gazebo can be run in single-step mode.

II.F. Advantages to the use of simulation

Several teams referred to specific advantages to the use simulation:

- Reproducibility

Team 2005-05 stated: "The replay mode allowed us to debug the ladar obstacle filters and the state estimators in a repeatable way, without having to drive the vehicle over and over." ([15], p. 531).

- Software development is independent of hardware development

Team 2005-15 stated: "With the use of the Gazebo simulator ... and tools for playing back recorded vehicle data, much of the debugging and development could be carried out on individual laptops; so development work could continue when the vehicle was not available." ([23], p. 582).

- The use of simulation increases the number of available test environments or conditions

Team 2005-22 stated: " A vehicle simulator program was also designed to test conditions and situations that would be difficult, if not impossible, for [the challenge vehicle] to encounter in Blacksburg. ... It also allowed for testing during conditions where it would normally not be possible, such as at night or times when [*sic*]" ([35], pp. 12 - 13).

Team 2005-23 stated: "A vehicle simulator program was also designed to test conditions and situations that would be difficult, if not impossible, for [the challenge vehicle] to encounter in Blacksburg. ... It also allowed for testing during conditions where it would normally not be possible, such as at night or during heavy rain." ([37], p. 6).

CHAPTER III. IDENTIFICATION OF SIMULATION TARGETS

The author identified the following potential “simulation targets” to determine if simulation could be used to evaluate the conclusions documented throughout this research.

III.A. Use Player and Gazebo to evaluate the rollover of a representative challenge vehicle entering 2004 GCE course segment 2570-2571-2572

Simulating 2004 GCE course segment 2570-2571-2572 would allow the author to test the conclusion that no challenge vehicle would have been able to make this turn at the RDDF-allowed speed of 60 mph and would have either rolled over or exceeded the lateral boundary offset and consequently left the course less than one kilometer (890.1 m), or less than two minutes (100 seconds), from the end of the course.

The risk of rollover can be evaluated in simulation by accelerating a realistic model of a challenge vehicle to the 2004 RDDF-allowed speed of 60 mph and then entering a simulation of 2004 GCE course segment 2570-2571-2572, and documenting:

- whether rollover occurs,
- whether the challenge vehicle exceeds the lateral boundary offset and consequently leaves the course, or
- whether the use of simulation provides no useful information.

If rollover occurs, the parameters under which rollover occurs in simulation can then be compared to real-world results to determine if the use of simulation would have enabled teams to identify the risk of rollover.

Realistically, this would require “fitting” a SSF to the challenge vehicle. The inclusion of sensors and computing hardware increases the weight of challenge vehicles, and the use of roof racks as mount points for sensors may have caused challenge vehicles to be “top-heavy” by raising their center of gravity (CG), either of which will affect SSF. Although it is possible to create a model of a challenge vehicle with the physical characteristics of a challenge vehicle in simulation, including dimensions and weight, creation of a realistic model of a challenge vehicle is not possible without knowing the relative positions and weights of the various components in use by the team.

For this reason, a simple model having a SSF matching a selected challenge vehicle was chosen. This model is described in detail in paragraph V.C.1. and Appendix F.

III.B. Modify Player and Gazebo to implement a two-material friction model and evaluate the stopping distance of a selected challenge vehicle

Implementing a two-material friction model would allow the author to modify the friction coefficient of challenge vehicle wheels and the surface of the course in simulation and more realistically evaluate the assertion that team challenge vehicles would not have been able to stop on obstacle detection due to the stopping distance of the vehicle. A two-material friction model would also allow Player and Gazebo to generically simulate low-friction surfaces like sand, mud, or rain-slicked roads, which may be useful for training the controlling intelligence to use one sensor to interpret others. See paragraph XI.A.

Realistically, this would require “fitting” a braking profile to a simulated challenge vehicle. The braking profile would have to be experimentally determined on a case basis.

III.C. Use Player and Gazebo to evaluate field-of-view limitations for selected sensors, specifically navigation RADAR

The author concluded effectively visualizing the interaction of the challenge vehicle with the environment was a key factor, and that lack of experience was a contributing factor.

A relatively simple simulation was designed to visualize the interaction of a selected challenge vehicle with the environment using obstacles DARPA identified as representative of obstacles challenge vehicles would encounter during the 2004 GCE. Specifically, the author chose to visualize the maximum distance between the path of travel in a constant-radius turn and the left- or right-limit of field-of-view, and demonstrate that sensors with a field-of-view of less than 40º should not have been selected as a primary obstacle avoidance sensor.

III.D. Use Player and Gazebo to evaluate the use of LIDAR, in particular the quality of the point map created by SICK LMS 200 and 291 LIDAR sensors, and increase in the number of SICK LMS 291 LIDAR sensors

The author concluded the increased use of high-quality LIDAR and STEREO sensors was a key factor because these sensors provide an accurate "point map" of the environment. The author considers this conclusion well-supported by the facts based on analysis and the success of teams which participated in the 2005 GCE.

Team 2005-06 successfully completed the 2005 GCE course using only two unknown SICK LIDAR sensors. All other successful teams used five LIDAR sensors during the 2005 GCE[10].

The two unknown SICK LIDAR sensors in use by Team 2005-06 were configured atypically compared to other successful teams which used LIDAR sensors, such as Teams 2005-13, 2005-14, and 2005-16. Team 2005-06 configured their LIDAR sensors to scan in a vertical plane, as opposed to a horizontal plane. The author therefore considers the orientation of LIDAR sensors to be testable, in addition to the number of LIDAR sensors. It is possible some patterns using fewer LIDAR sensors provide more useful information to the controlling intelligence than others using more LIDAR sensors.

III.E. Modify Player and Gazebo to simulate sensor "noise"

Simulated sensors are not subject to the same conditions encountered by research platforms. Rough terrain, rain, and fog, for example, are difficult to simulate realistically. However, several teams referred to these limitations specifically in their evaluation of the potential use of simulation. See paragraph II.E. As a result, the author identified simulation of sensor "noise" as a potential simulation target.

CHAPTER IV. GENERAL SIMULATION PROCEDURE

The general simulation procedure developed by the author is described as follows:

IV.A. Develop an installation procedure

An installation procedure was developed. This established a reproducible simulation environment and baseline for any changes. Development of the installation procedure is documented by Appendix A. The installation procedure is documented by Appendix B.

IV.B. Verify the installation procedure

The installation procedure was then verified. This ensured the simulation environment was reproducible. Verification of the installation procedure is documented by Appendix C.

IV.C. Verify Player and Gazebo using packaged world files, configuration files, and models

After developing the installation procedure in accordance with Appendix A, and verifying the installation procedure in accordance with Appendix C, the author attempted to verify the expected operation of Player and Gazebo using the packaged world files, configuration files, and models. Because the potential simulation targets required the author to implement a simulation of a challenge vehicle, the author first attempted to modify and use the packaged “simplecar” model. The author was unable to verify the expected operation of Gazebo due to several errors in the Gazebo source code, world files, and models. For various reasons, Gazebo would fail to load included world files, Player would not connect to Gazebo, and the `playerv` utility would not move the model. At one point while attempting to verify the expected operation of Gazebo, the use of Stage in lieu of Gazebo was evaluated because of problems encountered. The author concluded the use of Stage, which provides a “2.5-D” simulation environment, would not provide enough realism for simulation of a challenge vehicle.

The author spent several weeks modifying Gazebo world files and Player configuration files and reviewing source code to determine the cause of the problems encountered. While reviewing the code to determine the cause of the problems encountered, the author noted that the Gazebo code base is being actively developed, and that, for reasons unknown, some changes “break” Gazebo in unexpected ways, and that some revisions of the Gazebo source code include extensive debugging information.

Problems encountered by the author while attempting to verify Player and Gazebo using packaged world files, configuration files, and models are documented by Appendix G.

IV.D. Upgrade Player and Gazebo

Review of mailing list archives and resolutions to similar problems encountered by other users suggested by Gazebo's developers indicates that “upgrade to the latest svn version” is the general response given when bugs are encountered and ostensibly resolved. Therefore, while troubleshooting the errors encountered while attempting to verify Player and Gazebo using packaged world files, configuration files, and models, later versions of Player and Gazebo were downloaded and installed.

The author downloaded the source distribution of Player 3.0.1 (“`player-3.0.1.tar.gz`”) from the Player Project ([38]), un-installed Player version 3.0.0 in accordance with Appendix C, and installed Player version 3.0.1 in accordance with Appendix B.

The author uninstalled Gazebo version 8443 as described below, downloaded the latest revision (revision 8533) of the Gazebo 0.9.0 source code using the `svn` utility, and installed Gazebo revision 8533 in accordance with Appendix B.

As discussed in Appendix C, the author was unable to `make uninstall` or `make clean` Gazebo. As a result, when verifying the installation procedure the existing installation of Gazebo was archived by renaming the containing directory and manually deleting file `.gazeborc`. While researching the `cmake` utility, the author noted the `xargs` utility may be used to remove all files installed using the `cmake` utility ([39]) as follows:

```
xargs rm < install_manifest.txt
```

File `install_manifest.txt` provides a list of all files generated by the `cmake` utility during installation. This command was used to uninstall Gazebo.

When upgrading to Gazebo revision 8533, the author noted it was no longer necessary to modify file `audio.cc` in accordance with step "Install Gazebo" of Appendix B. File `audio.cc` had been revised to correct the error noted by the author.

IV.E. Configure the simulation for the selected simulation targets

After verifying Player and Gazebo using packaged world files, configuration files, and models and upgrading Gazebo, the simulation was configured for each selected simulation target. The author selected the following simulation targets:

- Simulation target 1: Use Player and Gazebo to evaluate the rollover of a representative challenge vehicle entering 2004 GCE course segment 2570-2571-2572.

- Simulation target 2: Use Player and Gazebo to evaluate the use of LIDAR, in particular the quality of the point map created by SICK LMS 200 and 291 LIDAR sensors, and increase in the number of SICK LMS 291 LIDAR sensors.
- Simulation target 3: Use Player and Gazebo to evaluate field-of-view limitations for selected sensors, specifically navigation RADAR.

Due to time constraints, and the difficulty "fitting" a braking profile to a simulated challenge vehicle, the author decided not to implement a two-material friction model.

Due to time constraints, the author decided not to modify Player and Gazebo to simulate sensor noise.

CHAPTER V. GENERAL CONFIGURATION PRACTICES AND PROCEDURES

V.A. Common practices

In addition to the tutorials and instructions available from online documentation ([38]), the author developed practices which proved to be useful while attempting to determine the causes of various errors encountered while configuring an arbitrary simulation:

V.A.1. Use valid "model_name::interface_name" addressing

For Player to communicate successfully with Gazebo, Player must know what interfaces Gazebo is providing. Determining the valid address for interfaces was more difficult than anticipated. However, valid values for "model_name::interface_name" addressing in Player configuration files may be determined by reviewing the available interfaces in directory "`/tmp/gazebo...`" corresponding to the user's running instance of Gazebo, which are defined by the world file. For example, loading the model used to evaluate rollover of a representative challenge vehicle entering 2004 GCE course segment 2570-2571-2572 (see Appendix F) creates a "`position.cv_model::position_iface_0`" interface in directory "`/tmp/gazebo...`". The corresponding Player configuration file "`gz_id`" for this interface is therefore "`cv_model::position_iface_0`", and this "`gz_id`" must be defined by the Player configuration file before launching Player for Player to communicate successfully with Gazebo.

V.A.2. Increase the controller update rate to increase the quality of logged data

A comment in file `playerv.c` (the `playerv` utility) states: "20 Hz update rate is good for user interaction". However, when the author began to log output generated by the improved steering controller ("improved controller") updates were being generated at 10 Hz.

File `Controller.cc` attempts to set parameter `updateRate` when a controller is loaded. Parameter `updateRate` is not in use by any packaged controllers, models, or world files. As a result, the author was unaware parameter `updateRate` could be declared until he reviewed the Gazebo codebase to identify parameters which could be declared. See paragraph V.A.3. below.

An `<updateRate>` declaration was included in the `<controller>` declaration to increase the update rate of the improved steering controller and the quality of logged data.

An update rate of 50 Hz gave good results for logging data, with few missed intervals, and no observable impact on the ratio of simulation time to real time. Increasing the update rate beyond 50 Hz did not result in a significant increase in the quality of logged data, and resulted in a greater number of missed intervals. Decreasing the update rate to 10 Hz resulted in instability when the model was traveling at high speed.

V.A.3. Review the Gazebo codebase to identify parameters which may be declared

The author was unaware parameter `updateRate` could be declared because it was not in use by any packaged controllers, models, or world files. Review of file `Controller.cc` identified two other parameters which may be declared: `name` and `alwaysOn`. Parameter `alwaysOn` was not in use by any packaged controllers, models, or world files. Parameter `name` was in common use.

The author reviewed the Gazebo codebase for occurrences of "`new ParamT`" in files to identify parameters which may be declared. Some files, for example, the packaged steering controller, use private class member variables in lieu of parameters.

V.B. Common procedures

The author developed common procedures, some of which were based on tutorials and instructions available from online documentation:

V.B.1. Use of the `ffmpeg` utility to create movies from captured images

The "Save Frames" command in Gazebo was used to capture images during simulation, then movies were created from the captured images using the following commands:

```
ffmpeg -f image2 -i UserCamera_0-%04d.jpg [destination]
ffmpeg -i UserCamera_0%04d.jpg [destination]
```

However, captured images were skewed to the right. See Figure 2. Although it was possible to create movies from the captured images, the resulting movies were also skewed to the right, making it difficult to effectively visualize the simulation. After several attempts, the author abandoned the use of the `ffmpeg` utility to create movies from captured images.

KSnapshot, a screenshot utility packaged with the K Desktop Environment (KDE), was used to capture images during simulation, and these images are the images included herein.

V.B.2. Patch generation

The following command was used to prepare patches submitted as a result of this research:

```
diff -rup /path/to/unmodified/source /path/to/modified/source
```

V.C. Model creation

Models of a representative challenge vehicle and obstacles DARPA identified as typical of obstacles challenge vehicles would encounter during the 2004 GCE were developed during this research.

V.C.1. Representative challenge vehicle

To maximize the re-usability of the model, the author selected a representative challenge vehicle which was:

- Successful.

Teams 2005-06, 2005-13, 2005-14, and 2005-16 successfully completed the 2005 GCE.

- A commercially-available SUV.

A commercially-available SUV was the most common platform selected by teams which participated in the 2004 or 2005 GCE. Commercially-available SUVs were in use by Teams 2005-06, 2005-14, and 2005-16.

- Described in sufficient technical detail in published records to model in simulation.

Neither challenge vehicle SSF nor height of vehicle CG were reported by published records for Team 2005-14 or 2005-16 challenge vehicles.

As a result, the Team 2005-06 challenge vehicle was selected as representative. A model was created using five bodies (“chassis_body”, “left_front_wheel”, “right_front_wheel”, “left_rear_wheel”, and “right_rear_wheel”) and associated geoms with the physical dimensions and other characteristics of the representative challenge vehicle. The representative challenge vehicle model is described in detail in Appendix F.

Because the selected simulation targets included an evaluation of the rollover condition, realistic physical dimensions and other characteristics of the model were selected to ensure the track width, height of vehicle CG, and curb weight in simulation were identical to those of the representative challenge vehicle.

By default, Gazebo places the CG of a body at its center. The author did not alter the default behavior. Realistically, the rollover condition is dependent on the location of vehicle CG, which may not be at the geometric center of the model's “chassis_body”. However, the author considers it likely the representative challenge vehicle CG was very close to the left-right centerline of the vehicle, although he acknowledges it may have been forward of the front-back centerline of the vehicle due to the weight of the engine.

The distance of the CG from the front-back centerline of the vehicle may affect vehicle dynamics, including rollover, but the effect will be much less than that of the distance of the CG from the left-right centerline due to the difference between the wheelbase and track width dimensions. For the representative challenge vehicle, the distance between front and rear axles (wheelbase) was 1.7 times the track width. The author is confident the contribution to vehicle dynamics, including rollover, of the distance between the CG and left-right centerline of the representative challenge vehicle is greater than the contribution of the distance between the CG and front-back centerline of the vehicle, and considers the model to be accurate enough to evaluate the selected simulation targets.

A mesh was created to provide the model with a visual similarity to the representative challenge vehicle. See Figures 3 and 4 for a visual comparison of the representative challenge vehicle to the model. Packaged meshes were used for the wheels of the model.

V.C.2. Representative obstacles

DARPA published a description of obstacles teams participating in the 2004 GCE could expect to encounter during the 2004 QID and which were representative of obstacles teams could expect to encounter during the 2004 GCE: “Dirt Hills”, “Tower Obstacle”, “Car Obstacle”, “Steep Hill”, “Sand Trap”, “Ditch”, “Cattle Guard”, “Overpass”, “Boulders”, “Moving Car Obstacle”, and “Washboard” ([10]).

To effectively evaluate the simulation targets, two obstacles were selected as representative: “Tower Obstacle” and “Car Obstacle”. The obstacles were modeled using the Player Project Model Creation Tutorial ([40]). The “Car Obstacle” model was based on the dimensions and weight of a 2009 Honda Accord. Meshes were created to provide the models with a visual similarity to the representative obstacles. Unlike the representative challenge vehicle, the representative obstacles were modeled using a “trimesh” geom primitive[11] to provide the most accurate interaction with sensors possible[12].

See Figures 5, 6, 7, and 8 for a visual comparison of the representative obstacles to the models created by the author.

V.D. Mesh creation

The representative challenge vehicle, representative obstacles, and guides used to visually evaluate the interaction of the representative challenge vehicle model with the environment required the creation of meshes having arbitrary shapes. As a result, the author created several custom meshes during this research. The author found the Player Project Mesh Creation Tutorial ([41]) to be a useful starting point when creating meshes, but it would have required the author to learn to use Blender or another 3D rendering application with which the author had no familiarity. However, the author determined Blender ([42]) could be used as an intermediate application.

The author installed `blender-2.49a-4.5` using YaST. Packages `openal-soft 1.9.616-1.1.1` and `libopenal1-soft 1.9.616-1.1.1` were installed by YaST to resolve dependencies. The author then installed the Blender Exporter ([43]).

The author created models using TurboCAD Mac Deluxe, an application with which the author had some familiarity, exported them, and imported them into Blender. To determine which file format provided the best compatibility, the author attempted to import several different file formats exported from TurboCAD Mac Deluxe (DXF, DWG, AI, RAW, WRL, and STL) into Blender, with varying results:

- Several files caused Python script errors.
- Attempting to import a DWG file caused a “DWG-Importer cant find external DWG-converter (DConvertCon.exe) in Blender script directory” error

("DConvertCon.exe" is a Windows executable).

- WRL files were imported "one-sided".
- Attempting to import AI files resulted in a "Not a valid file or an empty file" error.

The Autocad file format (DXF) had the best compatibility. As a result, all models created by the author were created using TurboCAD Mac Deluxe, exported as Autocad Revision 12 (R12) files, and imported into Blender using the DXF importer.

Because Blender was installed on a desktop computer running openSUSE 11.2, the author had to specify "Unix (CR)" line-end characters when exporting models from TurboCAD Mac Deluxe. All models created using TurboCAD Mac Deluxe were created using metric units.

The author experienced unexpected behavior when importing DXF files into Blender because the "origin point" in Blender does not necessarily correspond to the origin in TurboCAD Mac Deluxe. The Player Project Mesh Creation Tutorial states: "Move the mesh to the origin." Although this is straightforward, importing an Autocad file into Blender causes the origin point to shift, even though the apparent origin of the model does not appear to have changed from the intersection of the x-, y-, and z-axes. The author used Blender's "Center" or "Center Cursor" functions to align the origin point with the apparent origin of the model, resolving the problem.

The models were then exported as meshes using the OGRE Mesh Exporter. When exporting the meshes using the OGRE Mesh Exporter, the author disabled options

“Export Materials”, “Fix Up Axis to Y”, or “Require Materials”, enabled option “OgreXMLConverter”, and clicked “Export”.

The resulting OGRE mesh files were then copied to the `/gazebo/Media/models` directory or `models` subdirectory of one of three test directories for use.

CHAPTER VI. IMPROVED CONTROLLER IMPLEMENTATION

To properly simulate the handling of a four-wheeled vehicle, the author performed an analysis of Ackermann steering geometry, and improved a packaged steering controller to more closely conform to Ackermann steering geometry.

The author first attempted to use the packaged steering controller. Because the ability to turn at a constant radius at speeds typical of vehicles participating in the 2004 and 2005 GCE is central to validating several of the conclusions reached throughout this research, to demonstrate the ability to limit the turning radius of the model to the radius of the representative challenge vehicle turning circle (37.7 ft or 11.49 m), the author generated a mesh of two circular walls with a height and width of 10 cm: the inner wall with a outer radius of 9.49 m and the outer wall with an inner radius of 13.49 m. A world file was generated to include this mesh and the representative challenge vehicle model located in the turning circle with its CG at a position offset by CG to rear axle x-dimension (1.265 m), and one-half the sum of the rear track width and section width (0.882 m).

When controlling the model with the `playerv` utility using the packaged controller, the author noted an unexpected "flattening" of the path of travel when the steering angle was at a maximum at full right extent. The steering angle was selected based on the representative challenge vehicle turning circle.

Problems encountered during development of the improved controller are documented by Appendix G.

VI.A. Independent steering wheel angle

The author analyzed the packaged steering controller and determined that setting the angle of both steering wheels to the same angle had the effect of "dragging" the rear of the model around at certain points on the path of travel, and proposed the observed behavior was due to the effect of friction caused by the angle of the outer steering wheel. Because the packaged steering controller set the angle of both steering wheels to the same angle, the outer steering wheel in any turn was exerting more force toward the center of the turning circle than it should have been exerting.

After a period of time during which the model would travel in a circular path as expected, the effect of friction would cause the front wheels to drag the front end of the model around using the inner rear wheel as a pivot. The motion of the model would return to normal for a while, then the effect of friction would cause the front wheels to drag the front end of the model around using the inner rear wheel again. This would continue as long as the model was traveling in a circle.

The author improved the steering controller to determine and set the angle of the steering wheels independently. This conforms to Ackermann steering geometry. The steering angles and angular velocities of the inside and outside wheels are calculated using the steering angle and angular velocity at model CG, track width, and wheelbase. As a result, when the steering angle is not equal to zero:

- The steering angle of the inside wheel (i.e., the wheel on the inside of the turn) is greater than the steering angle at model CG.
- The steering angle of the outside wheel is less than the steering angle at model

CG.

VI.B. Odometry

Because several of the simulation targets require the ability to determine the position of the model in relation to detected obstacles, the author implemented odometry in the improved controller in a manner similar to the implementation in the packaged position controller for a robot using differential drive.

The position of the model is determined based on the distance traveled by the rear wheels, which is calculated using the angular velocity and radius of the rear wheels. One weakness of this method is that the reported position of the model is independent of the actual position of the model when the wheels are not touching the ground because the controller continues to calculate the distance traveled by the rear wheels.

However, if this is a problem in simulation it may also have been a problem during the 2004 and 2005 GCE. Thirteen of 25 teams in 2004 and seven of 23 teams in 2005 referred to the use of encoders as navigation sensors on each wheel, rear wheels only, or the drive axle to provide instantaneous velocity.

In addition, the error in reported position of the model increases as the distance traveled from the initial position of the model increases due to the accumulation of errors in calculation over thousands of simulation cycles. Because most of the tests performed by the author were over relatively short distances, the author did not attempt to resolve this problem.

The author attempted to use the reported position to determine the exact location at which rollover occurred to be able to analyze the path of the model until the onset of rollover. However, attempts to determine the onset of rollover were unsuccessful. See paragraph VII.E.

VI.C. Additional features of the improved controller

In addition to the parameters described above, the author also implemented several features to increase the usability and realism of the improved controller:

- Gas pedal

The author implemented a "gas pedal" by using the `cmdVelocity.pos.x` value returned by improved controller function `GetPositionCmd` to scale the maximum constant acceleration calculated by the controller. The maximum constant acceleration was determined by values for the final velocity and time to reach the final velocity. These values are read from the model XML file.

The `cmdVelocity.pos.x` value returned by function `GetPositionCmd` is between -0.1 and 0.5, depending on how far the user drags the cursor to the left or right, respectively. The minimum and maximum values were compiled into the `playerv` utility and may be modified by making changes to function `position2d_servo_vel` in file `pv_dev_position2d.c`. To minimize the number of changes required, the author decided not to revise the `playerv` utility to modify the minimum and maximum values but to use these values to scale acceleration via the improved controller.

To simulate a gas pedal, the author calculated forward velocity using a scaled acceleration equal to constant maximum acceleration multiplied by a factor of `cmdVelocity.pos.x/0.5`, and reverse velocity using a scaled acceleration equal to constant maximum acceleration multiplied by a factor of `cmdVelocity.pos.x/0.1`.

As a result, dragging the cursor farther to the left or right from center is analogous to depressing the gas pedal harder while in "Reverse" or "Drive", respectively.

- Brake pedal

The author implemented a "brake pedal" by reducing the velocity by three times the scaled acceleration calculated above when velocity was greater than zero. In practice, this reduction in velocity would have to be fitted to the braking profile of the simulated challenge vehicle.

- Elimination of redundant classes

The packaged controller used three classes to represent wheels: a base class ("Wheel"), and two derived classes ("DriveWheel" and "FullWheel"). The member variables and functions for these classes were very similar. The author collapsed the three classes into a single class ("Wheel") by defining an additional member variable: `type`, which is assigned when the wheels are created by reading the type from the model XML file. Three types are supported: `DRIVE`, `STEER`, and `FULL`.

- Revise the steering controller to increase use of parameters

The packaged controller used non-parameter private class member variables. Most Gazebo classes use parameters in lieu of private class member variables. The author revised the steering controller to make increased use of parameters for values used by the controller to more closely conform to other Gazebo classes, but did not eliminate the use of private class member variables to store values calculated from the parameters in use.

VI.D. Parameters in use by the improved controller

In addition to the characteristics required to accurately model the representative challenge vehicle, the author implemented the following parameters for the purposes of evaluating the simulation targets:

- `useSwaybars`

When parameter `useSwaybars` is TRUE, the improved controller will attempt to compensate for “up” and “down” forces in each joint in a manner similar to anti-sway bars by applying a counter force to each joint. The counter force is calculated using parameters `swayForce` and `swayForceLimit`. When FALSE, the controller does not attempt to compensate.

- `swayForce`

When parameter `useSwaybars` is TRUE, parameter `swayForce` defines the force used to determine the moment applied in each joint due to displacement of the joint. The moment is the product of the force and the displacement.

- `swayForceLimit`

When parameter `useSwaybars` is TRUE, parameter `swayForceLimit` defines the maximum moment used to compensate for “up” and “down” forces in each joint. Forces which would result in moment greater than the sway force limit are not applied to the joint.

- `useConstantVelocityMode`

The improved controller reads velocity commands from the `playerv` utility. When parameter `useConstantVelocityMode` is TRUE, the controller will maintain a constant velocity. When FALSE, the controller causes the model to coast to a stop.

- `useConstantSteeringAngleMode`

The improved controller reads steering commands from the `playerv` utility. When parameter `useConstantSteeringAngleMode` is TRUE, the controller will use parameter `constantSteeringAngle` to override the steering angle sent from the `playerv` utility. Steering angle at model CG will equal parameter `constantSteeringAngle`. When FALSE, the controller accepts steering commands sent from the `playerv` utility.

- `constantSteeringAngle`

When parameter `useConstantSteeringAngle` is TRUE, parameter `constantSteeringAngle` defines the steering angle at model CG and overrides steering commands from the `playerv` utility.

- `useSafeVelocity`

When parameter `useSafeVelocity` is TRUE, limits the maximum velocity at model CG to the maximum allowed by representative challenge vehicle and course geometry. When FALSE, the maximum velocity at model CG is equal to the final velocity. The final velocity is set using the `<velocityFinal>` declaration and, with the `<velocityFinalTime>` declaration, is used to calculate acceleration for the model, which is a constant.

- `velocityOffset`

When parameter `useSafeVelocity` is TRUE, parameter `velocityOffset` defines the amount by which to increase the calculated maximum velocity at model CG.

- `useTurnRadius`

When parameter `useTurnRadius` is TRUE, the calculation of the maximum steering angle, and maximum velocity and angular velocity at model CG, is based on parameter `turnRadius`. When FALSE, the calculation is based on representative challenge vehicle geometry and characteristics, specifically turning circle, track width, and section width.

- `turnRadius`

When parameter `useTurnRadius` is TRUE, parameter `turnRadius` defines the turn radius used to calculate the maximum steering angle, and maximum velocity and angular velocity at model CG.

VI.E. Validation of the improved controller

Output from the steering controller was logged to confirm the values calculated conformed to Ackermann steering geometry and other design decisions implemented by the author.

The author used parameters `useConstantSteeringAngleMode` and `useSafeVelocity` to limit the motion of the model to travel in a circle based on the representative challenge vehicle turning circle of 11.491 m at a constant steering angle of -0.376337 radians (i.e., a right turn at a constant steering angle of 21.56 degrees), launched Gazebo, and let the simulation run for 60 s. The author validated the controller by confirming:

- The radius used to calculate maximum velocity, maximum angular velocity, and maximum steering angle at model CG ("calculated radius") was 6.63 m, corresponding to one-half the sum of the model's turning circle, rear track width, and section width.

- The maximum velocity of the model was 8.72 m/s, corresponding to the calculated radius of 6.63 m and SSF of 1.17.
- The maximum angular velocity at model CG was 1.316 radians/s, corresponding to a maximum velocity of 8.72 m/s and circumference of 41.6 m.
- The maximum steering angle at model CG was 0.376337 radians, corresponding to a wheelbase of 2.619 m and calculated radius of 6.63 m.

The model completed twelve complete rotations in 60 s, with an average velocity of 8.71 m/s. Because the model accelerated from a complete stop, the average velocity was less than the maximum velocity at model CG. At a controller update rate of 50 Hz, the radius of the turning circle of the model ("reported radius") was 6.52 m. The error in the reported radius was -0.11 m (approximately 1.7 percent of the calculated radius). The author proposes this error may be due to the centripetal force applied by the steering wheels toward the center of the turning circle. No significant eccentricity was noted.

The author concluded the improved controller used representative challenge vehicle and course geometry, such as turning circle and SSF, to correctly calculate internal variables used to limit the maximum velocity and steering angle at left or right extent.

For example, to evaluate the rollover condition, in particular 2004 GCE course segment 2570-2571-2572, it was necessary to verify the improved controller calculated maximum angular velocity at model CG correctly using SSF and turning circle. Using a SSF of 1.17 and turning circle of 11.491 m, the author confirmed the controller calculated the correct maximum angular velocity of 22.0632 rad/s for all wheels when steering angle was equal to zero, corresponding to a velocity of 18.2 mph, which is consistent with the Team 2005-06 challenge vehicle.

Problems encountered while validating the improved controller are documented in Appendix G.

The author then used parameter `velocityOffset` to determine how much the maximum velocity could be increased before rollover occurred. Table I summarizes the results:

Table I. Approximate simulation time at which rollover occurs at velocity offset.

Velocity offset (m/s)	Simulation time (s)
0.10	-
0.20	-
0.22	-
0.24	-
0.25	48
0.26	19
0.27	14
0.28	12
0.29	10
0.30	9
0.31	8
0.32	6
0.33	5
0.34	4
0.35	3
0.36	2

With a velocity offset of 0.36 m/s, rollover occurred almost immediately after the model accelerated to maximum velocity, so the author discontinued the trials, concluding that increasing the velocity offset would not result in a reduction in the simulation time at which rollover occurs since constant acceleration was calculated based on parameters in use by the controller.

The simulation was then allowed to run for 300 s simulation time with a velocity offset of 0.24 m/s with no rollover event.

The information summarized by Table I is visually represented by Figure 1:

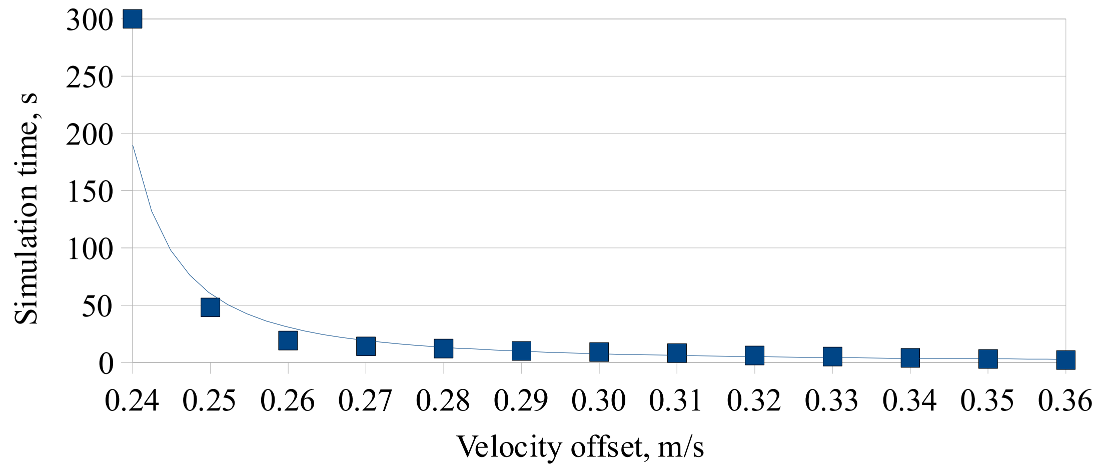

Figure 1. Approximate simulation time at which rollover occurs versus velocity offset.

The relationship between the simulation time at which rollover occurs and velocity offset suggests a power regression, but the data collected by the author is approximate. The author used the `playerv` utility to control the velocity of the model. As a result, there was a delay between starting the simulation and starting to accelerate the model of approximately 1.5 seconds.

Based on the results, the author concluded merely exceeding the maximum safe velocity may not result in rollover. The actual onset of rollover may be delayed, provided the model recovers before rollover occurs. As a result, the author proposed that the model could safely exceed the maximum velocity at model CG without rollover for a limited time without adverse consequences, and that the time by which the model could safely exceed the maximum velocity at model CG without rollover would decrease as the velocity offset increased.

The author calculated the effective SSF of the model using the maximum velocity at which the model was able to successfully complete the test (9.08 m/s). The effective SSF of the model was 1.27 corresponding to a turn radius of 6.63 m, which well exceeds the representative challenge vehicle SSF of 1.17.

In general, suspension and tire effects contribute to a reduction of up to ten percent in SSF, meaning the increase over the effective SSF of the representative challenge vehicle of 1.07 was approximately 20 percent.

Because one of the purposes of high-fidelity simulation is to model real-world interaction which it may not be possible to evaluate in the real world, the author is not confident this is not a valid result:

- The model is an unrealistically rigid body.

A rigid-body model cannot predict time-dependent details of rollover such as those observed above.

- The accumulation of error in joints over time may result in bodies drifting away from their expected positions.

The ODE Manual states: “There is a mechanism to reduce joint error: during each simulation step each joint applies a special force to bring its bodies back into correct alignment. This force is controlled by the *error reduction parameter* (ERP), which has a value between 0 and 1.” and “The ERP specifies what proportion of the joint error will be fixed during the next simulation step." ([44]).

Setting the ERP equal to 1.0 was not recommended. Setting the ERP to a value between 0.1 and 0.8 was recommended (0.2 is the default). An ERP of 0.8, the maximum recommended, was selected. This value conforms to the ERP in use in several packaged world files and models, and specifically world file "simplecar.world" which was used to evaluate the packaged steering controller.

- The representative challenge vehicle model does not model suspension and tire effects.

ODE provides a mechanism to control the "springyness" of joints, and which could be used to model challenge vehicle suspension and tire effects: Constraint Force Mixing (CFM). Setting the CFM to be less than zero was not recommended. The ODE Manual states: "If CFM is set to zero, the constraint will be hard. If CFM is set to a positive value, it will be possible to violate the constraint by 'pushing on it' (for example, for contact constraints by forcing the two contacting objects together). In other words the constraint will be soft, and the softness will increase as CFM increases." ([44]).

A CFM of 0.00001 (10^{-5}) was selected. This value conforms to the CFM in use in several packaged world files and models, and specifically world file "simplecar.world" which was used to evaluate the packaged steering controller.

- An effective SSF of 1.41 was calculated during evaluation of 2004 GCE course segment 2570-2571-2572.

This value exceeds the effective SSF calculated during validation of the improved controller. The author concluded the effective SSF of the model was independent of model geometry, but may be a function of force due to lateral acceleration:

$$F = ma = \frac{mv^2}{r}\text{, therefore } a = \frac{v^2}{r}$$

$$\text{The rollover condition is: } \frac{v^2}{rg} > SSF$$

Force due to lateral acceleration, *a*, is balanced by centripetal force (the sum of forces exerted by the tires toward the center of the turning circle) at the tire-surface interface. ODE calculates tangential forces at the tire-surface interface ("contact point") using a "friction pyramid" approximation of the Coulomb friction model.

As a result, the author proposed selection of ERP and CFM in combination with friction approximation may result in an effective SSF of the model which exceeds the effective SSF of the representative challenge vehicle, and concluded additional testing in simulation and using the representative challenge vehicle would be required to effectively "tune" the model.

The model was not revised to have an SSF of 1.17 or an effective SSF of 1.07 because it would have required altering the geometry of the model, which was based on the representative challenge vehicle.

CHAPTER VII. EVALUATION OF 2004 GCE COURSE SEGMENT 2570-2571-2572

VII.A. Configuration of the simulation environment

Two meshes representative of 2004 GCE course segment 2570-2571-2572 were created using TurboCAD Mac Deluxe in accordance with paragraph V.D. Both the first and second mesh were dimensionally accurate, having the length of adjacent segments 2570-2571 (137.1 m) and 2571-2572 (143.2 m), with an angle of -47.865 degrees between them (i.e., a 47.865 degree-turn to the right). Both meshes represented the course boundaries as walls ten cm high and ten cm wide at a distance equal to the lateral boundary offset (3.962 m) from the centerline of the course. However, the radius of the outer wall at the intersection was different for each mesh.

DARPA stated ([1]):

```
The Lateral Boundary Offset (specified in feet) is the
distance in any direction from the Track Line
(including a radius at the end points) that defines
the corridor in which Challenge Vehicles are permitted
to travel.
```

DARPA's instructions were not as specific during the 2004 GCE as they were during the 2005 GCE. As a result, the author does not know what DARPA intended by “including a radius at the end points”. However, DARPA provided specific examples of course geometry prior to the 2005 GCE ([2]).

As a result, the first mesh included an outer radius of 7.924 m (twice the lateral boundary offset) tangent to the outer wall at the intersection of the two adjacent course segments. The adjacent course segments were otherwise straight. The first mesh was representative of the actual course boundaries established by DARPA in 2005.

The second mesh included an outer radius of 46.1 m tangent to the outer wall at the intersection of the two adjacent course segments. This was not representative of the 2004 GCE course, but provided a way to visually evaluate how far from the centerline during a turn of constant radius of 46.1 m the model would travel.

A world file was generated to contain the model and either of the two meshes.

VII.B. Simulation procedure

The following procedure was developed to evaluate the simulation target:

1. Set the following parameters:

   ```
   useSwaybars = FALSE
   useConstantVelocityMode = TRUE
   useConstantSteeringAngle = FALSE
   useSafeVelocity = FALSE
   useTurnRadius = TRUE
   turnRadius = 46.1
   ```

2. Run the simulation. Accelerate the model to maximum velocity and attempt to negotiate the turn successfully.
3. If rollover occurs, set parameter `useSafeVelocity = TRUE` and confirm the model is able to make the turn at the maximum safe velocity.
4. If the model is able to make the turn at the maximum safe velocity, adjust the maximum velocity at model CG by increasing the velocity offset by 0.5 m/s until rollover occurs.
5. When the model is no longer able to make the turn at the adjusted

maximum velocity, revise the world file to relocate the model to maximize the potential radius of the turn.

6. Report the results.

With parameter `useSafeVelocity` set to `FALSE`, the maximum velocity at model CG was 26.822 m/s. The corresponding maximum angular velocity at model CG was 0.494 radians/s and maximum steering angle at model CG was 0.0557 radians.

With parameter `useSafeVelocity` set to `TRUE`, the maximum velocity at model CG was 23.218 m/s. The corresponding maximum angular velocity at model CG and maximum steering angle at model CG were unchanged. These values are determined by representative challenge vehicle and course geometry.

The author then started Gazebo, started Player, started the `playerv` utility, and observed the model as it attempted to successfully negotiate turn 2570-2571-2572 using the first mesh generated. The second mesh generated was used to determine how closely the model followed the centerline of a radius 46.1 m curve, with good results.

VII.C. Results

Multiple runs of nine trials in total were completed. The results are summarized below.

- Trial 1

The model was accelerated to the maximum speed allowed by the RDDF of 60.0 mph (26.822 m/s). When the steering wheel was not released during the turn to allow the model to recover, the model rolled over. When the steering wheel was released during the turn to allow the model to recover, the model exceeded the outer lateral boundary offset and left the course.

- Trial 2

  Parameter `useSafeVelocity` was set to `TRUE`. The model successfully completed the turn at the maximum safe velocity of 23.218 m/s without rolling over or leaving the course.

- Trials 3 through 5

  The maximum velocity at model CG was changed by increasing the velocity offset by 0.5 m/s in each trial. The model successfully completed the turn at velocities of 23.718 m/s (+0.5 m/s), 24.218 m/s (+1.0 m/s), and 24.718 m/s (+1.5 m/s).

- Trial 6

  The maximum velocity at model CG was changed by increasing the velocity offset by 0.5 m/s to 2.0 m/s. The model successfully completed the turn at a velocity of 25.218 m/s, but began to tip during the turn.

- Trial 7

  The maximum velocity at model CG was changed to 25.718 m/s by increasing the velocity offset by 0.5 m/s to 2.5 m/s. When the steering wheel was not released during the turn to allow the model to recover, the model rolled over. When the steering wheel was released during the turn to allow the model to recover, the model exceeded the outer lateral boundary offset and left the course.

- Trial 8

  The maximum velocity at model CG was not changed. The world file was revised to relocate model CG 3.072 m to the left of its original position. This is the maximum lateral displacement possible without violating the outer lateral boundary offset in segment 2570-2571-2572, increasing the potential radius to 49.172 m. The model rolled over during the turn before the steering wheel was released to allow the model to recover.

- Trial 9

The model XML file was revised to increase the radius used to calculate maximum velocity, maximum angular velocity, and maximum steering angle to 49.172 m, corresponding to the maximum lateral displacement of the model. As a result, the maximum velocity increased to 26.465 m/s, the maximum angular velocity decreased to 0.479 radians/s, and the maximum steering angle at model CG decreased to 0.523 radians. The model rolled over during the turn before the steering wheel was released to allow the model to recover.

VII.D. Conclusions

The author compared the results of the model entering segment 2570-2571-2572 with an allowed speed exceeding 48.0 mph to the results after configuring the controller to limit speed to 48.0 mph. A speed of 48.0 mph corresponds to the course segment 2570-2571-2572 maximum allowed turn radius of 46.1 m for a challenge vehicle with a SSF of 1.02, the worst case scenario. The maximum speed allowed by the RDDF of 60.0 mph corresponded to a turn radius of 72 m. Because the maximum allowed turn radius exceeded the turn radius corresponding to the maximum speed allowed by the RDDF, this turn represented a rollover risk if an arbitrary challenge vehicle entered this turn at a speed exceeding 48.0 mph.

The author concluded the representative challenge vehicle would not have been able to successfully complete the turn in segment 2570-2571-2572 at the maximum speed allowed by the RDDF and would have either rolled over or left the course. However, the author calculated the effective SSF of the model using the velocity at which the model was able to successfully complete the turn (25.218 m/s). An effective SSF of 1.41 was calculated using a turn radius of 46.1 m, which well exceeds the representative challenge vehicle SSF of 1.17 or effective SSF of 1.07. The author proposed selection of ERP and CFM in combination with friction approximation may result in an effective SSF of the model which exceeds the effective SSF of the representative challenge vehicle. See paragraph VI.E.

VII.E. Determination of the onset of rollover

To more reliably determine the onset of rollover, the author attempted to implement a rollover flag using functions `ODEBody::GetEulerRate`, `ODEBody::GetLinearVel`, and `ODEBody::GetAngularVel` without success. Although these functions returned information which may be interpreted as rollover, the author was unable to distinguish the onset of rollover from motion prior to or after the onset of rollover.

The author then attempted to use function `ODEJoint::GetFeedback` to return an ODE `dJointFeedback` structure containing the values of forces applied to each body of the wheel joints to determine when the forces for the inside wheels were zero. Use of this function caused a segmentation fault. The author did not attempt to resolve the problem.

CHAPTER VIII. EVALUATION OF THE USE OF LIDAR

VIII.A. Configuration of the simulation environment

A world file was generated containing the first mesh created during evaluation of 2004 GCE course segment 2570-2571-2572, which included an outer radius of 7.924 m (twice the lateral boundary offset) tangent to the outer wall at the intersection of the two adjacent course segments. The first mesh was representative of the actual course boundaries established by DARPA in 2005.

A tower obstacle was located so the tower was at the center of a circle with radius 2.0 m tangent to the inner wall at the intersection of 2004 GCE course segments 2570-2571 and 2571-2572. This is representative of towers encountered during the 2005 GCE (see Figures 9 and 10 for examples). The tower model file was revised to include a `<laserFiducialId>` declaration of "1" and `<laserRetro>` declaration of "0.5".

The world file was revised to attach a SICK LMS 200 model to the representative challenge vehicle model. The world file was revised to relocate the SICK LMS 200 model through the trials which followed, and the SICK LMS 200 model file was revised to adjust the field-of-view and angular resolution through the trials which followed, as documented below.

The Player configuration file was revised to include a driver for the laser device and to add the laser device to the `writelog` driver used to log output.

VIII.A.1. Selection of scanning frequency

The scanning frequency of SICK LMS 200 or 291 LIDAR sensors is 37.5 Hz with an angular resolution of 0.5º and 75 Hz with an angular resolution of 1º. As documented below, angular resolutions of 0.5º and 1º were used when evaluating the revised simulation target. However, the author was unable to determine the scanning frequency in use by teams participating in the 2004 or 2005 GCE. Team 2005-18 stated: "[SICK LMS-221-30206] LADARs have a maximum range of 80 meters and a scanning rate of 75 Hz." ([26], p. 10). Three other teams referred indirectly to a SICK LIDAR sensor scanning frequency of 75 Hz: Teams 2005-05, 2005-08, and 2005-19.

However, an informal review of technical proposals for teams participating in the 2007 Urban Challenge indicate a scanning frequency of 10 Hz was not uncommon. As a result, the author concluded, although the maximum scanning frequency of a LIDAR sensor was 75 Hz, it was likely that LIDAR sensors in use by teams participating in the 2004 and 2005 GCE were operated at a reduced scanning frequency, and that a scanning frequency of 20 Hz was reasonable, and revised the `<updateRate>` declaration of the representative challenge vehicle model XML file steering controller and SICK LMS 200 model file laser controller to change the update rate of both controllers to 20 Hz. An update rate of 10 Hz was not considered due to problems encountered with simulation fidelity when validating the steering controller.

VIII.A.2. Selection of parameters `rayCount` and `rangeCount`

As described via paragraph V.A., the author reviewed files `Sensor.cc`, `Controller.cc`, `RaySensor.cc`, and `MultiRayShape.cc` (and corresponding header files) to determine valid parameters used by the simulated SICK LMS 200 LIDAR sensor. Although the author was able to generate a list of potential parameters, he was unable to determine the effect of some parameters, specifically `rayCount` and `rangeCount`, without review of documentation for a much earlier version of Gazebo (version 0.5).

However, based on a review of documentation for Gazebo 0.5, parameters `rayCount` and `rangeCount` were set equal to the same value through the trials that followed.

VIII.A.3. Selection of parameter `maxRange`

Parameter `maxRange` was set to “30”, the typical range with ten percent reflectivity for SICK LMS 291 LIDAR sensors.

VIII.B. Revision of the simulation target

The original simulation target was: “Use Player and Gazebo to evaluate the use of LIDAR, in particular the quality of the point map created by SICK LMS 200 and 291 LIDAR sensors, and increase in the number of SICK LMS 291 LIDAR sensors”. See paragraph IV.E.

However, while configuring the simulation the author determined that there would be no difference between the quality of the point map generated by SICK LMS 200 and 291 LIDAR sensors in simulation based on the results of several trial runs. Both sensors have a maximum possible scanning angle of 180º and angular resolution of 0.25º, 0.5º, or 1.0º, and both sensors have identical response times and scanning frequencies ([45]).

Gazebo uses a generic “ray” sensor to simulate a LIDAR sensor. As a result, the point map generated by SICK LMS 200 LIDAR sensors would be virtually identical to the point map generated by SICK LMS 291 LIDAR sensors with the same parameters in simulation.

SICK LMS 200 LIDAR sensors have a typical range with ten percent reflectivity of 10 m and SICK LMS 291 LIDAR sensors have a typical range with ten percent reflectivity of 30 m ([45]). The author determined the difference in the quality of the point map generated by SICK LMS 200 or 291 LIDAR sensors at ranges typical of the two sensors in simulation was self-evident from review of logged data based on the results of several trial runs.

SICK LMS 200 LIDAR sensors do not have one feature SICK LMS 291 LIDAR sensors have: fog correction ([45]). Gazebo makes use of OGRE. As a result, a Gazebo world file may be configured to include fog through use of the `<rendering>` declaration. However, the author did not have data with which to correlate the accuracy of the range returns from a simulated SICK LMS 200 or 291 LIDAR sensor through fog. Although SICK provided reflectivity in fog data ([46]), the author was unable to correlate it with intensity data returned by Gazebo, which was one of two values "0" and "1", with a value of "0" being typical.

Team 2005-06 stated: "Rather than pointing the LADAR devices at the ground horizontally, we mounted the LADAR devices vertically. We chose to align them vertically because it made obstacle detection much easier. In the simplest case, by analyzing the measurement data beam by beam in angular order, obstacles were easy to locate as either clusters of similar distance or gaps in distance." ([47], p. 513). No other environment sensors were in use by Team 2005-06. However, Team 2005-06 successfully completed the 2005 GCE.

This was atypical. Several other teams stated vertically-aligned LIDAR sensors were in use as terrain analysis or ground profile estimation sensors, but no other team relied on vertically-aligned LIDAR sensors as the only obstacle and path detection sensors. As a result, the simulation target was revised: "Use Player and Gazebo to evaluate the use of a single SICK LMS 291 LIDAR sensor in various configurations in simulation to determine if the vertical LIDAR configuration in use by Team 2005-06 provided a competitive advantage over the horizontal LIDAR configurations in use by the majority of teams, and if an alternate configuration combining aspects of a horizontal and vertical configurations would be more effective".

VIII.C. Simulation procedure

The author then started Gazebo, started Player, started the `playerv` utility, accelerated the model past the tower obstacle, and analyzed log output to evaluate the quality of the point map generated for various LIDAR configurations. Specifically, the author counted the number of range returns, and recorded the maximum range when the obstacle was first detected and minimum range when the obstacle was last detected.

VIII.D. Results

Three runs of the first trial were completed. The author determined the results from each run were virtually identical, with almost no variation (typically less than the range resolution of the simulated SICK LMS 291 LIDAR sensor) in range reported from one run to the next and no variation in the number of range returns. The author concluded it was unnecessary to complete multiple runs for each trial.

Six trials in total were completed. The results are summarized in Table II below. As an objective measure of the quality of each configuration, the ratio of the number of returns to the number of rays (“Quality”) was calculated.

- Trial 1

The SICK LMS 291 LIDAR model was located at the front left corner of the roof of the representative challenge vehicle model with a rotation of -90° around the x-axis and -20° around the z-axis so the beam of the sensor swept a vertical plane at an angle of 20° clockwise across the path of travel of the vehicle. See Figure 11.

The location of the SICK LMS 291 LIDAR model was selected so the beam of the sensor crossed the path of travel of the representative challenge vehicle model to ensure the model would have the ability to detect obstacles directly in front of the model.

An angle of -20° was selected based on visual analysis which indicated obstacles near the inner lateral boundary offset were within the 30-m detection range. Increasing this angle by three degrees to -17° resulted in an inability to detect obstacles within the course boundaries. Decreasing this angle by three degrees to -23° reduced the ability of the sensor to detect obstacles at the maximum range possible. An angle of -20° was selected to ensure obstacle detection at a range slightly exceeding the inner lateral boundary offset.

Parameter `minAngle` was set to “-10”, parameter `maxAngle` was set to “25”, and parameters `rayCount` and `rangeCount` were set to “36”. Parameters `minAngle` and `maxAngle` were selected to limit the beam of the sensor to an area just clearing the hood of the model at left extent to slightly greater than horizontal at right extent. Parameters `rayCount` and `rangeCount` were selected based on an angular resolution of 1°.

- Trial 2

Parameters `rayCount` and `rangeCount` were set to “71”. Parameters `rayCount` and `rangeCount` were selected based on an angular resolution of 0.5º. The simulation configuration was otherwise identical to Trial 1.

- Trial 3

The SICK LMS 291 LIDAR model was relocated at the front center of the roof of the representative challenge vehicle model with a rotation of 5º around the y-axis so the beam of the sensor swept a horizontal plane at a down angle of 5º across the path of travel of the model. See Figures 12 and 13.

An angle of 5º was selected based on visual analysis which indicated the beam of the sensor completely crossed the path of travel at the maximum range possible. Increasing this value to 6º resulted in a reduced ability to detect obstacles near the ground because the beam did not intersect the ground within 30 m, which was the maximum range of the simulated SICK LMS 291 LIDAR sensor. Decreasing this value to 4º reduced the ability of the sensor to detect obstacles near the ground plane at the maximum range possible. See Figure 14.

Geometric analysis confirms this. At a height of 2.142 m and angle of 4º, the beam intersects the ground at a range of 30.6 m, exceeding the typical range with ten percent reflectivity for a SICK LMS 291 LIDAR sensor. At a height of 2.142 m and angle of 6º, the beam intersects the ground at a range of 20.4 m. An angle of 5º was selected to ensure obstacle detection at a range of 24.5 m, the maximum range at which the sensor completely crossed the path of travel.

Parameter minAngle was set to "-90", parameter maxAngle was set to "90", and parameters rayCount and rangeCount were set to "181". Parameters minAngle and maxAngle were selected based on the maximum scanning angle of SICK LMS 291 LIDAR sensors ([45]). Parameters rayCount and rangeCount were selected based on an angular resolution of 1°.

- Trial 4

Parameter minAngle was set to "-45", parameter maxAngle was set to "45", and parameters rayCount and rangeCount were set to "91". Parameters rayCount and rangeCount were selected based on an angular resolution of 1°. The simulation configuration was otherwise identical to Trial 3.

- Trial 5

Parameter minAngle was set to "-30", parameter maxAngle was set to "30", and parameters rayCount and rangeCount were set to "61". Parameters minAngle and maxAngle were selected to limit the beam of the sensor to an area including the outer wall and inner wall used to mark the lateral boundary offset. Parameters rayCount and rangeCount were selected based on an angular resolution of 1°. The simulation configuration was otherwise identical to Trial 3.

- Trial 6

The SICK LMS 291 LIDAR model was relocated at the front left corner of the roof of the representative challenge vehicle model with a rotation of -20° around the x-axis, 2° around the y-axis, and -10° around the z-axis so the beam of the sensor swept a diagonal plane across the path of travel of the model. These values were determined experimentally by making changes to the world file, loading the simulation, and observing the result. See Figures 15 and 16.

Parameter `minAngle` was set to "-10", parameter `maxAngle` was set to "45", and parameters `rayCount` and `rangeCount` were set to "56". Parameters `minAngle` and `maxAngle` were selected to limit the beam of the sensor to an area including the outer wall used to mark the lateral boundary offset to slightly greater than horizontal at right extent. Parameters `rayCount` and `rangeCount` were selected based on an angular resolution of 1°.

Table II. Results of the evaluation of the use of LIDAR.

Trial number	**Number of range returns**	**Number of rays**	**Quality**	**Maximum range[a] (m)**	**Minimum range[b] (m)**
1	33	36	0.917	20.5	14.6
2	65	71	0.915	20.5	14.6
3	150	181	0.829	24.0	4.1
4	104	91	1.143	24.0	6.1
5	88	61	1.443	24.0	8.3
6	19	56	0.339	29.6	17.6

Notes:

[a] Maximum range when the tower obstacle was first detected.

[b] Minimum range when the tower obstacle was last detected.

An immediate reduction in the ratio of simulation time to real time from approximately 0.4 for the previous evaluation to 0.2 during this evaluation was observed. Although Gazebo documentation stated: "Reducing the number of rays is a good way to save CPU cycles (at the expense of simulation fidelity).", the author did not find this to be the case. Through the trials documented above, changing the number of rays had little observable effect on the ratio of simulation time to real time. The author also observed no effect when initially reducing the update rate of the steering and laser controllers from 50 Hz to 20 Hz. The author concluded it was possible some other factor resulted in the reduction, such as frequent filesystem access caused by logging data.

VIII.E. Conclusions

With the vertical or diagonal configurations it was immediately obvious through visual analysis alone that the ability to detect obstacles in the path of travel was compromised, creating a “blind spot” or spots, and that the maximum range at which an obstacle would be detected in the path of the representative challenge vehicle model was greatly reduced. The data reflect this. Of the three configurations tested:

- The vertically-aligned LIDAR configurations produced fewer range returns than every horizontally-aligned LIDAR configuration, even when the scan was completed with twice the angular resolution. The diagonally-aligned LIDAR configuration produced the fewest range returns of any of the three configurations.
- The maximum range when the tower was first detected for the vertically-aligned LIDAR configurations was the least. The maximum range for the diagonally-aligned LIDAR configurations was the greatest. The horizontally-aligned LIDAR configurations detected the tower near the maximum range possible, considering the angle of the sensor was selected to ensure obstacle detection at the maximum range at which the sensor completely crossed the path of travel.

- The minimum range when the tower was last detected was greatest for the diagonally-aligned LIDAR configuration, and slightly less for the vertically-aligned LIDAR configurations. The minimum range when the tower was last detected was least for the horizontally-aligned LIDAR configurations.
- The quality of tested configurations was a maximum for Test 05. This test represented a horizontally-aligned LIDAR sensor with a down angle of 5º, able to detect obstacles with the area including the outer wall and inner wall used to mark the lateral boundary offset, or the entire possible path of travel of the representative challenge vehicle model. This configuration was the most popular LIDAR configuration in use by teams which participated in the 2004 or 2005 GCE.

As a result, the author concluded Player and Gazebo could be used to evaluate LIDAR sensor configurations successfully, allowing a team to very quickly reduce the number of possible configurations to those which best utilize existing computing resources, and to visualize the interaction of the challenge vehicle with the environment.

However, it is easy to misinterpret the results of this evaluation. Several teams reported a greater number of LIDAR sensors were in use oriented so they intersected the ground at different distances from the challenge vehicle, or in fixed horizontal or vertical planes. For example:

- Four SICK LIDAR sensors were in use by Team 2005-18 which were pointed “horizontally”, 3 m, 20 m, and 35 m away.

- Two “nearly horizontal” and three “vertically oriented” unknown SICK LIDAR sensors were in use by Team 2005-05.

By using vertically-aligned LIDAR sensors, Team 2005-06 was able to gain a competitive advantage over other teams, such as Team 2005-18, which reported multiple LIDAR sensors were in use which intersected the ground at different distances from the challenge vehicle. Vertically-aligned LIDAR sensors, by scanning a vertical plane, returned range readings to the maximum effective range of the LIDAR sensors in a horizontal plane despite the attitude of the vehicle, i.e., whether the vehicle was traveling downhill or uphill.

In addition, by using an oscillating mount, Team 2005-06 was able to use two vertically-aligned LIDAR sensors to detect obstacles directly in front of the vehicle and eliminate the field-of-view limitations consistent with fixed-mount vertically-aligned LIDAR sensors noted by Team 2005-05. In reference to the vertically-aligned LIDAR sensors in use by the team, Team 2005-05 stated: “The disadvantage, of course, is that since each ladar looks in only a single azimuthal direction, instantaneous azimuthal coverage is poor and obstacles between the vertical ladar scan planes will be missed.” ([48], p. 6).

Team 2005-06 reported the maximum effective range for the unknown SICK LIDAR sensors in use by the team was “approximately 40 to 50 m” ([47], p. 516). As a result, Team 2005-06 was able to extend the maximum effective range of the LIDAR sensors in use by the team to twice the maximum effective range reported by Teams

2005-13, 2005-14, and 2005-16.

By using an oscillating mount, Team 2005-06 was able to reduce the number of sensors to the minimum necessary, while retaining some redundancy.

The author considers this a key distinguishing factor which differentiated Team 2005-06 from all other teams which participated in the 2004 QID or GCE or 2005 GCE, and which contributed to Team 2005-06 successfully completing the 2005 GCE.

The author did not attempt to simulate the oscillating mount in use by Team 2005-06 due to time constraints, but concluded it would be possible to simulate an oscillating mount using Player and Gazebo. The author identified “Experiment with different LIDAR configurations” as a future research opportunity based on the results of this evaluation, and proposes a greater number of fixed-mount, horizontally-aligned LIDAR sensors may, in fact, provide a less dense point map than fewer oscillating-mount, vertically- or diagonally-aligned LIDAR sensors. See paragraph XI.I.

CHAPTER IX. EVALUATION OF FIELD-OF-VIEW LIMITATIONS

IX.A. Configuration of the simulation environment

A world file containing the turning circle mesh created to validate the steering controller (see Chapter VI.) was generated. A tower obstacle (see paragraph V.C.2.) was located at a position 5.128 m to the right and 7.128 m to the rear of the representative challenge vehicle model, ensuring the tower was in the path of travel of the model as the model traveled around the turning circle.

With the exception of the 70-degree field-of-view of the Navtech DS2000 RADAR in use by Teams 2005-13 and 2005-14[13], the Epsilon Lambda ELSC71-1A RADAR ("ELSC71-1A") has the widest field-of-view of any navigation RADAR in use by teams participating in the 2004 or 2005 GCE. To simulate and visualize the field-of-view limitations of the ELSC71-1A, the world file was revised to attach a SICK LMS 200 LIDAR model to the representative challenge vehicle model with field-of-view characteristic of the ELSC71-1A. The ELSC71-1A has a field-of-view of +/- 20 degrees in wide-scan mode. This corresponds to a maximum distance between the path of travel in a constant-radius turn and the left- or right-limit of field-of-view of 0.400 m. The SICK LMS 200 LIDAR model file was revised to limit the field-of-view to +/- 20 degrees.

The representative challenge vehicle model file was revised to set parameters `useConstantSteeringAngleMode` to `TRUE`, `constantSteeringAngle` to `-0.3764`, and `useSafeVelocity` to `TRUE`.

IX.B. Simulation procedure

The author started Gazebo, started Player, started the `playerv` utility, and observed the model as it accelerated through the turning circle toward the tower.

IX.C. Results

See Figures 17, 18, and 19. As predicted, the field-of-view of the simulated ELSC71-1A was not wide enough to detect the tower obstacle located 0.5 m from the representative challenge vehicle path of travel. As a result, the obstacle was not detected, virtually guaranteeing a collision.

IX.D. Conclusions

The author concluded Player and Gazebo could be used to effectively visualize sensor field-of-view limitations successfully. This may have eliminated the use of navigation RADAR as a primary obstacle detection sensor, reduced cost to the teams, and enabled less experienced teams to more effectively visualize the interaction of their challenge vehicle with the environment.

CHAPTER X. OVERALL CONCLUSIONS

In an attempt to determine what problems, exactly, were solved during the Grand Challenge, the author identified key factors. Several key factors which could have been evaluated through the use of simulation prior to the Grand Challenge were identified as potential simulation targets. Although installation and use of Player and Gazebo presented challenges, the author successfully evaluated three simulation targets using Player and Gazebo.

Overall, the author concluded the use of simulation would have enabled teams to effectively visualize the interaction of their challenge vehicles with the environment, and quickly and easily prototype and evaluate ideas such as the oscillating sensor mount in use by Team 2005-06.

During the evaluation of LIDAR configuration, the use of XML configuration files by Player and Gazebo to configure the simulation greatly increased flexibility and ease-of-use. The author was able to modify the simulation configuration quickly between trials by changing one or two lines in a text file.

In addition, the use of simulation made the results *reproducible*, with a high degree of fidelity. As a result, the author concluded attempted solutions to problems encountered during this research could be confirmed to be effective in simulation by eliminating variability in initial conditions.

The four teams that successfully completed the 2005 GCE completed the course in 06:53:58 hours (Team 2005-16), 07:04:50 hours (Team 2005-14), 07:14:00 hours (Team 2005-13), and 07:30:16 hours (Team 2005-06), although Teams 2005-13 and

2005-14 adopted a "dual speed" strategy and estimated completion of the 2005 GCE in 06:19:00 and 07:01:00 hours, respectively.

Teams 2005-13 and 2005-14 later stated ([21, p. 500]):

> While the strategy was successful in that both robots completed the challenge, it limited [the challenge vehicle] below its ability and, in retrospect, prevented it from winning the Grand Challenge.

Team 2005-13 did not complete the course in the projected time because of undiagnosed engine problems ([21, p. 502]).

The author implemented an improved steering controller that limited the velocity of the vehicle to the maximum allowed by vehicle and course geometry and was able to demonstrate that a model using this controller would not be subject to rollover in 2004 GCE course segment 2570-2571-2572. The representative challenge vehicle and many other challenge vehicles would have been at risk of rollover if entering 2004 GCE course segment 2570-2571-2572 at the maximum velocity allowed by the 2004 GCE RDDF.

This was similar in concept to the method Team 2005-16 reported was in use by the team during the 2005 GCE, and which the author considers a key distinguishing factor. Team 2005-16 stated ([49], p. 10):

> ...[the challenge vehicle] always assumes an allowable velocity according to pre-processed RDDF file, and it slows down in curves so as to retain the ability to avoid unexpected obstacles. The vehicle also adapts its velocity to the roughness of terrain, and to the

nearness of obstacles. The specific transfer function emulates human driving characteristics, and is learned from data gathered through human driving.

To attain a suitable trajectory and associated maximum velocity, the RDDF file is processed by a smoother. The smoother adds additional via points [*sic*] and ensures that the resulting trajectory possesses relatively smooth curvature. The preprocessing then also generates velocities so that while executing a turn, the robot never exceeds a velocity that might jeopardize the vehicle's ability to avoid sudden obstacles. This calculation is based on a physical model of the actual vehicle.

However, the author concluded that by eliminating the risk of rollover, it was possible for teams to complete the 2004 GCE course at the maximum speed allowed by the RDDF, with the sole exception of 2004 GCE course segment 2570-2571-2572, in less time than Team 2005-16 completed the 2005 GCE (6.90 hr), *with no pre-planning or pre-mapping required*.

DARPA stated ([50]):

Course speeds that are between 26mph and 50 mph (inclusive) are advisory and are provided for guidance purposes.

As a result, the author concluded, due to the safety factor inherent in the design of the 2004 and 2005 GCE courses, it would have been possible for a challenge vehicle to have completed the course in less time than the ideal time by traveling at speeds higher than the “advisory” speed limits allowed by the RDDF at no additional risk of rollover to the vehicle.

Although the teams could not have known this before receiving the 2004 GCE RDDF, they could have performed the analysis documented by the Technical Report based on a model of their challenge vehicle dynamics. The author asserts this may have altered the pre-mapping or path editing strategies in use, and might have provided a competitive basis for the 2004 GCE similar to that of the 2005 GCE.

CHAPTER XI. FUTURE RESEARCH

XI.A. Use a sensor to train the controlling intelligence to interpret other sensors

Team 2005-16 used LIDAR sensors to train a single color camera to detect obstacles at a range which exceeded the maximum effective range of LIDAR sensors[14]. Team 2005-16 stated: “To extend the sensor range enough to allow safe driving at 35 mph, [the challenge vehicle] uses a color camera to find drivable surfaces at ranges exceeding that of the laser analysis.” ([51], p. 672). However, this strategy could be extended to other combinations of sensors in simulation. For example:

- GPS/INS/IMU output could be used to train the controlling intelligence to detect “slippage” of steering position and odometry. DARPA stated an “independent technical evaluation team identified the following technology from Grand Challenge 2004 noteworthy”: “Sensor-based slippage detection (conceptual)” ([3], pp. 10 - 11)[15].
- LIDAR sensors could be used to train RADAR sensors to see farther up the road, increasing the maximum effective range of RADAR sensors, or providing a basis for the development of more effective navigation RADAR.
- Position sensors could be used to develop algorithms to integrate incremental distance measurements provided by sensors such as magnetic or optical encoders on axles or the drive shaft, differential odometers, etc. more effectively.

    This is similar to the strategy utilized by COTS components. For example, Team 2005-06 stated: “[Team 2005-06] chose to use the RT3000 from Oxford Technical Solutions to provide vehicle localization. ... The integrated INS allows the RT3000 to survive GPS outages of up to 30 seconds with virtually no performance degradation. Because the GPS and INS are integrated together, each can compensate for problems with the other. For example, if the INS started to drift laterally, the integrated GPS will automatically correct that drift." ([53], p. 9).

- Distance could be estimated by throttle position for unit time and slope, and integrated over changes in terrain roughness, providing an alternative to dead-reckoning.

In addition, this strategy could be extended to combinations of sensors which are not obviously complementary. For example:

- Team 2004-07 described how the controlling intelligence used information such as time of day, orientation, and lighting conditions to detect obstacles: "Since the system will know the time of day, its orientation, and the lighting conditions, it can employ a shape-from-shading and shape-from-shadow system to determine the approximate position and dimensions of obstacles like large rocks or craters." ([54], p. 5). However, there is no reason the controlling intelligence would not be able to determine the time of day, orientation, or lighting conditions using the approximate position and dimensions of obstacles.
- Team 2004-09 stated: "Road boundaries and obstacles will be reliably detected when the vehicle is bouncing over rough terrain and turns. We will use a rapid shutter speed of 1/8000 sec. to minimize blurring. We will mount the camera and

other sensors on a platform designed to absorb shock. Inertial data will normalize the image perpendicular to the ground when the vehicle is tilted one direction or the other. In addition, when the vehicle is driving over uneven terrain, the normalization process attempts to use information from previous images to locate the horizon and road. Topographic information may also be used to locate the horizon and road. Images that do not normalize to recognizable data can be skipped because the frame rate of 30 frames/sec. is more than sufficient to allow us to dispose of 'bad frames.' If the vehicle is tilted upward or downward so that the camera is facing images of sky or ground, the autonomous control can use pitch information to discard those frames." ([55], p. 7).

This is similar in concept to Team 2005-16's later use of LIDAR sensors to train a single color camera to detect obstacles at a range which exceeded the maximum effective range of LIDAR sensors, but using shock, vehicle attitude relative to the horizon, and accelerometer data to normalize data. In their technical proposal, Team 2004-09 does not report their controlling intelligence was trained to normalize the data, but learning to normalize visual processing data is a potential task for a controlling intelligence.

XI.B. Emergence of unexpected behavior

In general human beings observe certain "rules of the road": they navigate roads with recognizable characteristics such as color, texture, lane markings, and signage which establishes context and allows them to determine what is, and is not, a road; they travel from point to point in lanes, the width of which varies depending on location; and they must obey posted speed limits. However, human beings are not constrained by the electronic equivalent of an overwhelming compulsion.

A truly autonomous vehicle would be able to evaluate its own objectives within the constraints imposed on it by its programming, and it would violate some rules if necessary to accomplish them, for example, by proceeding on a more direct course to its destination if requested to travel a circuitous course similar to the 2005 GCE course which crosses and overlaps itself in several areas. There is no evidence that the emergence of unexpected behaviors was a goal or outcome of the Grand Challenge. If the development of artificial intelligence is a goal of autonomous vehicle development, the emergence of unexpected behaviors would be a measure of successful development.

XI.C. Development of novel sensor technologies

Several teams attempted to use low-cost photoelectric, ultrasonic, or short-range RADAR sensors to provide useful information to the controlling intelligence. Because of their limited utility in practice, these sensors were discounted by the author. However, DARPA stated an "independent technical evaluation team identified the following technology from Grand Challenge 2004 noteworthy": "Extended range of low-cost, ultrasonic sensors" and "Single-point laser rangefinder as a low-cost distance sensor" ([3], pp. 10 - 11).

Simulation might enable the identification and development of novel sensor technologies, such as a SONAR sensor array that provides a 3D point map as accurate as that provided by a LIDAR sensor, but using SONAR returns, the effective use of non-scanning LIDAR sensors, or the development of a goniometer (direction-finding antenna) for providing accurate position information. As a minimum, the use of simulation might provide an environment in which the practical applications of such sensors could be explored.

In addition, combined sensor strategies in use by teams which participated in the 2004 and 2005 GCE included the use of LIDAR in combination with high-quality STEREO or RADAR, but alternate strategies were in use. Each strategy was specifically tailored to a challenge vehicle. Simulation might increase the likelihood the generic application of the combined sensor strategies in use by most teams would be adequately explored and potential commercial applications identified.

XI.D. Use simulation to train the controlling intelligence to recover from a loss of sensor data or other sensor failure

XI.D.1. Primary obstacle and path detection sensor

Several teams reported a single sensor was in use by the team as the primary obstacle and path detection sensor:

- A proprietary stereo camera pair was in use by Team 2004-06.
- One SICK LMS 291-S05 was in use by Team 2004-12.
- One Epsilon Lambda ELSC71-1A was in use by Team 2004-21.
- A proprietary video system was in use by Team 2004-22.
- A proprietary LIDAR sensor was in use by Team 2005-03.
- A Point Grey Bumblebee stereo camera pair was in use by Team 2005-12.

Neither Team 2004-06, 2004-12, 2004-21, 2004-22, 2005-03, nor 2005-12 reported how the challenge vehicle controlling intelligence would respond to the loss of the single primary obstacle and path detection sensor.

In general, teams which reported multiple obstacle and path detection sensors were in use by the team also did not describe how the controlling intelligence would respond to the loss of a sensor. Three teams which participated in the 2005 GCE reported a sensor, type of sensor, or array of sensors was "redundant" in the sense that it provided obstacle and path detection information in the event a sensor failed. The author considers this to be functional redundancy. For example:

- Several obstacle and path detection sensors were in use by Team 2005-08, including three Delphi Forewarn ACC3 RADAR. Team 2005-08 stated: "[The Delphi Forewarn ACC3 RADAR] can act as a redundant sensor for the [challenge vehicle]." ([56], p. 9).
- Although the author concluded ultrasonic sensors were not in use by Team 2005-15, Team 2005-15 stated: "...the ultrasound sensors act as additional redundant sensors, which are less susceptible to dust or fog." ([22], p. 9).
- Team 2005-20 stated: "Our goals were to... develop a sensor array that contains redundancy for accuracy and reliability..." ([29], p. 2).

The author considers it likely teams selected multiple complementary obstacle and path detection sensors by necessity and to have functional redundancy. For example, Team 2005-10 stated: "There does not appear to be any one sensor that can 'do it all'. Each sensor has its strengths and its weaknesses." ([57], p. 7).

However few teams reported how the controlling intelligence would respond to the loss of sensor data or other sensor failure, perhaps because DARPA did not explicitly request teams provide such information. In contrast, DARPA explicitly requested teams determine how the controlling intelligence would respond to "GPS outages". As a result, teams generally reported how the controlling intelligence would respond to the loss of GPS data or GPS failure. See paragraph XI.D.2.

Several teams acknowledged the loss of sensor data or other sensor failure would affect challenge vehicle performance. For example:

- Team 2004-01 stated: "Speed setting algorithms will take into consideration the following and reduce speed appropriately: ... Sensor obstruction ... Sensor disagreement, Data discontinuities or gaps ... Component failure" ([58], pp. 6 - 7).
- Team 2004-02 stated: "Component failure testing: Since [the challenge vehicle] cannot operate without power, testing will be done to insure that the vehicle has power the whole race. These tests will include cutting power to individual sensors, computers, and support electrical units." ([59], p. 13).
- Team 2005-04 stated: "These sensors are monitored for changes in their operating state, validated using both dynamic and rule based tests, and finally fused using a Kalman filter based approach to provide continuous position and orientation information even [*sic*] the presence of individual sensor dropouts, reduced accuracies, or complete failures." ([13], p. 11).
- Team 2005-11 stated: "Hardware and software have been designed to minimize the impact of temporary failed components. However, limited redundancy in

components means that permanent outages of sensors will have a detrimental effect on [the challenge vehicle's] performance." ([18], p. 7).

Finally, a few teams reported specific action to be taken to resolve a loss of sensor data or other sensor failure. None of these teams described how the controlling intelligence would respond to a loss of sensor data or other sensor failure:

- Team 2005-12 stated: "The emergency brake's pneumatic system is setup such that any failure of the [the challenge vehicle's] software or hardware will result in an emergency brake application." ([60], p. 3).
- Team 2005-20 stated: "A failure of any individual sensor results in no information being broadcast from that specific sensor." ([29], p. 7).
- Teams 2004-13, 2004-14, and 2005-15 reported an emphasis on the isolation of hardware and software modules from each other so that a failure in one module does not cause an overall failure, and Teams 2005-13, 2005-14, 2005-16, and 2005-19 reported an emphasis on restarting modular hardware and software components.

Three teams reported a loss of obstacle and path detection sensors or other sensor failure during the 2005 GCE: Teams 2005-14, 2005-15, and 2005-18. Team 2005-14 successfully completed the 2005 GCE. Team 2005-15 reported a loss of all LIDAR sensor and internal state data due to a "USB hub" failure. Team 2005-18 reported a loss of "midrange" LIDAR sensor data.

The author concluded the failures were preventable system integration failures. Because Team 2005-14 had significant experience but neither Team 2005-15 nor 2005-18 had significant experience, the author proposes the use of simulation may have helped "level the playing field", by enabling teams without significant experience to learn how to recover from a loss of sensor data or other sensor failure as well as an experienced team and eliminate the causes of the preventable system integration failures which resulted in their failure to complete the 2005 GCE.

XI.D.2. GPS sensor failure

GPS "drift" or "jumps" were consistently reported by teams which participated in the 2004 QID or GCE or 2005 GCE. For example, Team 2005-05 stated: "Very often, especially when the vehicle would drive near a wall or approach a tunnel, there would be highly erratic jumps in the GPS measurements due to multipath reflections." ([15], p. 542).

In addition, GPS sensor failure was directly implicated in the failure of five teams to complete the 2005 GCE: Teams 2005-02, 2005-09, 2005-15, 2005-18, and 2005-19. For example, Team 2005-02 stated: "... it appears that the calculated GPS position drifted by approximately 20 feet causing the vehicle to want to move to the right of the actual road.", which caused "a corresponding shift of the boundary smart sensor that eliminated the actual sensed road as an option to the planner." ([12], p. 621).

DARPA, via 2004 SQ 1.g.2 and 2005 SQ 2.2.1 requested teams describe how they would handle "GPS outages"[16]. In general, teams described how the challenge vehicle controlling intelligence would continue to determine position reliably in the absence of

GPS data. A few teams described test and evaluation to determine the effect of GPS outage on the challenge vehicle controlling intelligence. For example:

- Team 2004-17 stated: "We have tested the ability of various materials to block antenna reception. Flat sheets of aluminum and Lucite were unable to block the GPS, as multi-path reflections off of the ground still reached the antenna. Wrapping the antenna in aluminum foil cut off reception (we can selectively cut off satellites and simulate GPS outages)." ([61], p. 12).
- Team 2005-06 stated: "Another extremely effective test involved manually steering the vehicle off course at high speed and then switching back to autonomous mode. This simulated a GPS jump, which can occur rather frequently. After noticing that the navigation system abruptly turned the steering wheel to counteract this jump, the navigation system was updated to eliminate this abrupt movement." ([53], p. 12).

In addition, the military deployment of autonomous ground vehicles will result in the development of countermeasures to preclude their use[17]. For example:

- Strong magnetic fields may confuse magnetometers, causing the vehicle's controlling intelligence to incorrectly interpret compass headings.
- The U. S. government's ability to control the accuracy of GPS position information using "selective availability" is a strategic limitation on the use of GPS. Although U. S. military ground vehicles would not likely be affected by selective availability, an effective controlling intelligence should be able to

identify the problem if it occurs and adjust the weight of other sensors appropriately or take corrective action to determine geolocation using some other method, such as dead-reckoning. Alternate strategies, such as using beacons or reflectors delivered by artillery, or aerial drones, to provide stable “known” geolocation similar to survey markers may also be successful.

Although effective, the test method employed by Team 2005-06 represented a real risk to the team challenge vehicle. Effective simulation may have allowed teams to develop strategies to mitigate the effects of GPS drift or jumps and to gracefully recover from a temporary or permanent loss of GPS sensor data by allowing a model to be driven off course, then “switching back to autonomous mode” in a manner similar to that reported by Team 2005-06, but without risk to the team challenge vehicle.

XI.E. Standardization and standard references

XI.E.1. Standard dictionary, acronyms, and abbreviations

Develop a standard dictionary of terms and their associated acronyms and abbreviations for use in future research similar to the Grand Challenge to be maintained as a set of user-defined dictionaries for various word-processing applications. The author acknowledges that current word-processing software is limited in the amount of customization that it provides. For example, OpenOffice.org Writer (version 3.0.1) supports custom dictionaries, but does not yet provide a way to import or export custom dictionaries. In addition, there is a finite limit to the number of words allowed in a custom dictionary.

XI.E.2. Standard reference terrain

Develop a library of standard reference terrains using available sensors to gather complete data using environment and geolocation sensors consistent with the state of the art. For example, using a research platform with roof-mounted cameras, and LIDAR, RADAR, and GPS sensors visit:

- the Pennsylvania Turnpike, to record mountainous terrain, including several extremely long tunnels during which GPS reception will be lost
- the Mojave Desert, to record desert terrain, including the "negative obstacles" typically encountered in desert terrain such as wadis
- the California coast on US-1 (the Pacific Coast Highway), to record coastal highway, extending north through the Redwood National Forest
- Interstate 40, to record a long traversal across the United States with many different reference terrains

While traversing reference terrain, record the precise geolocation on a continuous basis. Use existing technologies to subtract vehicles and other obstructions from the reference terrain as recorded by the environment sensors in use. Correlate GPS position with the terrain in simulation. Use the LIDAR data to produce a "point map" of the reference terrain, and map the return from camera sensors onto this point map as a trimesh, providing simulated cameras with more realistic data.

This would make it possible to add vehicles and other obstructions as desired, or to test the controlling intelligence in an environment completely devoid of risk to other vehicles while still allowing it to perceive at the limit of available environment sensor technology.

In addition to recording the standard reference terrain in different locations, record the standard reference terrain at different times of the day and year. Although there may be little difference to a LIDAR sensor from night to day, the difference to a camera will be significant. In addition, there will be a significant difference between the efficiency of a camera or LIDAR sensor pointed into the sun at sunrise, heading east, or sunset, heading west, and at other times of the day. Terrain details may be obscured by snow during the day, or brought into sharper contrast at night. All of this is useful information to the controlling intelligence.

Simulation environments such as the Player Project could be modified to use simulated reference terrain for real-time testing.

XI.E.3. Standard obstacle and position problems

Develop a library of standard obstacle and position problems (herein "standard problems"), and acceptable responses based on human driving tests. These problems should first be implemented in simulation to support the development of algorithms and acceptable responses. Acceptable responses should then be verified during real world testing. For example:

Every state has established a standardized program of driver education which requires a minimum level of competency to be demonstrated by drivers prior to licensure. For example, in Virginia, this program is called "The Driver Education Standards of Learning and the Curriculum and Administrative Guide for Driver Education in Virginia" (herein “Guide”) ([62]). The Guide describes a series of “Modules” presenting required course content. Module 11 is titled “Laboratory Instruction – Behind-the-Wheel and In-car Observation”. Module 11 describes a series of “Lessons”, “Basic Skills”, and “Driving Procedures”, which ensure the driver has achieved a minimum level of competency ([63]). Successful completion of the 2007 Urban Challenge was determined, in part, by the challenge vehicle's controlling intelligence's ability to obey California state traffic laws.

It is not unreasonable to require an autonomous vehicle's controlling intelligence to meet or exceed the basic minimum level of competency expected of a human driver, in effect making the standard problems, and acceptable responses, a “Turing test” for autonomous vehicle controlling intelligences.

It is unreasonable to expect the public to be forgiving of an autonomous vehicle which loses contact with a GPS signal, for example, and unexpectedly stops in a tunnel during rush-hour traffic, or to accept the loss of life and property damage that may be caused by an autonomous vehicle that loses the ability to distinguish between the road and terrain in the rain, and crosses the center line of a divided highway with disastrous consequences. As a result, standard problems must also evaluate the controlling intelligence's ability to meet or exceed the basic minimum level of competency expected of a human driver in similar situations.

Also, this approach would allow the controlling intelligence to be trained to respond to situations in a manner uncharacteristic of human drivers. For example, a human driver reacting to a vehicle entering the lane next to his or her vehicle might react out of fear, pulling the steering wheel suddenly to the right or left to avoid collision, and entering the next lane, unintentionally causing an accident. An autonomous vehicle's controlling intelligence would be able to more effectively estimate the position of the autonomous vehicle in relation to its surroundings, and decide not to attempt to avoid a collision if attempting to avoid the collision will cause a collision with another vehicle and if the autonomous vehicle will not be seriously damaged. However, if the vehicle pulling into the lane next to it is a 40-ton tractor-trailer, the autonomous vehicle's controlling intelligence might conclude a collision is unavoidable, and decide to collide with a lighter vehicle, due to the tractor-trailer's greater damage potential.

XI.E.4. Team descriptions of standard reference terrain and standard problems

Several teams described attempts to gather standard reference terrain or proposed the implementation of standard problems. However, no team proposal was comprehensive. For example:

- Teams 2004-13 and 2004-14

Teams 2004-13 and 2004-14 were co-competitors during the 2004 GCE, and stated: “During field trips to the Mojave desert, we have recorded more than 7 hours of video from a vehicle-mounted camera, recording the path ahead. We have run parts of these video sequences through our path tracking software.” ([64], p. 6 and [65], p. 7). However, this approach was not comprehensive, in that it did not allow the teams to adjust the mounting of the camera to optimize the performance of their path tracking software or experiment with different types of cameras.

- Team 2004-20

Team 2004-20 stated: "The road-follower software has been tested against video recordings of desert roads, with marginally satisfactory results. The imagery used was too narrow. The road follower is being revised and will be retested with wider-field imagery." ([52], p. 9). As noted by Team 2004-20, this approach was not comprehensive, in that it did not allow the team to adjust the field-of-view of the camera to optimize the performance of their road-following software.

- Team 2004-23

Team 2004-23 described a special type of terrain called "Robot", and stated ([34], p. 6):

```
"Robot" is a special terrain/location where the
vehicle has to go through a specific exercise,
possibly with a set of predetermined operations, to go
past an obstacle or through a narrow constrained
passage.
Examples where Robot behavior may be needed include
underpasses, gates, sharp turns at roadway
intersections and possible passage through mazes of
natural and synthetic obstacles.
```

- Team 2005-01

In response to 2005 SQ 2.4.1, Team 2005-01 stated ([66], p. 11):

```
Extensive testing in the field has led to extensive
development of these corner cases.  [The challenge
```

vehicle] does not return to missed waypoints, since in many cases the road is not wide enough to make a full turn to reach the missed waypoint. The vehicle will continue along the assigned path in this case.

When the vehicle is "stuck", this may occur with wheels slipping, and the vehicle is not actually driving forward. For this case, we detect this condition in the National Instruments software, and reverse a few meters to free ourselves from this condition.

If the vehicle travels out of bounds, the "boundary" voter immediately pushes us back into bounds by providing a strong negative weight along any path that continues out of bounds. If an obstacle is detected in the path, the vehicle detects this with either the four LADAR sensors or the five bumblebee cameras. Upon detection, the vehicle's path is adjusted to pass the obstacle by with a safety margin.

- Team 2005-04

Team 2005-04 described a special case for braking or starting on a hill: "The speed set point is generated regardless of the slope of the ground. The speed controller has the 'integration' part that keeps increasing the throttle if the vehicle is slower than the speed set point so that we can climb a hill. In order to stop short in some situations, the vehicle applies the maximum brake pressure." ([13] , p. 13).

At least one team failed to complete the 2004 GCE due to an inability to increase throttle sufficiently to climb a steep hill. Team 2005-05 participated in the 2004 GCE as Team 2004-07. Team 2005-05 later stated: "[The Team 2004-07 challenge vehicle] traveled 5.1 miles in the 2004 Challenge... before stopping on a steep slope because of an excessively conservative safety limit on the throttle control." ([15], p. 528). As a result, the author considers this problem a potential standard problem.

- Team 2005-08

Team 2005-08 stated: "...in December 2004 a team of engineers with two sensor instrumented platforms drove large segments of the course, collecting navigation, image, and laser data for algorithm development and design validation for components such as the shock isolation sled." ([56], p. 22). However, this effort was not comprehensive. Although Team 2005-08 collected standard reference terrain similar to that expected to be encountered during the 2005 GCE, the development of fully autonomous vehicles will require a greater library of reference terrain be available.

- Teams 2005-13 and 2005-14

Teams 2005-13 and 2005-14 stated: "Standardized tests must be developed that measure a robot's ability to sense and accurately localize obstacles of varying size. These tests should account for differing perception sensing modes. Standard tests that measure an autonomous vehicle's ability to safely and reliably interact with other vehicles and humans are needed. These tests and others are required in order to move autonomous ground vehicles from technological curiosities to common tools used by people everywhere." ([21], p. 499).

However, a library of standard obstacle detection tests is not enough. Terrain affects obstacle detection and avoidance. Autonomous vehicles must also be taught to recognize degraded sensor performance not caused by simple failure of the sensor, such as lack of calibration or misalignment. For example, Team 2005-05 stated: "The virtue of ladars used in this vertical-plane configuration is that the ground profiles are easy to interpret, and are not particularly prone to confusion due to rolling, pitching, or bouncing motion of the vehicle. (Of course, a six-degree error in pitch could make a marginally-traversable 27-degree slope appear to be a marginally-untraversable 33-degree slope, or vice versa." ([48], p. 6).

In this particular example, the purpose of the standard pose estimation problem would be to teach the challenge vehicle controlling intelligence to recognize the error in pitch is caused by sensor misalignment, and compensate accordingly, and not treat the error as a permanent change in slope resulting in a determination that traversable terrain is not traversable, thus overcoming sensory input that is contra-indicative of the challenge vehicle's capabilities.

Multiple solutions to such a problem exist, depending on available sensors. In this example, the controlling intelligence may be able to estimate the slope of a road by measuring the distance known acceleration moves the challenge vehicle in a given time; the controlling intelligence may be able to utilize an altitude sensor or the elevation reported by commercial GPS to arrive at an independent estimate of the slope of the path of travel; or the controlling intelligence may be able to navigate the challenge vehicle over terrain with known characteristics, such as alternating high and low “striping”, i.e., asphalt or concrete of alternating heights, to determine the error in pitch of the LIDAR sensors in use.

XI.F. Time- and space-shifting

Player and Gazebo provide a “passthrough” construct which allows a client program connecting to the Player server to receive sensor output from a client program connecting to another Player server, and to effectively “see through their eyes”. The author proposes using this or a similar construct to allow notional vehicles (vehicles with no density or which do not implement ODE collision callback functions) to be “stacked” in time or space, allowing the controlling intelligence to receive sensor output from the simulation at some time offset in the future. This would allow the controlling intelligence to use the future results of current decisions to make more informed decisions, and effectively give the controlling intelligence the ability to “see” into the future as a training tool.

XI.G. Acclimation

Develop a process of “acclimation”, whereby the controlling intelligence queries a hardware- and software-independent abstraction layer to discover available sensors, and then uses standard reference terrain and standard problems to acclimate itself to their use. The acclimation process would require the controlling intelligence to learn how its outputs correlate with inputs to the abstraction layer, and vice versa, in effect calibrating itself[18]. This would make the controlling intelligence portable between vehicles, and allow one team to install their controlling intelligence in another team's challenge vehicle. As a result, teams would be competing not on the basis of hardware available to the team, but on the basis of their use of information available from standard interfaces.

For example:

- a challenge vehicle controlling intelligence could determine its own braking profile in a manner consistent with the method used by the U. S. Department of Transportation if visible markers with known spacing for VISION or STEREO sensors were painted, or vertical markers for LIDAR or RADAR sensors were placed, on a stretch of asphalt where they could be detected by a challenge vehicle's sensors.
- a challenge vehicle could similarly determine its own turn radius, or calibrate control of the steering wheel, gas pedal, or brake pedal, allowing a controlling intelligence using a hardware- and software-independent abstraction layer to calibrate itself to the specific vehicle in which it is installed.

XI.H. Least free energy state

Develop a process for correlating “desirability” or “traversability” maps to a concept such as the Gibbs free energy, to allow already-existing concepts to be used to describe the cost associated with moving from one metastable state to another.

Obstacles would be represented as local maxima, regardless of whether they were “positive” or “negative” obstacles. The height of the obstacle could be correlated to the potential damage the vehicle that would result in the event of a collision. Each sensor or combination of sensors would contribute an individual state map.

Road boundaries would be represented as continuous maxima “walls” of varying height, depending on how tolerant the terrain is to the controlling intelligence deciding to leave the road.

The difference between the height of the road and obstacle height maxima would determine, for example, whether the autonomous vehicle would attempt to leave the road to avoid an obstacle.

A route would be represented as a continuously decreasing “valley” in the local terrain map.

Local maxima representing obstacles detected by LIDAR sensors would be added to local maxima representing obstacles detected by RADAR, road boundaries, and the route to produce a final traversability map.

The autonomous vehicle's controlling intelligence would always seek to travel from one potential energy state to another, always moving from a greater potential energy state to a lower one, like water flowing downhill.

For example:

- Team 2004-07

Team 2004-07 stated: “...a nominal minimum-cost route from each waypoint to the next will be computed based on map data using a wavefront-propagation path planner.” ([54], p. 5).

- Team 2004-15

Team 2004-15 described a “desirability map” ([67], p. 9) that suggests the controlling intelligence was using a map similar to a free energy diagram, with geolocation represented by the x- and y-axes, and “desirability” by the z-axis. This suggests a decrease in desirability represents a positive slope in the free energy diagram, or negative reinforcement to the controlling intelligence, and that an increase in desirability represents a negative slope, or enticement. However, this model suggests the controlling intelligence would not be able to enter an area which represents a temporary increase in free energy (or lower desirability) to cross to an area at a net decrease in free energy (or higher desirability). As a result, the controlling intelligence might become stuck in a metastable state, from which it would not be able to free itself.

In addition, this approach would eliminate the potential problem of long-term “statelessness” described by Team 2004-15 as the “heading circle”, and as a result of which the controlling intelligence might be unable to ascertain if it is moving back and forth between two positions of high desirability.

- Team 2004-20

Team 2004-20 maintained an extensive online repository which contained several revisions of their technical paper prior to the final version accepted by DARPA ([52]), including DARPA responses to their first and second revisions indicating that DARPA requested Team 2004-20 report: “How will the potential field path planner escape from local minima?”. No 2004 SQ contains the words “local maxima” or “local minima”. Team 2004-20 stated: “Escaping from local minima is the job of the 'higher level' processing...” ([52], p. 3).

• Team 2005-13

Team 2005-13 stated: “Fusion of perception data is via a terrain cost map and binary obstacle map. Terrain cost maps are generated by evaluating the relative height of a sensed area to its neighbors and assigning a cost of 0 to 255 to that area. Binary obstacle maps are created in a two step process. First, an object detection algorithm, customized for each sensor group, detects and localizes obstacles. Second, detected obstacles are written into a map at the detected location.” ([19], p. 10).

In addition, most vehicles have a rollover threshold, a slope on which the vehicle will roll. For example, Team 2004-23 stated: “The vehicle can traverse a 60% grade and a 30% side slope.” ([34], p. 1). Typically, the left-right rollover threshold is much less for “side slope” than the front-back threshold for “grade”. Therefore, any solution utilizing a desirability or traversability map should assign a higher traversability to a sloped surface it will be required to traverse parallel to the slope, versus a sloped surface it will be required to traverse perpendicular to the slope.

XI.I. Experiment with different LIDAR configurations

By not orienting the sensors so that they intersected the ground at a fixed distance from the vehicle, Team 2005-06 was able to make effective use of LIDAR sensors by detecting obstacles as far from the vehicle as possible, and by using an oscillating mount, Team 2005-06 was able to reduce the number of sensors to the minimum necessary to accomplish this with some redundancy. The author considers this a key distinguishing factor which differentiated Team 2005-06 from all other teams which participated in the 2004 QID or GCE or 2005 GCE, and which contributed to Team 2005-06 successfully completing the 2005 GCE. See paragraph VIII.E. The use of simulation might allow alternate mounting configurations to be objectively evaluated, revealing which are of interest to further study.

XI.J. Extend the maximum effective range of high-quality sensors

Extend the obstacle detection range of high-quality sensors to enable the controlling intelligence to detect obstacles at ranges consistent with speeds an autonomous vehicle may reasonably be expected to travel. For example, in general highway speed limits in the United States are between 60 and 70 mph. However, the maximum effective ranges of sensors in use by teams participating in the 2004 and 2005 GCE correspond to a maximum speed of 47.6 mph (VISION sensors), 40.2 mph (RADAR sensors), 36.0 mph (long-range LIDAR sensors), and 25.5 mph (short-range LIDAR sensors).

In addition, no team reported a maximum speed greater than 38.0 mph. The maximum speed reported by Team 2004-10 during the 2004 GCE was 36 mph ([68], p. 31) and the maximum speed reported by Team 2005-16 during the 2005 GCE was 38.0 mph ([51], p. 688). The maximum reported speed corresponds to a maximum effective range of 44.6 m, between the maximum effective ranges for RADAR and long-range LIDAR sensors.

Extending the maximum effective range of high-quality sensors will be necessary before an autonomous vehicle will be able to achieve speeds consistent with general highway speed limits.

XI.K. Use alternate speed setting strategies

Implement a controlling intelligence that wants to drive as fast as it can, and in the most direct bearing to goal. The nodes in this example would exert a “negative pressure”, that is, they would exert the equivalent of a braking force to the autonomous vehicle as it attempts to drive with the throttle wide open, or the equivalent of a third hand on the steering wheel providing a change in bearing. The resistance “felt” by the steering wheel or gas pedal to negative pressure would be tuned to circumstances in the local environment. For example, under normal driving conditions at high speed, the controlling intelligence would resist minor pressure at high speeds, but not low speeds; under normal driving conditions at low speeds, the controlling intelligence would experience the equivalent of a driver in the passenger seat reaching across to suddenly grab the steering wheel and change course to radically alter bearing, or to prevent the controlling intelligence from turning into an obstacle.

XI.L. Make provisions to maintain the published record

The 2004 and 2005 GCE made extensive use of the Internet to solicit participation, provide access to team resources, publish requirements, and present results. This was a deliberate decision on the part of DARPA. DARPA stated: "DARPA developed a website devoted to providing information about the Grand Challenge... Interested participants and entrants used the website to communicate directly with DARPA. The website contained a discussion forum that participants used to share ideas about technical approaches for autonomous ground vehicles, including obstacle detection, navigation and position location, sensing, control software, and vehicle components." ([3], p. 3).

In general, this was a successful strategy. DARPA used the Internet effectively to communicate with teams and the public prior to the 2004 and 2005 GCE. However, the published record is rapidly disappearing. For example:

- DARPA made resources and references available to teams participating in the 2004 QID or GCE or 2005 GCE via the Grand Challenge Website, such as several versions of the 2004 GCE rules, a "description of the mandatory subjects to be addressed" in the team technical proposal, and the 2004 QID and GCE RDDFs. The Grand Challenge Website was substantially redesigned prior to the 2005 GCE. DARPA re-published portions of the Grand Challenge Website as the Archived Grand Challenge 2004 Website, but did not retain all published records. As a result, the Archived Grand Challenge 2004 Website is itself an incomplete record of events.

- Some teams which participated in the 2004 or 2005 GCE have since disappeared entirely from the Internet, leaving traces only in resources and references published by DARPA, or press about the Grand Challenge. Some of the teams which have since disappeared and which participated in the 2005 GCE did not publish their results via the Journal of Field Robotics. As a result, published records of their activity are practically non-existent.
- Some companies formed at the time of the 2004 and 2005 GCE to provide engineering or other services to teams participating in the 2004 QID or GCE or 2005 GCE have since disappeared.

At best, the Internet is an ephemeral resource. Future research which makes extensive use of the Internet should establish requirements for the maintenance of a permanent record of events as part of the published record.

In addition, DARPA established no requirement to publish in an academic journal or similar publication, and there is no evidence that DARPA required teams which participated in the 2004 and 2005 GCE to maintain records of their activities that would allow future researchers to re-construct team challenge vehicles. The author considers it likely other teams, in particular teams with a primary group identity of “Academic”, maintain repositories similar to the repository maintained by Team 2004-20, but these are of limited utility as they were not published.

DARPA intended team technical proposals to be the official published record of the 2004 and 2005 GCE. Prior to the 2004 QID or GCE, DARPA stated: “Publication of the technical completion of [*sic*] papers after completion of Challenge [*sic*] will ensure they become part of the legacy of this event. They will be the primary mechanism from which knowledge gained from this event is utilized in future research and development. The technical paper does not need to be so detailed that someone could immediately build the vehicle themselves, but it should be detailed enough to teach an interested individual about the design." ([69]). However, 2004 and 2005 team technical proposals provided insufficient technical detail and contained many errors, omissions, and inconsistencies which caused the author to conclude that they were unreliable as records let alone the "primary mechanism from which knowledge gained from this event is utilized in future research and development".

CHAPTER XII. RESEARCH METHODOLOGY

The foundation of much of this research is *critical scholarship*. To the extent possible, any conclusions presented by the author are supported by *objective evidence*. For the purposes of this research, objective evidence is considered to be that presented by primary public and academic sources which can be independently confirmed. These sources are referred to herein as “published records”. The complete body of published records is referred to herein as “the published record”.

To support objective, independent analysis, the author has attempted to separate the reputations of the universities and corporations involved from analysis where possible through the use of team numbers, in lieu of names, focus on participation in the 2004 and 2005 GCE in lieu of competition, and eliminate completely the use of informal testimony, or hearsay.

It is possible that participants in the 2004 and 2005 GCE are able to remember details and events which did not become part of the published record, and many of the teams which participated in the 2004 and 2005 GCE maintain websites providing points-of-contact through which the author could have solicited additional technical information or requested clarification of published records. However, the author determined that reliance on informal testimony or hearsay would introduce an additional element of uncertainty into what is already an uncertain record, and the decision was made early to rely on published records alone. As a result, no attempt was made to reconcile the published record with informal testimony or hearsay through email or telephone conversations with the teams.

The author does not consider manufacturer product literature to which access is directly controlled by the manufacturer or indirectly controlled by an agent of the manufacturer to be published records. Although the manufacturer may have a practice of granting access to product literature on a non-discriminatory basis, the manufacturer is in the sole position of being able to revise such literature without review. Although access to academic sources is similarly controlled, in general, publishers grant access to academic sources on a non-discriminatory basis, and academic sources are peer-reviewed. The author considers the scrutiny of peer review to be essential to the reliability of academic sources as published records. The lack of equivalent independent peer review of manufacturer product literature is a significant deficiency.

Where the author was unable to present adequate objective evidence, anecdotal evidence is presented, and is so noted.

In addition, from detailed review of technical guidance published by DARPA, technical proposals published by teams participating in the 2004 and 2005 GCE, and final published results, it is clear that published records are self-contradictory, provide incomplete or incorrect technical information, and do not provide enough information to answer key questions concerning team strategies during the 2004 and 2005 GCE, which would allow the author to independently assess the success of the DARPA Grand Challenge in one of its principal goals ([3], p. 2):

> Accelerate autonomous ground vehicle technology development in the United States in the areas of sensors, navigation, control algorithms, vehicle systems, and systems integration.

As a result, the decision was made early to reconcile published records with other published records where possible.

Since the conclusion of the 2007 Grand Challenge, the author has become aware of two additional sources of published records: a “privately compiled” collection of public domain files and documents ([70]) and a book about the Grand Challenge ([71]).

The publisher alternately stated the author of the collection ([70]) was the Department of Defense and: “Our news and educational discs are privately compiled collections of official public domain U.S. government files and documents - they are not produced by the federal government.” ([72]). The author concluded review of the collection, as a “privately compiled” collection of public domain files and documents, would not result in improvement in quality over the existing published record. As a result, the author did not review the collection.

Review of the table of contents for the book ([71]) hosted by an Internet retailer ([72]) indicates the articles published by the Journal of Field Robotics constitute the majority of source material. The author concluded review of the book would not result in improvement in quality over the existing published record. As a result, the author did not review the book.

The 2004 and 2005 GCE were highly publicized, and received a great deal of attention from the public. DARPA stated: “There was significant publicity as a result of the event, which increased the public’s awareness about the DoD desire to develop autonomous ground vehicles.” ([3], p. 9). DARPA continued with a detailed description of media coverage of the 2004 GCE.

The author was ultimately unable to determine whether the DARPA Grand Challenge was an engineering challenge or an exercise in public relations, and believes the evidence supports a conclusion that DARPA was unable to adequately determine what problem Grand Challenge participants were being asked to solve because the difference between the stated goal of the Grand Challenge and actual goal of the Grand Challenge resulted in proposed solutions which did not result in significant progress toward the actual goal of the Grand Challenge. Offered solutions were too expensive, and improvement in challenge vehicle average speed was more a result of improvements in processing speed due to Moore's Law than any other factor.

As a result of the emphasis on public relations, DARPA made several unfortunate decisions concerning team participation. As a result of the enormity of the problem domain, teams did not have enough time to fully document their efforts, or complete all planned work or testing. Consequently, the overall quality of published records is low.

In addition, the precise definition of the Grand Challenge as a system integration exercise which required some expertise in the area of artificial intelligence applied to autonomous ground vehicle navigation was concealed by the format of the Grand Challenge as a race. Yet the results of the 2004 and 2005 GCE confirm this conclusion. The teams with the most experience in the problem domain were more successful, not because they were better able to code an artificial intelligence, but because they more quickly realized the limits of their sensors and computing equipment, and were able to optimize their solution to make full use of limited sensor technology.

In addition, if an unstated goal of DARPA was to “seed” industry with graduates with experience in autonomous vehicle development, it was a failure. The Grand Challenge was not designed to reward or even emphasize the most important skill: competent system integration at a reasonable procurement cost.

Team 2005-12, for example, successfully completed[19] the major portion of the 2005 GCE course several weeks following the 2005 GCE, after having corrected the programming error responsible for failure to complete the course during the 2005 GCE. Team 2005-12 provided the following account ([74]):

> Early Monday morning, October 31, 2005, ironically Halloween, we set out to run the 2005 Grand Challenge course exactly as we did during the actual Grand Challenge. [The challenge vehicle] was using the same RDDF (file of GPS waypoints that define the course) and the same global constraints and control coefficients. The only substantive difference was the change in the "one line of code"...
>
> Launch came at PST and was uneventful. Everything was perfect until just miles into the course when a mirage seemed to appear in the distance. Not to worry, it's the desert; however, it quickly became apparent that the "dry" lake was not so dry. It had rained since the Grand Challenge and the course was not traversable in a non-amphibious vehicle. The decision was to

cease autonomous operation in order to not lose the vehicle. A precise autonomous run of the 2005 GC course was infeasible because of the rain. With the current condition, no Grand Challenge vehicle could have made it beyond this point. In fact, if this condition would have existed during the Grand Challenge, DARPA would have altered the course. It now became evident why, during the Grand Challenge, the course was not divulged earlier than 2 hours before the race. I [*sic*] was to ensure that the course was a fair one and that some environmental condition had not made a part of the course impassable.

Rather than go home, the decision was to continue to uncover [the challenge vehicle's] autonomous operational limits by continuing on the traversable portions of the 2005 GC course. The first limit had been established: it can’t traverse lakes and isn’t smart enough to figure out a way around them, if the “desired” course is through them. That’s the first thing that was discovered that we need to work on.

After a brief diversion around the lake, autonomous operation was reinitiated at reemergence of the 2005 GC course. This incident made it apparent that two people were needed inside the vehicle to properly monitor the road ahead. Other than the lake situation (which occurred at 2 other points), the only non-autonomous diversions were due to

1. places where the "road" had been "bulldozed" probably to discourage exactly what we were trying to do. These places existed at XXXX and XXXX, and

2. on XXXX a public road, where we pulled over to let a cement truck pass us (if this situation would have occurred during the Challenge, DARPA would have paused the vehicle and instructed the cement truck to carefully pass the vehicle).

These two incidents refine the operational limits that need to be worked on. Specifically, [the challenge vehicle] needs the capacity to be able to violate its desired route constraints and set out to find any feasible path ahead. At present, it does not have this capability.

Also, [the challenge vehicle] was paused several times, much the same way that DARPA may have legitimately paused the vehicle during the Grand Challenge. Pauses were instituted prior to crossing public roads, the Union Pacific at-grade crossing, upon encountering closed gates, that once opened, were negotiated autonomously and for preparing the onboard camera to record the traverse of Beer Bottle Pass at night.

Except for the above constraints, none of which existed during the Grand Challenge, [the challenge vehicle] autonomously traversed the course. No changes, corrections or alterations were made to any of [the challenge vehicle's] autonomous systems. It can be argued that [the challenge vehicle] autonomously traversed an even more challenging course than that of the 2005 Grand Challenge. Except for the two lakes and the two "bulldozed" areas, [the challenge vehicle] was autonomous, including places where the road was significantly rougher than what existed in early October.

This accomplishment is significant because Team 2005-12 is the only team known to have completed the 2005 GCE course, as described above, using only a STEREO sensor: one Point Grey Bumblebee stereo camera pair. No other environment sensors were in use by Team 2005-12. As a result, the author considers Team 2005-12 to be the most successful potentially-disruptive team to have participated in the 2005 GCE.

The most successful team overall, Team 2005-06, was not declared the winner of the 2005 GCE. This was because the fundamental problem of the Grand Challenge favored teams with significant experience and sponsorship. The utility of technical solutions proposed by other successful teams is suspect. It is unreasonable to expect the DOD to pay for a sensor package which exceeds a significant portion of the cost of the vehicle on which it installed. DARPA did not establish a relative weighting scheme which would allow challenge vehicle performance to be directly compared, and the published record is utterly inadequate to the task.

The use of simulation, including the development and application of standard reference terrain and standard problems, would provide a framework for evaluating the application of artificial intelligence to autonomous ground vehicle navigation free of the distraction of system integration problems which plagued teams participating in both the 2004 and 2005 GCE. As a result, the emphasis on artificial intelligence would be restored.

Teams participating in the Grand Challenge should first have been required to implement a challenge vehicle in simulation. This would minimize real cost to the teams. In addition, some team programming hours would have been focused on improvements to the simulation environment.

The development and testing of a challenge vehicle should have been an iterative process, first of "tuning" the simulation environment to accurately model real world interaction, then increasing the difficulty and duration of field testing of team challenge vehicles via a series of challenges, moving from concept to an actual prototype and culminating in a 2004 or 2005 GCE-like event. This would have resulted in the development of a simulation environment which would have made it possible to fully separate the development of artificial intelligence applied to autonomous ground vehicle navigation from the system integration portion of the Grand Challenge, allowing continued participation by teams lacking the resources of some teams participating in the 2004 or 2005 GCE.

DARPA's selection of teams to continue to field testing should have been made on the basis of the performance of team implementation of a challenge vehicle controlling intelligence in simulation when compared to the real world. Field testing should have been accompanied by a requirement that teams participating in the Grand Challenge deliver periodic updates documenting the results of test and evaluation, culminating in an event similar to the 2004 or 2005 GCE.

In addition, teams participating in the Grand Challenge should have been provided a budget, required to follow basic accounting rules, and accounted for their expenses via the published record. This would have helped "level the playing field" by mitigating the advantage of teams with significant sponsorship.

Table III. Team reference numbers[a] .

Team name[20]	**2004**	**2005**
A. I. Motorvators	01	
Axion Racing	02	01
The Blue Team	03	
Center for Intelligent Machines and Robotics (CIMAR)	04	02
CyberRider	05	
Digital Audio Drive (Team DAD)	06	03
Desert Buckeyes		04
The Golem Group (2004) / The Golem Group/UCLA (2005)	07	05
The Gray Team[b]		06
Insight Racing	08	07
Intelligent Vehicle Safety Systems I		08
Mitre Meteorites		09
Mojavaton		10
MonsterMoto		11
Princeton University		12
Palos Verdes High School Warriors	09	
Red Team	10	13
Red Team Too		14
Rob Meyer Productions	11	
Rover Systems	12	
SciAutonics I (2004) / SciAutonics/Auburn Engineering (2005)	13	15
SciAutonics II	14	
Stanford Racing Team		16
Team Arctic Tortoise	15	
Team Cajunbot	16	17
Team Caltech	17	18
Team Cornell		19
Team ENSCO	18	20
Team LoGHIQ	19	

Team Overbot	20	
Team Phantasm	21	
Team Spirit of Las Vegas	22	
Team TerraMax	23	21
Terra Engineering	24	
Virginia Tech (2004) / Virginia Tech Grand Challenge Team (2005)	25	22
Virginia Tech Team Rocky		23

[a] Teams will be referred to by the unique combination of year and identifier. For example, Axion Racing is referred to herein as “Team 2004-02” for the 2004 QID and GCE, and “Team 2005-01” for the 2005 GCE.

[b] The title of the technical proposal hosted by the archived Grand Challenge 2005 website ([77]) is “GreyTeam.pdf”, although all other references are to “The Gray Team” or “Gray Team”. The team's preferred spelling is used herein.

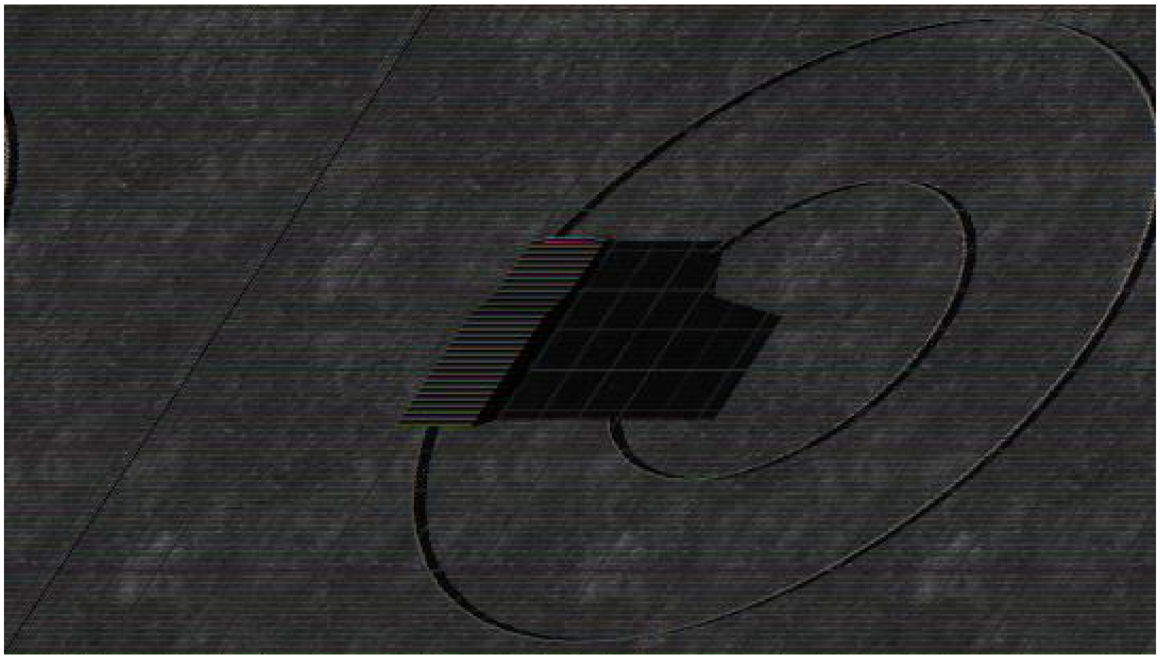

Figure 2. Example image saved by Gazebo.

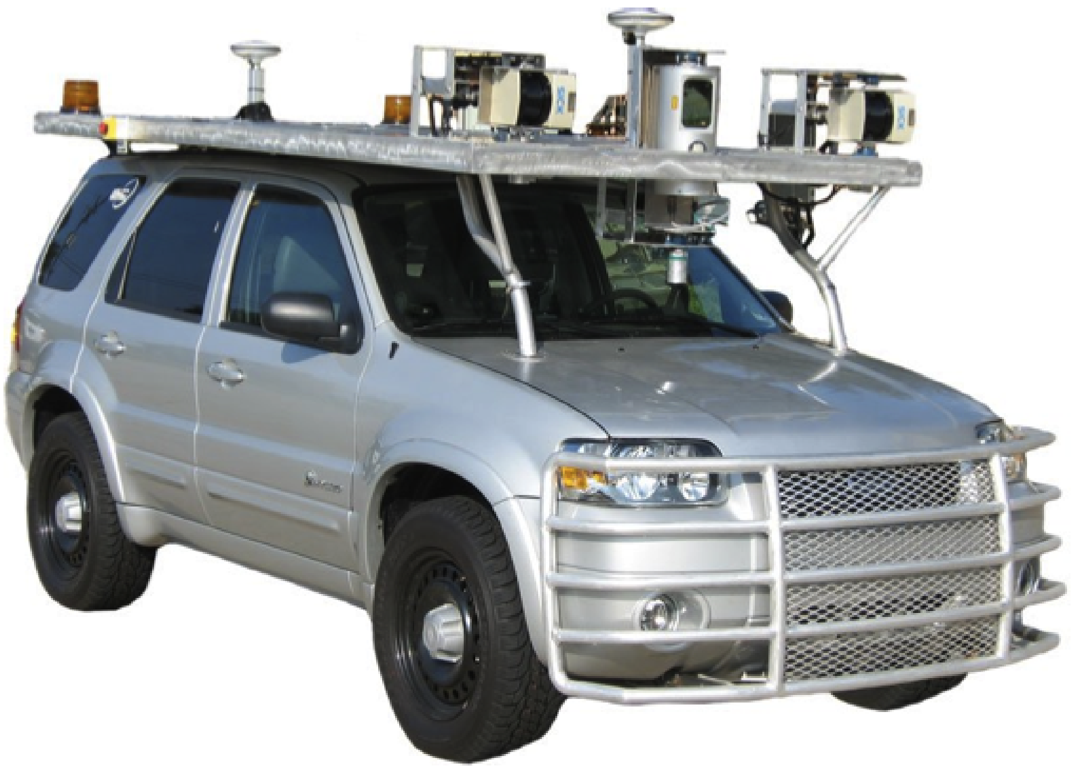
Figure 3. Team 2005-06 challenge vehicle.

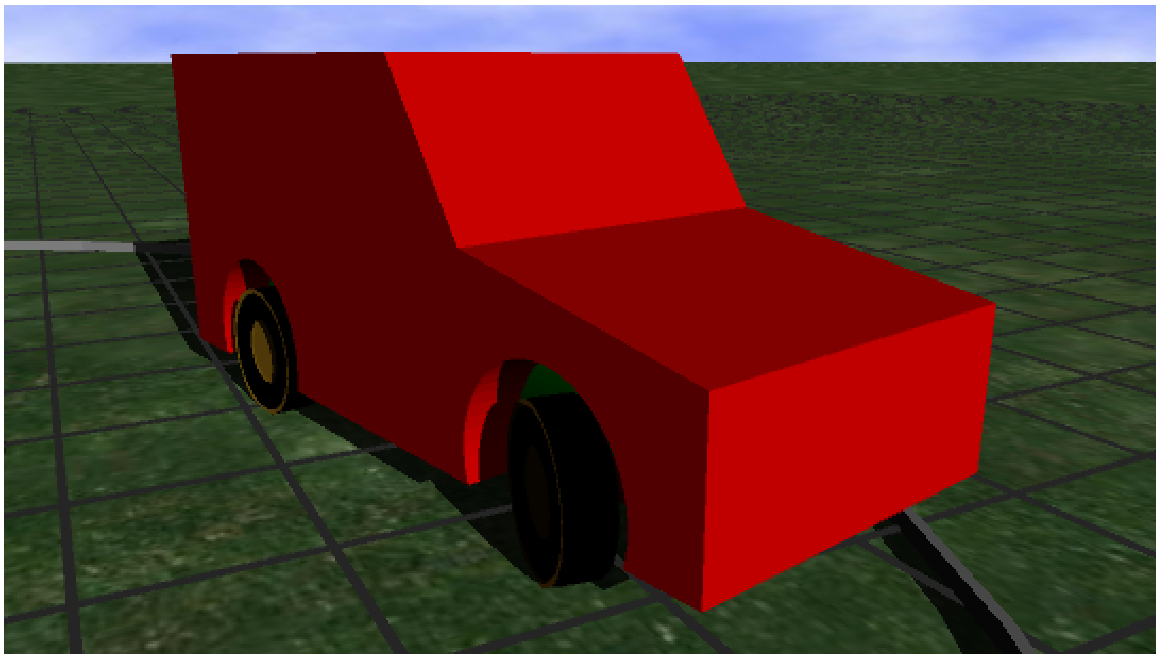

Figure 4. Model of the representative challenge vehicle.

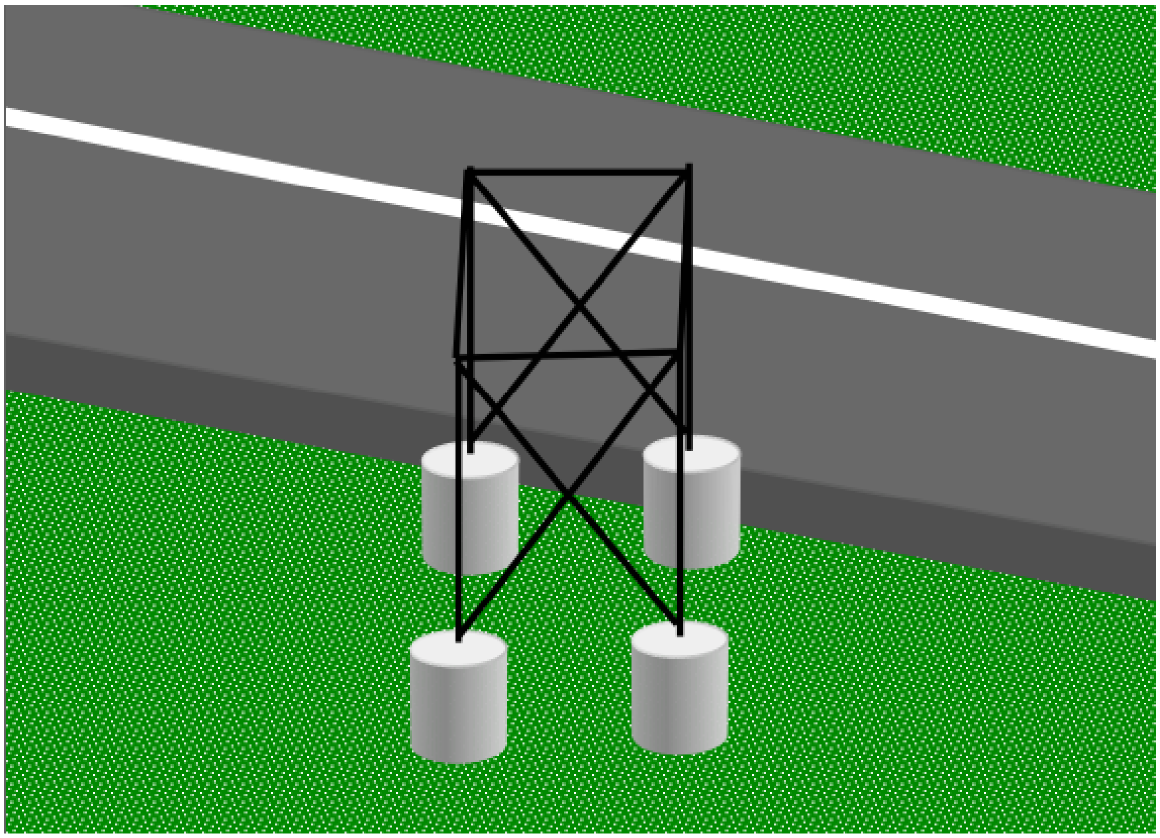

Figure 5. Tower obstacle (DARPA description).

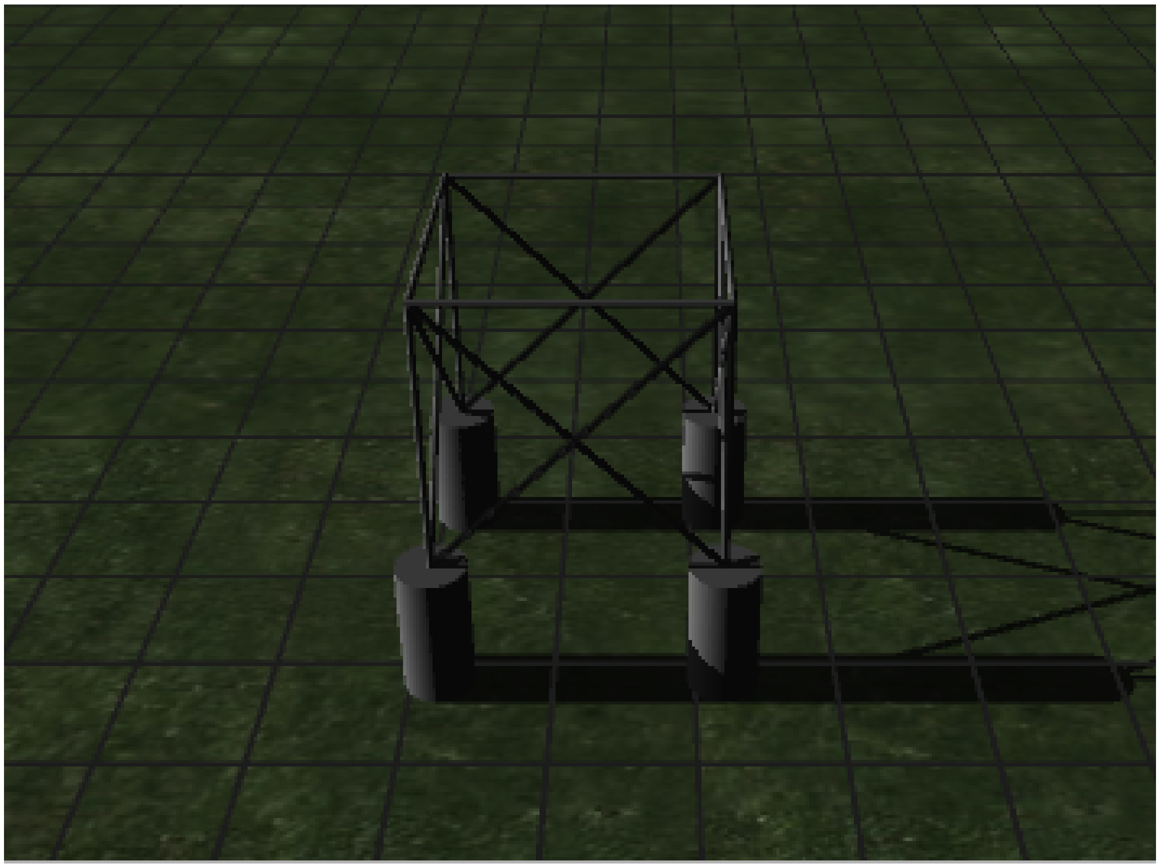

Figure 6. Tower obstacle (Gazebo model).

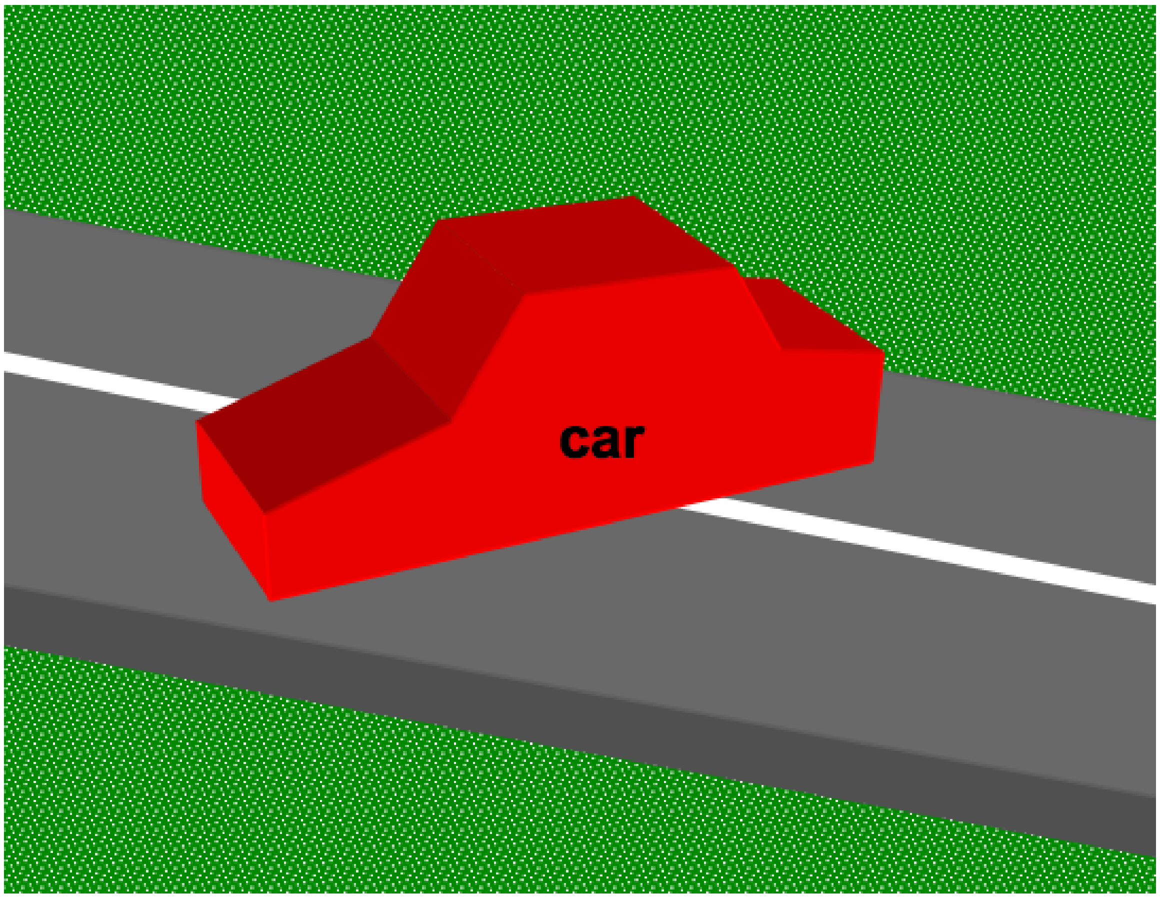

Figure 7. Car obstacle (DARPA description).

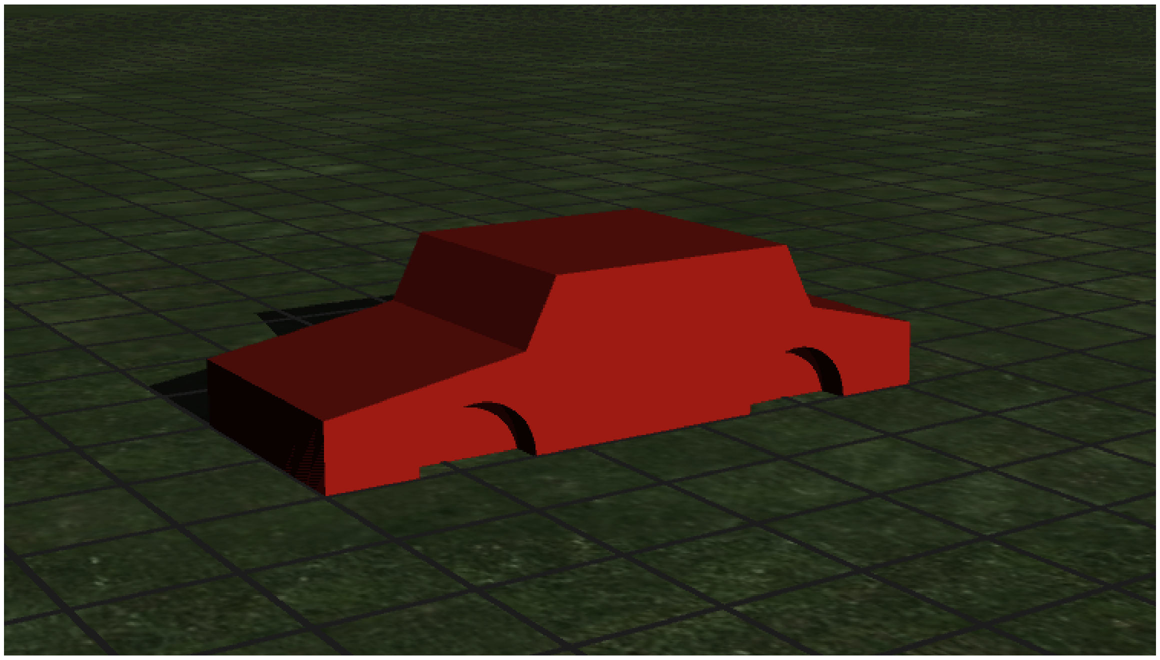

Figure 8. Car obstacle (Gazebo model).

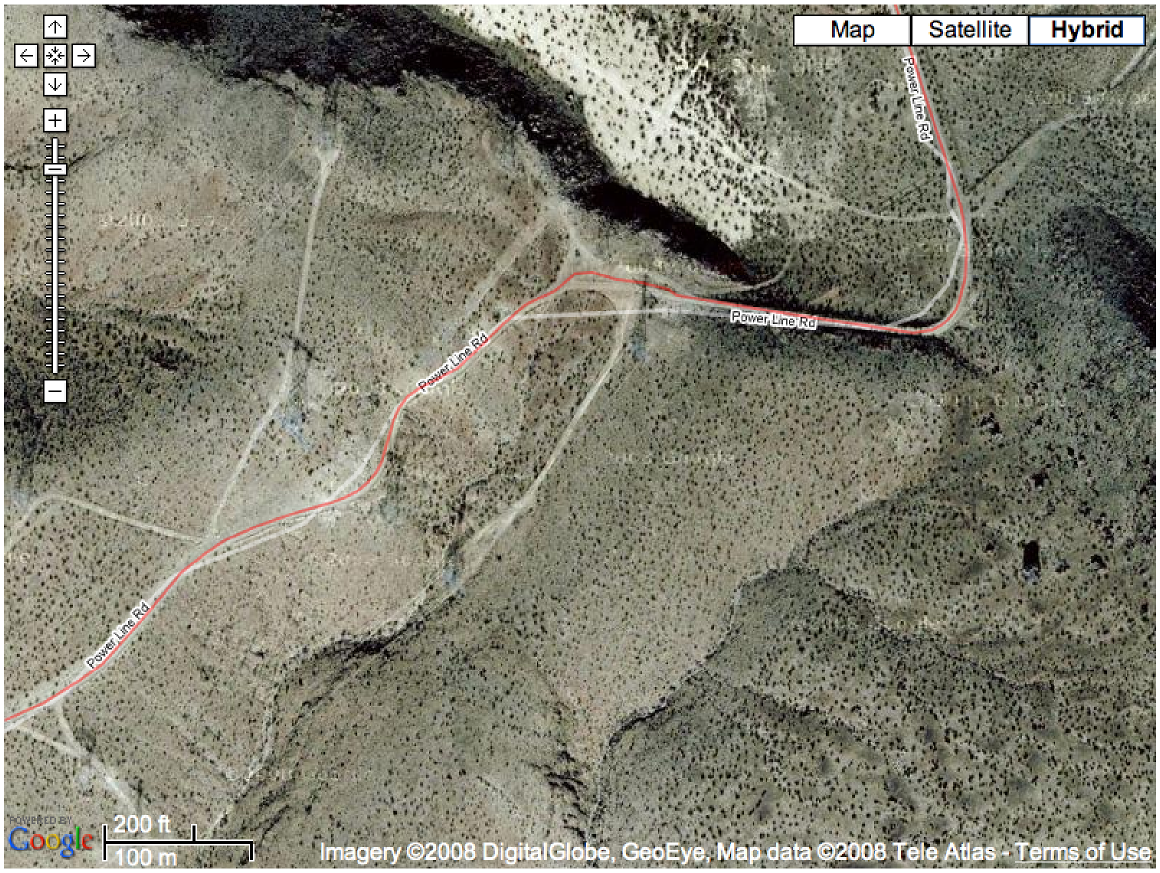

Figure 9. 2004 GCE course superimposed on map and satellite view (Powerline Road 1).

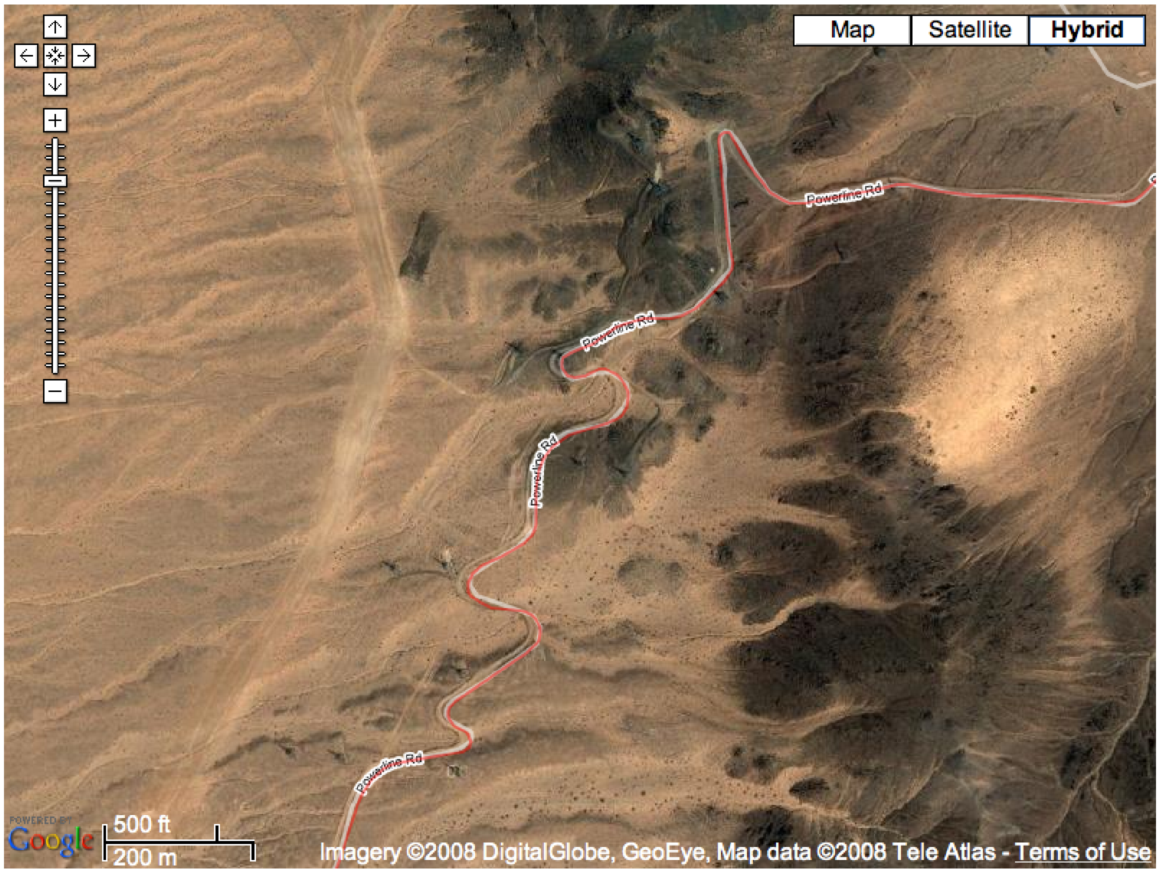

Figure 10. 2004 GCE course superimposed on map and satellite view (Powerline Road 2).

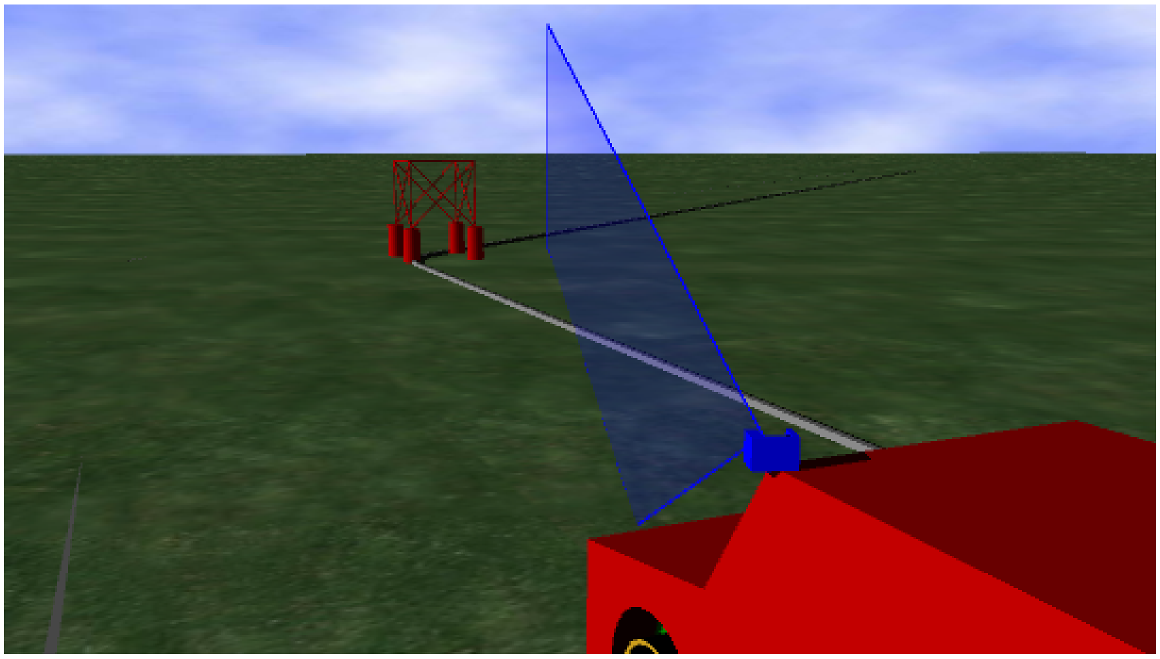

Figure 11. Vertically-aligned LIDAR sensor configuration.

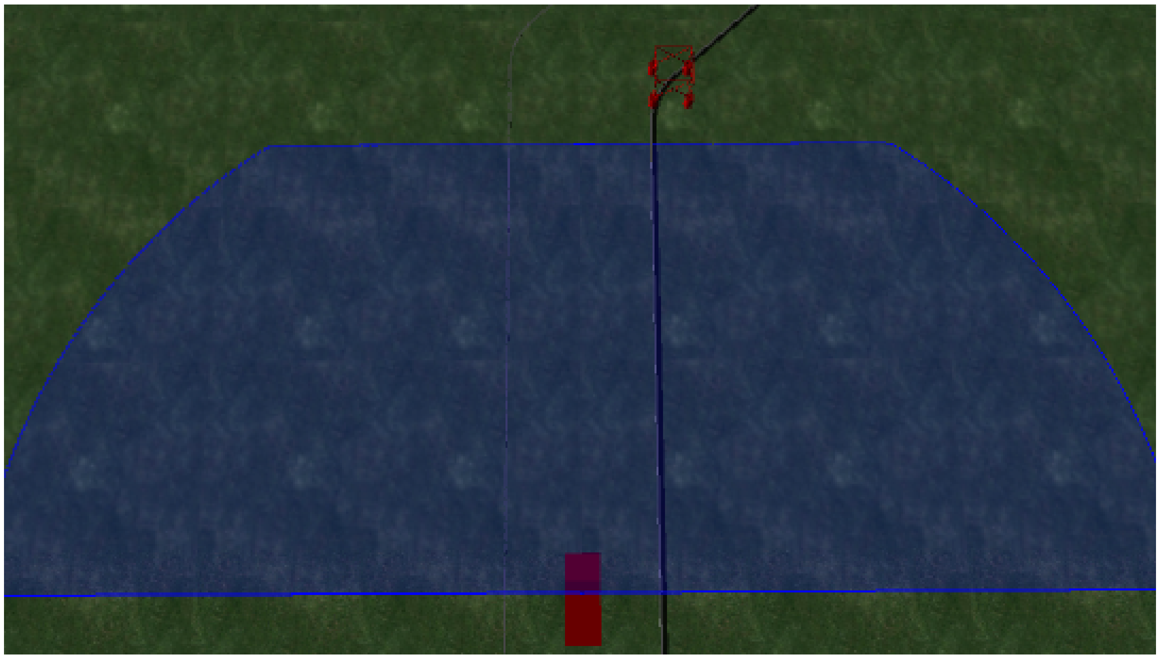

Figure 12. Horizontally-aligned LIDAR sensor configuration (overhead view).

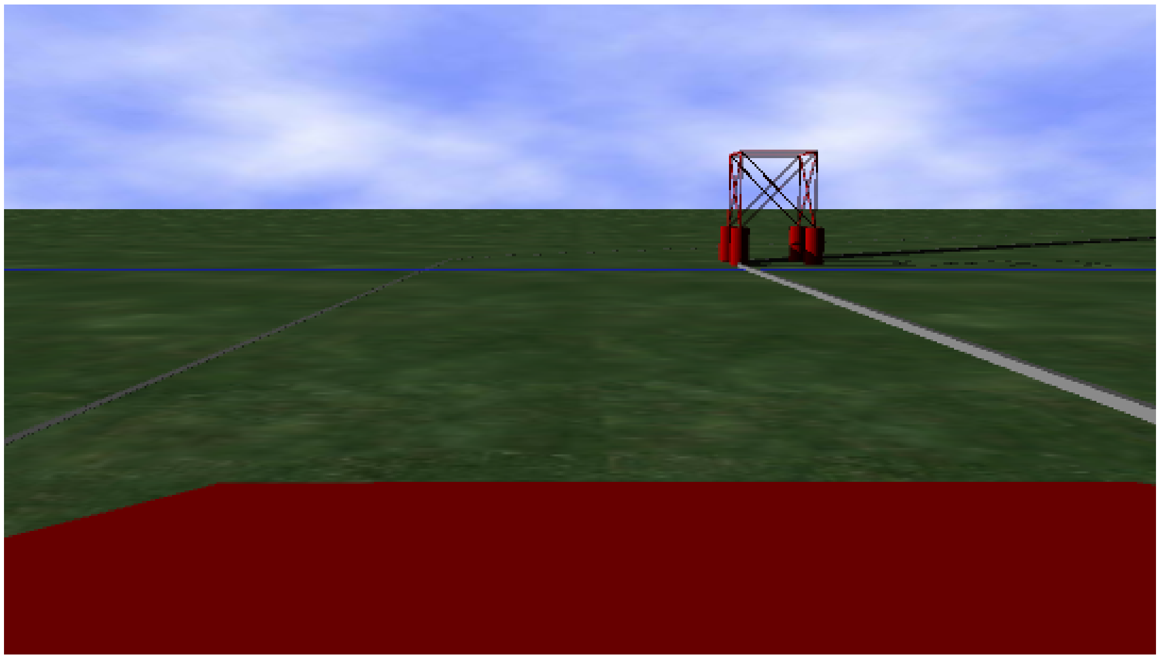

Figure 13. Horizontally-aligned LIDAR sensor configuration (driver's seat view).

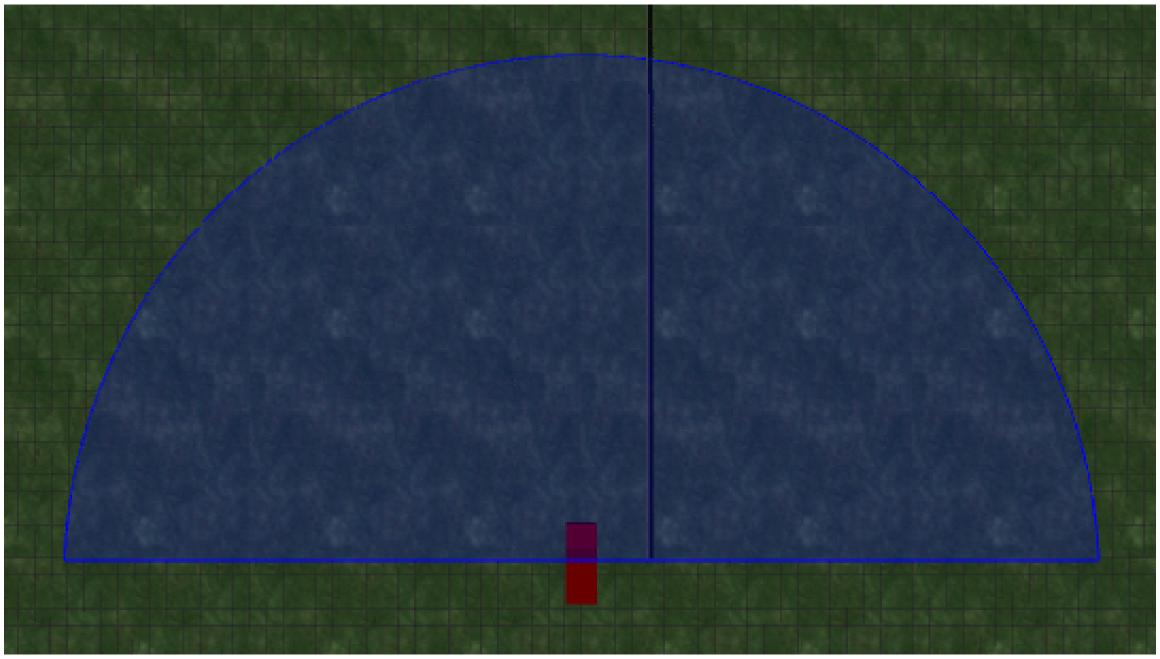

Figure 14. Horizontally-aligned LIDAR sensor configuration with a down angle of 4 degrees (overhead view).

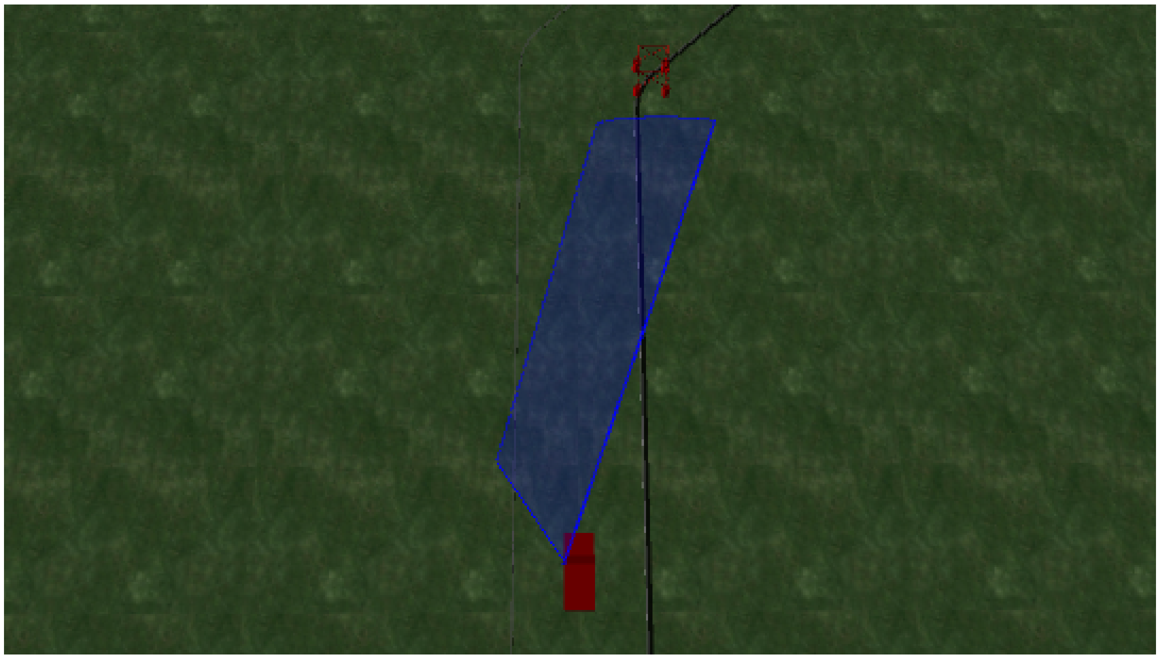

Figure 15. Diagonally-aligned LIDAR sensor configuration (overhead view).

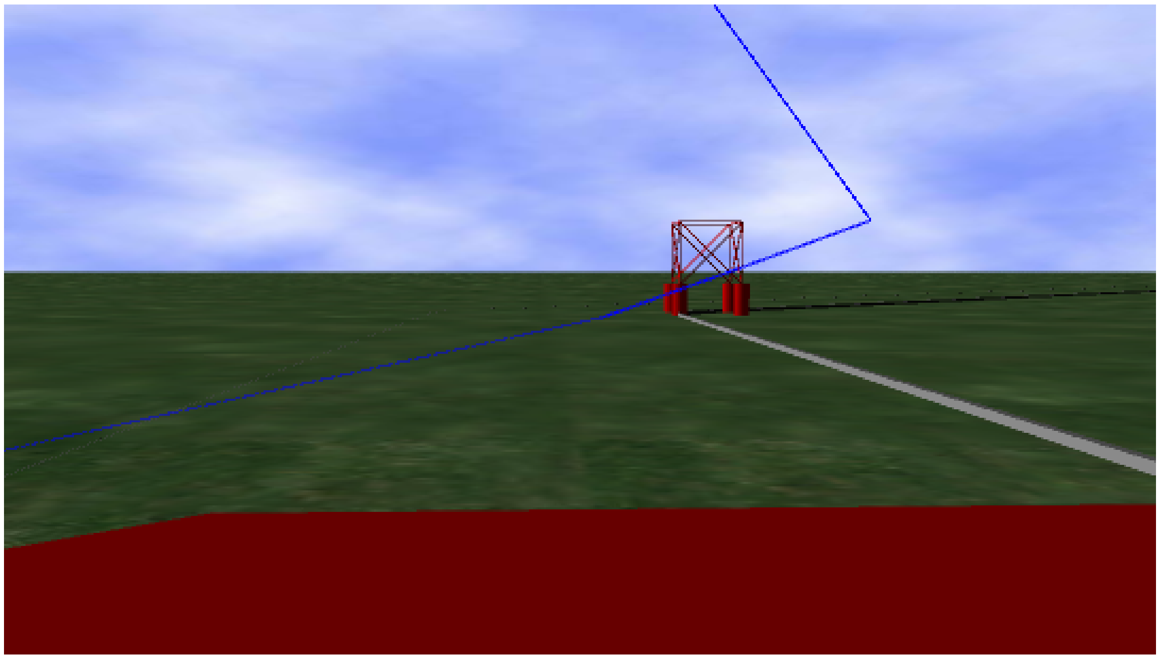

Figure 16. Diagonally-aligned LIDAR sensor configuration (driver's seat view).

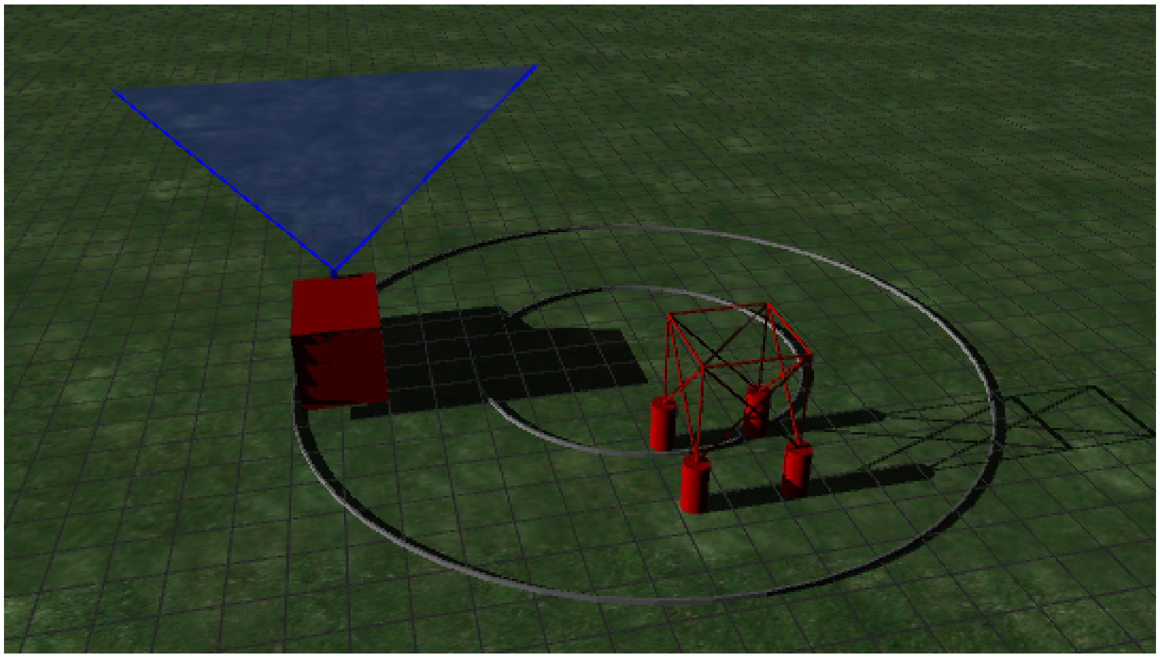

Figure 17. Epsilon Lambda ELSC71-1A RADAR field-of-view limitation (simulation initial state).

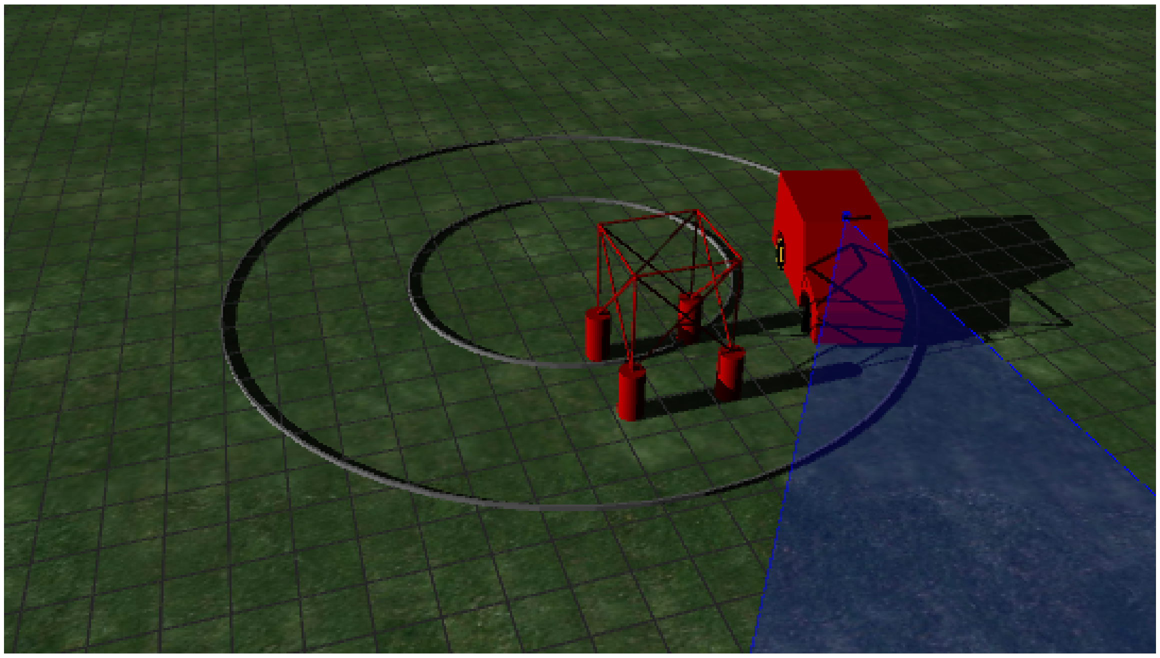

Figure 18. Epsilon Lambda ELSC71-1A RADAR field-of-view limitation (showing the sensor cannot detect an obstacle in the path of travel).

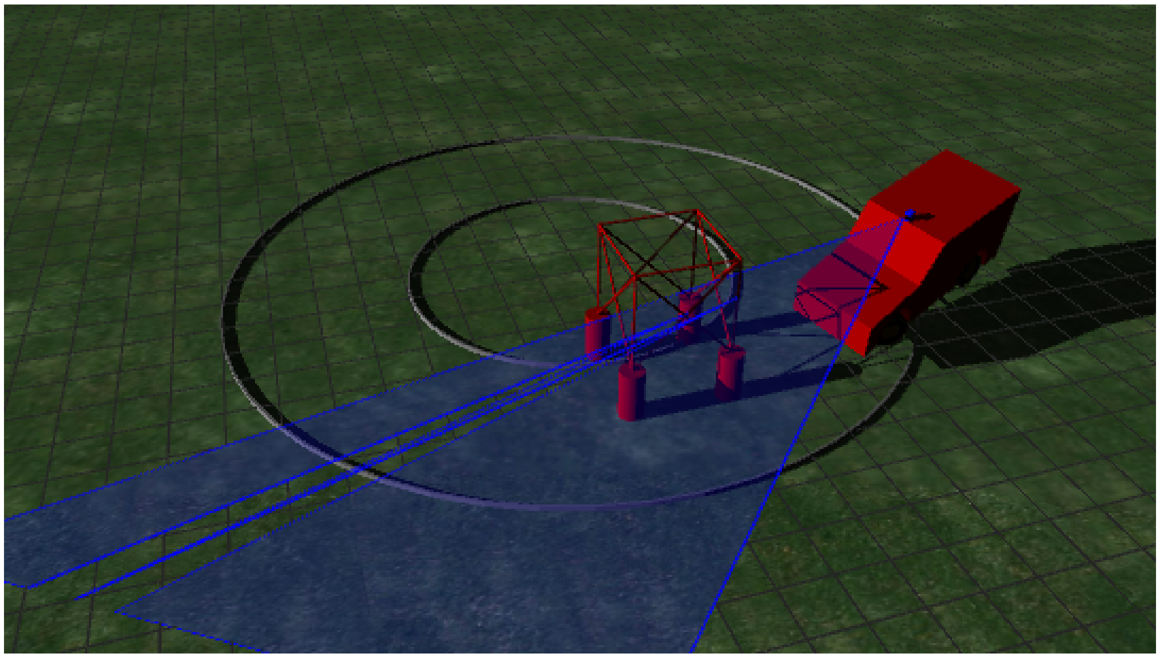

Figure 19. Epsilon Lambda ELSC71-1A RADAR field-of-view limitation (after collision with the tower obstacle).

Appendix A: Development of the Installation Procedure

CHAPTER I. DEFINITIONS AND CONVENTIONS

Throughout the discussion that follows:

- “Player” refers to the Player server.
- “Gazebo” refers to the Gazebo high-fidelity robot simulator.
- The versions of relevant applications and source are documented herein.
- When a specific version of a documented or undocumented dependency was required by installation instructions, that version is documented herein. If no specific version was required, “-none-” is recorded.
- Development (header and library) files for documented or undocumented dependencies were installed where available.

CHAPTER II. METHODOLOGY

II.A. Use current, stable, and release versions of applications and source

Because the potential simulation targets required unknown potential changes to Player, Gazebo, and ODE, the author developed an installation procedure to ensure a reproducible simulation environment and establish a reliable baseline from which to proceed with any changes.

Ideally, the author would have used versions of applications and source available during the 2004 or 2005 GCE. With a few exceptions, they are no longer available. Although the author has versions of Player and Gazebo dating from 2004 and 2005, specifically Player 1.6 and Gazebo 0.5.1, the author concluded attempting to base the simulation environment on these versions of Player and Gazebo would ultimately be unproductive due to problems encountered during the development of the installation procedure.

As a result, the simulation environment was based on current, stable, release versions of applications and source, with one exception: Gazebo. As noted, ideally the author would have used versions of applications and source available during the 2004 and 2005 GCE. However, the author asserts an increase in processing power was a key factor during the 2005 GCE, and contends that, if ten percent of the development cost of team challenge vehicles had been invested in improvements to the simulation environment, resulting in more rapid development, the same increase in processing power may have been a key factor during the 2005 GCE by allowing the use of more realistic simulation, for example, to compare the performance and capabilities of various SICK LIDAR sensors or to effectively visualize the interaction of the challenge vehicle with the environment.

The author concluded an exception must be made for Gazebo when repeated attempts to compile the current, stable, release version of Gazebo ("gazebo-0.9.0.tar.bz2") failed due to "undefined reference" errors to "FreeImage_Rescale" and "FreeImage_ConvertFromRawBits".

These errors also occurred during the initial attempt to develop the installation procedure (herein "initial attempt"). Although the author concluded the problem was caused by linking errors, and most probably by a missing or inaccessible (for whatever reason) library, attempts to manually compile past this point, modify configuration files, and engage the community directly were unsuccessful. The author was unable to resolve the errors and eventually abandoned the initial attempt. Similar errors occurred with OpenGL and FFmpeg (specifically libavcodec) during the initial attempt, which were successfully resolved.

As a result, the author downloaded the latest revision (revision 8443) of the Gazebo 0.9.0 source code using `svn`:

```
$ svn co https://playerstage.svn.sourceforge.net/
svnroot/playerstage/code/gazebo/trunk gazebo
```

II.B. Use documented installation instructions, when available

Based on comments made by teams participating in the 2004 and 2005 GCE, the author concluded most teams wanting to use Player and Gazebo as a simulation environment would not have had the time required to troubleshoot an installation procedure, and would have relied on documented installation instructions, where available. As a result, that author installed applications or source code in accordance with installation instructions documented by the application's developer(s) or through the use of an automated tool when possible.

As used herein, “documented installation instructions” means installation instructions included with applications and source code (“packaged documentation”) or installation instructions available through online documentation (“online documentation”). When both packaged documentation and online documentation was available, the author favored online documentation because he believed it would be more up-to-date than packaged documentation. As noted throughout the paragraphs that follow, this was true for some applications but not others.

Reliance on documented installation instructions caused several problems:

- No comprehensive installation procedure was available for Gazebo. Gazebo required several third-party libraries and the lack of a comprehensive installation procedure was one of the two significant problems resolved while developing this installation procedure. [78] published an alternate comprehensive installation procedure based on Fedora 9, in lieu of openSUSE 11.2, and earlier versions of Player, Gazebo, and their dependencies. See paragraph II.D.
- Installation instructions for some applications were incomplete.
- Installation instructions for some applications were incorrect.
- No reliable list of dependencies was available for some applications, and, in fact, the definition of a “dependency” varied from application to application.

Configuration of some packages failed because "optional" libraries were not installed. This was the cause of one of the two significant problems resolved while developing this installation procedure. See paragraphs II.C.1. and III.G.4.

- There were conflicts between packaged documentation and online documentation which had to be resolved on a case basis.

II.C. Troubleshoot the installation procedure

The initial attempt to develop an installation procedure for the simulation environment failed. Attempting to resolve errors in documentation, in particular inadequately documented dependencies and conflicting installation instructions, consumed several weeks during which more productive research was delayed.

Problems encountered during the initial attempt and later successful attempt to develop an installation procedure, and their resolutions, are documented herein. Because the author revised the order of installation, used a later revision of Gazebo, and maintained inadequate records of the initial attempt, problems encountered during the initial attempt may not be reproducible using this order of installation and revision of Gazebo.

However, the author developed a method for troubleshooting the installation procedure as a result of experimentation and review of packaged and online documentation which was used to develop the installation procedure:

II.C.1. "Optional" libraries

During the initial attempt, configuration of some applications and source failed because "optional" libraries were not installed. Because configuration should not have failed because optional libraries were not installed, the author considered these undocumented dependencies, and resolved the problem when it occurred by installing the undocumented dependency.

However, due to problems encountered during the initial attempt ordering the installation procedure and determining which applications and source were true dependencies, the author decided to limit the use of “optional” libraries to simplify the installation procedure to the extent possible during the development of the installation procedure. Any resulting errors were dispositioned on a case basis as either configuration errors or evidence of undocumented dependencies. In general, configuration errors were resolved by disabling some feature during configuration and undocumented dependencies were resolved by installing the undocumented dependency. Resolution is documented throughout this Appendix.

II.C.2. Order the installation procedure

Because of inadequately documented dependencies, the author used 5” X 8” index cards to record application name, package or source file name, documented dependencies, undocumented dependencies as they were noted, specific configuration options, installation instructions used to compile applications from packages or source, and any information necessary to troubleshoot installation for each application. This allowed the author to easily re-order the steps of the installation procedure as necessary.

II.C.3. Maintain a record of errors encountered

The initial attempt ultimately failed when the author was unable to resolve many “undefined reference” errors to OpenGL, avcodec, and FreeImage functions while attempting to build the Gazebo executable. The specific step during which the "undefined reference" errors occurred was "Linking CXX executable gazebo".

Attempts to determine the exact cause of the errors by reviewing the Gazebo mailing list archives were unsuccessful. An attempt by the author to solicit help by engaging the Player Project's community more directly was also unsuccessful. A reply to a post to the playerstage-gazebo mailing list on 25 September 2009 stated: "The latest SVN version of Gazebo should have this problem fixed." However, the author was unable to determine the cause of the problem from the response and the problem was not resolved. Attempts to compile the "latest SVN version" of Gazebo failed with the same errors.

While attempting to resolve the errors, openSUSE 11.2 was released. The author upgraded to openSUSE 11.2 from 11.1, and subsequently developed the installation procedure documented by this Appendix. The initial attempt was not preserved, and the author retained only a few records of errors encountered while attempting to build a reproducible simulation environment. The author concluded this was a mistake which made it more difficult to troubleshoot specific errors encountered during the initial attempt because it was impossible to determine the effects of specific actions taken to identify or resolve errors without being able to compare output from one attempted solution to the next.

This resulted in a change in methodology. During the development of this installation procedure, standard output and standard error from `./bootstrap`, `./configure`, `make`, `cmake`, and `make install` commands were re-directed to files to document any errors and their successful resolution.

II.D. Comparison between the installation procedure and alternate installation procedures

II.D.1. First alternate comprehensive procedure

[78] published an alternate comprehensive installation procedure ("alternate procedure") for Gazebo 0.7.0 and 0.8.0, which was based on [79], and which pre-dates [80] and [81]. The alternate procedure for Gazebo 0.8.0 ([82]) was based on Fedora 9, in lieu of openSUSE 11.2. Based on the initial attempt, the author determined there were several additional problems with the use of the alternate procedure for Gazebo 0.8.0, in addition to those noted above:

II.D.1.a. Problem: The alternate procedure refers, incorrectly, to the NVIDIA Cg library as an "OGRE dependency"

[82] states: "OGRE dependency: nVidia Cg Toolkit". The author considers this to be additional evidence supporting the author's assertion that the definition of "dependency" varies from application to application, which is discussed in more detail throughout this Appendix.

II.D.1.b. Resolution: Re-evaluate the installation of Cg

The author re-evaluated the installation of the NVIDIA Cg library ("Cg"), and revised the installation procedure to include step "Install Cg". See paragraph III.H.

II.D.1.c. Problem: The alternate procedure does not require the installation of CEGUI

[82] does not require the installation of CEGUI.

II.D.1.d. Resolution: Re-evaluate the installation of CEGUI

The author determined the only purpose of CEGUI in the installation procedure was to provide support for OGRE demos. The successful installation of Gazebo was a research objective. The author's only interest in OGRE was as a dependency for Gazebo. As a result, the author re-evaluated the installation of CEGUI and revised the installation procedure to delete step "Install CEGUI". See paragraph III.G.4.

II.D.1.e. Confirm path environment variables

After the author successfully installed Gazebo using this installation procedure, the first attempt to run Gazebo resulted in the following error:

```
error while loading shared libraries:
libavformat.so.52
```

This was similar to errors encountered during the initial attempt. The command:

```
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
```

appeared to resolve the error, but the author did not modify `.bashrc`, `.profile`, or any variants thereof and subsequent attempts to run `gazebo <worldfile>` from a new shell were successful without exporting `LD_LIBRARY_PATH`. This error did not recur during the verification of the installation procedure. As a result, this error was not reproducible.

However, as a result of this error and similar errors encountered during the initial attempt, the author re-evaluated the need to either confirm the following path environment variables include the following paths prior to verification of the installation procedure, or export them as necessary, as noted by the alternate procedure:

```
export PATH=/usr/local/bin:$PATH
export CPATH=/usr/local/include:$CPATH
export LIBRARY_PATH=/usr/local/lib:$LIBRARY_PATH
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:
$PKG_CONFIG_PATH
```

The author revised the installation procedure to include step "Path environment variables".

II.D.2. Second alternate installation procedure

Shortly after verification of the installation procedure in accordance with Appendix C, the author located a second alternate installation procedure available via [38] while reviewing documentation for Gazebo world files and Player configuration files. The installation instructions are hosted on the project wiki, which provides a kind of version control documented by the page "history". Review of the history indicates an earlier version of the second alternate installation procedure was available while the author was attempting to develop the installation procedure documented in Appendix B in accordance with this Appendix. As a result, the earlier revision ([83], which is dated March 3, 2009) is referred to as the "second alternate installation procedure" herein. The later revision of this procedure ([84], which is dated January 12, 2010) is referred to as the "revised second alternate installation procedure" herein.

The second installation procedure identified the following documented dependencies:

- `pkg-config`
- `zziplib`

- FreeImage
- OGRE
- ODE

The author noted the following problems with the second installation procedure. The second installation procedure:

- provides no installation instructions for Gazebo
- does not refer to the installation of development (header and library) files, for example `zziplib-devel` for OGRE and `libxml2-devel` for Gazebo
- does not identify compatible versions of documented dependencies
- does not identify `libxml2` as a dependency
- does not identify OIS as a dependency
- does not identify CEGUI as a dependency
- does not identify Cg as a dependency, although it does recommend the use of the `--disable-cg` flag to disable building Cg support when configuring OGRE
- does not require Player to be installed before Gazebo

The revised second installation procedure identified the following documented dependencies:

- FreeImage >= 3.10
- OGRE 1.70
- ODE 0.11.1

- boost >= 1.35
- FLTK 1.1

The revised second installation procedure provided installation instructions for Gazebo, conforming in general to [81], and includes a section titled "Troubleshooting the Install". The author noted the following problems with the revised second installation procedure. The revised second installation procedure:

- does not identify `zziplib` as a documented dependency of Gazebo
- does not refer to the installation of development (header and library) files, for example `zziplib-devel` for OGRE and `libxml2-devel` for Gazebo
- does not identify `libxml2` as a dependency
- does not identify OIS as a dependency
- does not identify CEGUI as a dependency, and does not recommend the use of the `--disable-ogre-demos` flag to force `./configure` to continue without building the OGRE demos when configuring OGRE
- does not identify Cg as a dependency, and does not require the use of the `--disable-cg` flag to disable building Cg support when configuring OGRE
- does not require Player to be installed before Gazebo

Section "Troubleshooting the Install" states, in part: "If gazebo doesn't compile... Check the output of the configure step. Resolve all errors by installing the necessary 3rd party packages." However, as noted throughout this Appendix, during the initial attempt, configuration of some applications and source failed because optional libraries were not installed. Although installation of Gazebo did not result in any configuration errors due to optional libraries, installation of OGRE, which is a documented dependency of Gazebo, failed due to configuration errors. These errors initially caused the author to install CEGUI, leading to one of the two significant problems encountered.

Section “Troubleshooting the Install” also states, in part: “If gazebo doesn't compile... Make sure that the 3rd party packages are the correct versions.” However, installation of Gazebo failed because a later version of a documented dependency was used (OGRE version 1.6.4 was used in lieu of 1.6.3 during the initial attempt). In addition, to configure Gazebo a later version of a documented dependency was required than was documented (OGRE version 1.6.3 in lieu of 1.4.4. See paragraph III.L.2.a.).

II.D.2.a. <u>Conformance of the installation procedure to the second alternate installation procedure</u>

Aside from the use of `scons`, which was the build system used by Gazebo 0.8.0, the alternate procedure generally conforms to the installation procedure developed by the author, with the following exceptions:

- The order of installation of all applications and source is similar but not identical. However, the order of installation of the group of OGRE dependencies is relatively unimportant since they are not interdependent unless CEGUI is also installed. As a result, the true order of installation is: OGRE dependencies, OGRE, Player, and Gazebo. ODE, having no dependencies other those provided by the base installation, may be installed at any time prior to Gazebo.

- The names and versions of installed applications and source are different between Fedora 9 and openSUSE 11.2.
- The alternate procedure uses `yum` in lieu of YaST as a package manager.
- A base installation of Fedora 9 provides different installed packages than openSUSE 11.2, requiring the installation of different dependencies. For example, both the alternate procedure and the installation procedure require the installation of `mesa` and `mesa-devel` (as noted above, the names of installed applications and source are different), but the alternate procedure does not require the installation of `libxml2-devel`, which is a documented dependency of Gazebo.

Therefore, although the order of installation of dependencies selected by the author differs, the elimination of configuration errors which caused various installation failures for the three interdependent packages OIS, CEGUI, and OGRE by revising the "Install OGRE" step to use the `--disable-ogre-demos` flag to force `./configure` to continue without building the OGRE demos resolved one of the two significant problems the author encountered by eliminating the interdependence between OIS, CEGUI, and OGRE.

Resolving this problem during the initial attempt may have eliminated weeks of troubleshooting, and may have resulted in a working installation of Gazebo months before the author was able to verify the installation procedure. Although the author includes detail in this Appendix to provide a more comprehensive history of the development of the installation procedure, including the installation of CEGUI, the author cannot stress enough that if the successful installation of Gazebo, not OGRE, is a research goal, CEGUI should, under no circumstances, be installed.

II.D.2.b. Resolution of problems noted during review of the second and revised second alternate installation procedures

Because the author successfully verified the installation procedure documented in Appendix B in accordance with Appendix C, problems noted during review of the second and revised second alternate procedures were not dispositioned individually in paragraph II.D.2., but rather as documented below:

- Installation instructions for Gazebo documented by [84] confirm the installation instructions documented by Appendix B, step “Install Gazebo”.
- Development (header and library) files for documented and undocumented dependencies should be installed where available.
- There are no “correct” versions of “3rd party packages”. Versions of applications and source other than those documented by [84] may be used, for example OGRE version 1.6.4 in lieu of 1.7.0 and FLTK version 1.1.9 in lieu of 1.1.
- Packages `zziplib` and `zziplib-devel` are documented dependencies of OGRE, not Gazebo. The author concluded [83] is in error.
- Packages `libxml2` and `libxml2-devel` are documented dependencies of Gazebo. Output of the `./cmake ..` command stated, in part:

```
--checking for module 'libxml-2.0'
--  found libxml02.0, version 2.7.3
```

The author concluded [84] is in error.

- [85] states OIS is a documented dependency of Gazebo. The author concluded this may be an error. Output of the `./cmake ..` command does not confirm `cmake` checks for the presence of OIS when configuring Gazebo. The author did not revise the installation procedure to test this, and notes OIS may be an undocumented dependency of OGRE if CEGUI is also installed due to the interdependence between OIS, CEGUI, and OGRE. See paragraphs III.D.1.a., III.G.2.a., and III.I.1.d.
- CEGUI is not a dependency of OGRE, but OGRE must be compiled with the `--disable-ogre-demos` flag to eliminate a configuration error which may be misinterpreted as a statement that CEGUI is a "necessary 3rd party package" when configuring OGRE.
- Cg is not a dependency of OGRE, but OGRE must be compiled with the `--disable-cg` flag to eliminate a configuration error which may be misinterpreted as a statement that Cg is a "necessary 3rd party package" when configuring OGRE.
- Player must be installed before Gazebo.
- Installation instructions for Gazebo documented by [84] confirm `boost-devel` was an undocumented dependency of Gazebo. See paragraph III.L.2.g.

Overall, the author concluded neither [83] nor [84] were comprehensive, or represented sufficient improvement over the installation procedure documented by

Appendix B to cause the author to revise Appendix B.

CHAPTER III. DEVELOPMENT OF THE INSTALLATION PROCEDURE

III.A. Base installation

Package (source package name)	Description (Version)
openSUSE (openSUSE-11.2-NET-i586.iso)	openSUSE Linux distribution (11.2)
Base Development	A minimal set of tools for compiling and linking applications (11.2)
C/C++ Development	Tools and libraries for software development using C/C++ and other derivatives of the C programming language (11.2)
nvidia_gfx_kmp_default	NVIDIA graphics driver kernel module for GeForce4 GPUs (96.43.11_2.6.31.5_0.1-20.2)
x11_video_nvidia	NVIDIA graphics driver for GeForce4 GPUs (96.43.11-21.4)

Based on familiarity with several generations of the SUSE Linux distribution, in particular YaST ("Yet another Setup Tool") for installation of software and management of software updates, the author installed openSUSE 11.2 using the "Internet Installation Boot Image" ("openSUSE-11.2-NET-i586.iso") disk image.

The base installation included the following package groups and packages:

- the default selections for a KDE-based desktop
- "Base Development" package group (YaST pattern view)
- "C/C++ Development" package group (YaST pattern view)
- NVIDIA graphics driver kernel module ("nvidia-gfx-kmp-default 96.43.11_2.6.31.5_0.1-20.2")
- NVIDIA graphics driver ("x11-video-nvidia 96.43.11-21.4")

The NVIDIA graphics driver kernel module and graphics driver were installed from the NVIDIA repository (http://download.nvidia.com/opensuse/11.2) in accordance with [86] and [87] using YaST. All other packages were installed from the openSUSE repository (http://download.opensuse.org/).

Before installing any additional applications from packages or source, the author archived the base installation using the YaST "System Backup" utility. The archive took several hours to complete.

III.B. Path environment variables

Confirm the following path environment variables include the following paths, or export them as necessary:

```
export PATH=/usr/local/bin:$PATH

export CPATH=/usr/local/include:$CPATH

export LIBRARY_PATH=/usr/local/lib:$LIBRARY_PATH

export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:
$PKG_CONFIG_PATH
```

III.C. FreeImage

Package (source package name)	Description (Version)

`FreeImage`
`(FreeImage3130.zip)`
	`Open source image library`
	`(3.13.0)`

III.C.1. Dependencies

Documented dependency	Description (Version)
`-none-`	

Undocumented dependency	Description (Version)
-none-	

III.C.2. Installation instructions

[88] provided documented installation instructions.

III.C.3. Install FreeImage

Compile and install FreeImage as follows:

- `$ make`
- `$ su`
- `$ make install`
- `$ exit`

III.D. Object-oriented Input System (OIS)

Package (source package name)	Description (Version)
OIS (ois_1.2.0.tar.gz)	Cross-platform object-oriented library for handling input devices (1.2.0)

III.D.1. Dependencies

Documented dependency	Description (Version)

`X11`	`X Window System`
	`(-none-)`

`OGRE`
`(GLX Platform)`
	`Object-oriented Graphics Rendering Engine`
	`(-none-)`

`CEGUI`
`(if building CEGUIOgre OIS Demo)`
	`Library providing windowing and widgets for graphics APIs / engines`
	`(0.4.0)`

`"Newer Linux Kernel"`
`(for Event API, otherwise use --disable-joyevents)`
	`Linux kernel`
	`(2.6+ ?)`

Undocumented dependency	Description (Version)
`-none-`	

III.D.1.a. <u>Problem: OIS, CEGUI, and OGRE are interdependent</u>

The following dependencies were documented by [89]:

- `X11`
- `Ogre (GLX Platform) & CEGUI 0.4.0 If building CEGUIOgre OIS Demo`
- `Newer Linux Kernel (2.6+?) for Event API - else, use --disable-joyevents`

OGRE is a documented dependency for OIS if building the "CEGUIOgre OIS demo". OIS is an undocumented dependency for CEGUI if building "OGRE CEGUI demos". OGRE cannot be installed before CEGUI, therefore OGRE cannot be installed before OIS. See paragraphs III.G.2.a. and III.I.1.d.

III.D.1.b. Resolution: None.

This problem had no known impact on the successful installation of OIS because the author had no interest in either the “CEGUIOgre OIS demo” or “OGRE CEGUI demos”.

III.D.2. Installation instructions

[89] provided documented installation instructions.

III.D.2.a. Problem: Option `--disable-joyevents` is an unrecognized option.

The author configured the installation of OIS using:

```
./configure --disable-joyevents
```

which resulted in the following error:

```
configure: WARNING: unrecognized options: --disable-joyevents.
```

III.D.2.b. Resolution: None. Option `--disable-joyevents` is a valid option.

[89] states option `--disable-joyevents` is a valid configuration option. This problem had no known impact on the successful installation of OIS. The author considers this a configuration error.

III.D.3. Install OIS

Configure, compile, and install OIS as follows:

- `$ ./bootstrap`
- `$ ./configure --disable-joyevents`

- `$ make`
- `$ su`
- `$ make install`
- `$ exit`

III.E. Open Dynamics Engine (ODE)

Package (source package name)	Description (Version)

`ODE`
`(ode-0.11.1.tar.gz)`
	`Open source library for simulating rigid body physics`
	`(0.11.1)`

III.E.1. Dependencies

Documented dependency	Description (Version)
`-none-`	

Undocumented dependency	Description (Version)
`-none-`	

III.E.2. Installation instructions

[90] provided documented installation instructions.

III.E.3. Install ODE

Configure, compile, and install ODE as follows:

- `$ ./configure`
- `$ make`
- `$ su`

- `$ make install`
- `$ exit`

III.F. Fast Light ToolKit (FLTK)

Package (source package name)	Description (Version)
FLTK	Cross-platform GUI toolkit (1.1.9)

III.F.1. Dependencies

Documented dependency	Description (Version)
X11 header and library files	X Window System files and libraries required for development (-none-)
OpenGL (or Mesa) header and library files	Open Graphics Library files and libraries required for development (-none-)
JPEG header and library files	JPEG files and libraries required for development (-none-)

Undocumented dependency	Description (Version)
-none-	

III.F.1.a. Install Mesa-devel

Package `Mesa 7.6-3.1` was installed as part of the base installation. To satisfy a documented dependency, the author installed package `Mesa-devel 7.6-3.1` using YaST. Package `libdrm-devel 2.4.14-2.1` was installed by YaST to resolve a dependency.

III.F.2. Installation instructions

[91] provided documented installation instructions.

III.F.2.a. <u>Problem: FLTK failed to compile because of an “invalid conversion” error</u>

The exact error is reproduced below:

```
filename_list.cxx: In function 'int
fl_filename_list(const char*, dirent***, int (*)
(dirent**, dirent**))':
filename_list.cxx:70: error: invalid conversion from
'int (*)(const void*, const void*)' to 'int(*)(const
dirent**, const dirent**)'
filename_list.cxx:70: error:   initializing argument 4
of 'int scandir(const char*, dirent***, int (*)(const
dirent*), int(*)(const dirent**, const dirent**))'
```

III.F.2.b. <u>Resolution: Install FLTK using YaST</u>

Because the teams did not have the time required to troubleshoot an installation procedure (see paragraph II.B.) and because FLTK was available for installation from the openSUSE repository using YaST, the author did not attempt to resolve the error, and instead installed packages `fltk 1.1.9-36.1` and `fltk-devel 1.1.9-36.1` using YaST.

III.F.3. Install FLTK

1. Install `Mesa-devel` (see paragraph III.F.1.a.).

2. Install `fltk` and `fltk-devel` from the openSUSE repository using YaST.

III.G. CrazyEddie's GUI System (CEGUI)

Package (source package name)	Description (Version)

`CEGUI`
`(CEGUI-0.6.2b.tar.gz)`
	`CrazyEddie's GUI System (CEGUI)`
	`(0.6.2b)`

III.G.1. Dependencies

[92] provided a list of documented dependencies.

Documented dependency	Description (Version)

`FreeType2`	`Software font engine`
	`(-none-)`

`PCRE`	`Perl-Compatible Regular Expressions library`
	`(-none-)`

III.G.1.a. Install `pcre-devel`

Packages `freetype2 2.3.9-2.2`, `freetype2-devel 2.3.9-2.2`, and `pcre 7.9.0-2.3.1` were installed as part of the base installation. To satisfy a documented dependency, the author installed package `pcre-devel 7.9.0-2.3.1` using YaST. Packages `libpcre0 7.9.0-2.3.1`, `libpcreposix0 7.9.0-2.3.1`, `libpcrecpp0 7.9.0-2.3.1` were installed by YaST to resolve a dependency.

Undocumented dependency	Description (Version)
`-none-`	`-none-`

III.G.2. Installation instructions

[93] provided documented installation instructions.

III.G.2.a. Problem: OIS, CEGUI, and OGRE are interdependent

The initial attempt was based on documented installation instructions and used documented dependencies to establish the installation procedure. OIS is a documented dependency of Gazebo, however OIS is not a documented dependency of CEGUI. Because OIS was not installed, configuration of CEGUI resulted in a warning which stated, in part:

```
You do not have OIS installed.  This is required to
build Ogre CEGUI demos.
```

and continued:

```
If you do not want to build the demos, you can safely
ignore this.
```

During the initial attempt, the author installed OIS before continuing with the installation of CEGUI.

III.G.2.b. Resolution: None.

The author revised the installation procedure to install OIS before CEGUI, thus resolving the problem. The author does not consider this a configuration error because configuration of CEGUI continued and was successfully completed.

Based on a review of online documentation, the author determined that the normal installation procedure for CEGUI and OGRE is: CEGUI, OGRE, then CEGUI again for the "OGRE CEGUI demos". OIS is required to be installed both before CEGUI and OGRE, so perhaps the normal installation procedure should be: OIS, CEGUI, OGRE, OIS (for the "CEGUIOgre Demo"), and finally CEGUI (for the "OGRE CEGUI demos").

The author had no interest in CEGUI except as a dependency for OGRE, and so did not attempt to determine the correct order of installation to build the "CEGUIOgre Demo" or "OGRE CEGUI demos" using OIS, CEGUI, and OGRE. See paragraphs III.D.1.a. and III.I.1.d.

III.G.2.c. Problem: `install` failed when attempting to overwrite an existing just-created file

The very first line output by `./configure` is:

```
checking for a BSD-compatible install...
/usr/bin/install -c.
```

However, `man install` states the `-c` flag is `(ignored)` and `info install` states the `-c` flag is: `Ignored; for compatibility with old Unix versions of 'install'.` As a result, `./configure` reported success:

```
Now you can do make && make install.  Good luck!
```

But attempts to `make install` failed as a result of "will not overwrite just-created" file errors.

Based on a search of the CEGUI wiki ([93]) for similar problems reported by other users, the author initially determined this problem is due to a difference between the `-c` and `-C` options, but based on an evaluation of the results of attempting to `install` files twice in a single `install` invocation with and without the `--compare (-C)` option finally concluded the error is due to the fact that install will attempt to copy a file onto itself, and fail when unsuccessful unless the `--compare (-C)` option is used.

By default, `install` fails when attempting to overwrite an existing just-created file. However, the `-C` flag causes `install` to ignore subsequent attempts to overwrite an existing just-created file. `man install` states the -C flag causes `install` to "compare each pair of source and destination files, and in some cases, do not modify the destination at all" but does not explain how this is prevented "in some cases".

The author reviewed the GNU Core Utilities ("`coreutils`") source code ([94]) to determine when the `-C` option prevents file copying. As a result of that review, the author concluded there is some confusion concerning the intended use of the `-C` option. The "ChangeLog" included with coreutils (`coreutils 7.1` was installed as part of the base installation) states functions `have_same_content` and `need_copy` were added to `install.c` when it was modified to recognize the `--compare (-C)` option to install files only when necessary.

`man install` recognizes the `--compare (-C)` option, but also states:

```
The full documentation for install is maintained as a
Texinfo manual.  If the info and install programs are
properly installed at your site, the command
     info coreutils 'install invocation'
should give you access to the complete manual.
```

However, the manual doesn't recognize the `--compare (-C)` option. The manual states (of `install`): "It refuses to copy files onto themselves." As demonstrated below, this is misleading. `install` will copy a file onto itself, but not if the file name occurs more than once in the same invocation.

The "ChangeLog" states function `copy_file` was revised to: "Skip file copying if not necessary." However this is also misleading, and is not the observed behavior. Function `copy_file` and, by extension `install`, attempts to copy even if not necessary unless the `--compare (-C)` option is invoked.

After reviewing the relevant source code (`install.c` and `copy.c`), the author concluded the intended use of the `--compare (-C)` option is to cause `install` to check to see if there is a difference between two files by several methods and prevent copying if there is no difference between them if the option is used, but to otherwise allow copying to fail due to a "will not overwrite just-created" error if copying a file onto itself twice in the same invocation. As noted above, `install` will copy a file onto itself, and also fail when attempting to copy a file onto itself in the same invocation unless the `--compare (-C)` option is used.

The `make install` output indicates `install` failed on a line which attempted to install the files `CEGUIListHeader.h` and `CEGUIListHeaderProperties.h` twice. The author confirmed `install` <u>`source`</u> <u>`destination`</u> will copy a source file to the destination. Subsequent attempts to install the source file to the same destination will copy the source file to the destination after first removing the existing file at destination:

```
$ install -v test.h ./test
$ 'test.h' -> './test/test.h'
$ install -v test.h ./test

$ removed './test/test.h'

$ 'test.h' -> './test/test.h'
```

Attempting to copy the file twice with a single install command fails with a "will not overwrite just-created" file error if the file does not already exist at the destination:

```
$ install -v test.h test.h ./test

$ install: will not overwrite just-created

'./test/test' with 'test.h'
```

If the file already exists at the destination, attempting to copy the file twice with a single install command results in the file first being overwritten, then install failing as above with a "will not overwrite just-created" file error:

```
$ install -v test.h test.h ./test

$ removed './test/test.h'

$ 'test.h' -> './test/test.h'

$ install: will not overwrite just-created

'./test/test' with 'test.h'
```

Attempting to copy the file multiple times with a single `install` command when using the `-C` flag results in the file being installed at the destination, but `install` ignores subsequent attempts to overwrite the file:

```
$ install -C -v test.h test.h test.h ./test

$ 'test.h' -> './test/test.h'
```

If the file already exists at the destination, install ignores subsequent attempts to overwrite the file when using the `-C` flag:

```
$ install -C -v test.h ./test

$ install -C -v test.h test.h ./test

$ install -C -v test.h test.h test.h ./test
```

III.G.2.d. <u>Resolution: revise `./configure` so that `install -C` is invoked in lieu of `install -c`</u>

The author resolved this problem by modifying `./configure` to invoke `install -C` in lieu of `install -c` on lines 2325, 2329, 2333, 2408, and 2644. The author notes that the duplicate file names could also have been deleted from each line which invoked `install`.

III.G.3. Install CEGUI

1. Install `pcre-devel.`
2. Revise `./configure` to invoke `install -C` in lieu of `install -c.`
3. Configure, compile, and install CEGUI as follows:
   - `$ ./configure`
   - `$ make`
   - `$ su`
   - `$ make install`
   - `$ exit`

III.G.4. After developing the installation procedure, the author determined CEGUI was not an undocumented dependency

Neither [95] nor [85] refer to CEGUI as a dependency. [95] refers to CEGUI as an “optional” library, and [85] identifies OGRE as a dependency. Despite being an “optional” library, the author concluded CEGUI was an undocumented dependency for OGRE based on output from `./configure` when attempting to install OGRE during the initial attempt. Resolution of problems identified during the installation of OIS, CEGUI, and OGRE during the initial attempt, in particular the required installation order made effective troubleshooting more difficult.

However, as noted in paragraph II.C. above, based on failure of the initial attempt the author developed a method for troubleshooting the installation procedure which was used to develop the installation procedure herein. Maintaining a record of errors encountered allowed the author to determine CEGUI was not an undocumented dependency during the development of this installation procedure. Also, because the author had no interest in building either the “CEGUIOgre Demo” when installing OIS or “OGRE CEGUI demos” when installing CEGUI, the author concluded CEGUI should not be installed.

As a result, the author revised the installation procedure to delete step “Install CEGUI” in its entirety, revised step “Install OGRE” to use the `--disable-ogre-demos` flag to force `./configure` to continue without building the OGRE demos, and confirmed OGRE installation using Cg during verification of the installation procedure.

The author considers this a configuration error. Configuration of OGRE should not have failed because the “optional” CEGUI library was not installed. However, the author does not consider CEGUI an undocumented dependency of OGRE because

CEGUI support could be disabled.

III.H. Cg

Package (source package name)	Description (Version)
`Cg`	`Compile and runtime libraries for the Cg graphics language (2.2)`

III.H.1. Dependencies

Documented dependency	Description (Version)
`-none-`	

Undocumented dependency	Description (Version)
`-none-`	

III.H.2. Installation instructions

[95] states only that Cg is an "optional" library. Installation instructions for Cg were unavailable from either [95] or [96]. During the initial attempt the author downloaded Cg (version 2.2: "Cg-2.2_April2009_x86.tgz") from NVIDIA ([96]) and extracted the file using the following command into the `ogre` directory:

```
sudo tar xzf ./Cg-2.2_April2009_x86.tgz
```

III.H.2.a. Problem: after extracting Cg into the ogre directory during the initial attempt, it was not available to `./configure`

Attempts to configure OGRE failed because Cg must be extracted into the root directory.

III.H.2.b. Resolution

The author deleted the directory containing the files extracted into the `ogre` directory.

III.H.2.c. Problem: extracting Cg into the root directory during the initial attempt preserved the existing user identification and group identification of all files in the archive

The author extracted the files using the following command into the root directory:

```
sudo tar xzf ./Cg-2.2_April2009_x86.tgz
```

However, the existing user identification (uid) of 2402 and group identification (gid) of 30 of all files in the archive were preserved. The author notes that this is the default behavior for the root user, unless the option `--no-same-owner` is used.

III.H.2.d. Resolution

The author identified affected files as follows:

```
ls -a | grep 2402
```

The author modified the attributes of the affected files as follows:

```
chown root <filename>

chgrp root <filename>
```

Due to the failure of the initial attempt, the author decided to limit the installation of “optional” libraries to the extent possible and did not install the NVIDIA Cg library (“Cg”) during the development of this installation procedure, and configured OGRE with the command:

```
./configure --with-platform=GLX --disable-cg
```

to force `./configure` to continue without Cg support.

However, after successfully building Gazebo, the author re-evaluated the installation of Cg based on the similarity of the alternate procedure to the installation procedure. Because the author was only able to successfully install Cg from source with difficulty (see paragraphs III.H.2.a. and III.H.2.c., above), the author confirmed that Cg was available for installation from the openSUSE repository using YaST, and installed packages `cg 2.2-1.1.1` and `cg-devel 2.2-1.1.1` using YaST.

The author then revised the "Install OGRE" step of the installation procedure to not disable Cg support using the `--disable-cg` flag and confirmed OGRE installation using Cg during the verification of the installation procedure.

III.H.3. Install Cg

1. Install `cg` and `cg-devel` from the openSUSE repository using YaST.

III.I. OGRE

Package (source package name)	Description (Version)

`OGRE`
`(ogre-v1-6-4.tar.bz2)`
	`Object-oriented Graphics Rendering Engine`
	`(1.6.4)`

III.I.1. Dependencies

Documented dependency	Description (Version)
`automake`	`A program for generating makefiles (1.9.5 (1.6+ required))`
`autoconf`	`A tool for configuring source code (2.59a (2.50+ required))`
`make`	`The make command (3.80)`
`libtool`	`A tool to build shared libraries (1.5.6 (1.4+ required)`
`pkg-config`	`A library management system (0.17.2)`
`gcc`	`The system C compiler (3.3.5)`
`g++ (gcc-c++)`	`The system C++ compiler (3.3.5)`
`cpp`	`The system preprocessor (3.3.5)`
`FreeType2`	`Software font engine (2.1.x+)`
`zziplib`	`Zip compression library (0.13.x+)`
`FreeImage`	`Open source image library (-none-)`

III.I.1.a. Install `zziplib` and `zziplib-devel`

To satisfy a documented dependency, the author installed package `zziplb 0.13.56-2.1` and `zziplib-devel 0.13.56-2.1` using YaST.

Undocumented dependency	Description (Version)
`GLEW`	`OpenGL Extension Wrangler library (-none-)`

III.I.1.b. Problem: GLEW is an undocumented dependency

The first attempt to compile OGRE failed due to a "No such file or directory" error, a portion of which is reproduced below:

```
from OgreGLXGLSupport.cpp:35:
../../../../RenderSystems/GL/include/GL/glew.h:1128:20
: error: GL/glu.h: No such file or directory
```

The error suggested the problem was related to GLEW, which was not installed as part of the base installation. Because `./configure` did not warn or fail as a result, the author considers this an undocumented dependency.

III.I.1.c. Resolution: install `glew` and `glew-devel`

To satisfy an undocumented dependency, the author installed packages `glew 1.5.1-2.1` and `glew-devel 1.5.1-2.1` using YaST. Package `libGLEW1_5 1.5.1-2.1` was installed by YaST to resolve a dependency.

III.I.1.d. Problem: OIS, CEGUI, and OGRE are interdependent

The initial attempt was based on documented installation instructions and used documented dependencies to establish the installation procedure. OGRE is a documented dependency for Gazebo, however neither OIS nor CEGUI are documented dependencies of OGRE. During the initial attempt, configuration of OGRE failed because neither OIS nor CEGUI were installed.

III.I.1.e. Resolution: None.

The author revised the installation procedure to install OIS before CEGUI and CEGUI before OGRE, thus resolving the problem. Based on a review of online documentation, the author determined that the normal installation procedure for CEGUI and OGRE is: CEGUI, OGRE, then CEGUI again for the “OGRE CEGUI demos”. OIS is required to be installed both before CEGUI and OGRE, so perhaps the normal installation procedure should be: OIS, CEGUI, OGRE, OIS (for the "CEGUIOgre Demo"), and finally CEGUI (for the "OGRE CEGUI demos").

The author had no interest in CEGUI except as a dependency for OGRE, or OGRE except as a dependency for Gazebo, and so did not attempt to determine the correct order of installation to build the "CEGUIOgre Demo" or "OGRE CEGUI demos" using OIS, CEGUI, and OGRE. See paragraphs III.D.1.a. and III.G.2.a.

III.I.2. Installation instructions

[95] provided documented installation instructions.

III.I.2.a. <u>Problem: configuration failed because the NVIDIA Cg library was not installed. The NVIDIA Cg library is optional.</u>

The first attempt to configure OGRE with the command:

```
./configure --with-platform=GLX
```

failed, resulting in an error which stated, in part:

```
You do not have the nVidia Cg libraries installed.
```

and continued:

```
You can disable the building of Cg support by
providing --disable-cg to this configure script but
this is highly discouraged as this breaks many of the
examples.
```

The author considers this a configuration error. `./configure` should not have failed because the "optional" NVIDIA Cg library ("Cg") was not installed. However, the author does not consider Cg an undocumented dependency because building Cg support could be disabled.

III.I.2.b. Resolution: Install Cg using YaST

See paragraph III.H.

III.I.3. Install OGRE

1. Install `zziplib` and `zziplib-devel`.
2. Install `glew` and `glew-devel`.
3. Configure, compile, and install OGRE as follows:
   - `$ ./bootstrap`
   - `$ ./configure --with-platform=GLX --disable-ogre-demos`
   - `$ make`
   - `$ su`
   - `$ make install`
   - `$ exit`

III.J. FFmpeg

FFmpeg was neither a documented nor undocumented dependency of Gazebo, but was installed to provide access to libavcodec.

Package (source package name)	Description (Version)
FFmpeg (ffmpeg-0.5.tar.bz2)	command line tool to convert multimedia files between formats (0.5)

III.J.1. Dependencies

Documented dependency	Description (Version)

`make`	`The make command`
	`(3.81+)`

Undocumented dependency	Description (Version)
`-none-`	

III.J.2. Installation instructions

[97] provided documented installation instructions.

III.J.2.a. Problem: shared libraries were not enabled by default

The first attempt to configure, compile, and install FFmpeg was successful. However, a subsequent attempt to compile Gazebo resulted in failure due to numerous "undefined reference" errors to functions beginning with "av_". The author determined these functions were provided by libavcodec, which is provided by FFmpeg.

III.J.2.b. Resolution: enable shared libraries

The author uninstalled FFmpeg and configured the build to enable shared libraries:

```
./configure --enable-shared
```

The author then compiled and installed FFmpeg successfully. This resolved the "undefined reference" errors encountered when attempting to compile Gazebo.

III.J.3. Install FFmpeg

Configure, compile, and install FFmpeg as follows:

- `$ ./configure --enable-shared`
- `$ make`
- `$ su`
- `$ make install`
- `$ exit`

III.K. Player

Package (source package name)	Description (Version)

`Player`
`(player-3.0.0.tar.gz)`
	`the Player server`
	`(3.0.0)`

III.K.1. Dependencies

Documented dependency	Description (Version)
`-none-`	

Undocumented dependency	Description (Version)

`cmake`
`(cmake-2.6.4-3.3)`
	`Cross-platform, open source make system`
	`(2.6.4)`

III.K.1.a. Problem: `cmake` was an undocumented dependency

Online documentation for the Player Project was not up-to-date. The application cmake was an undocumented dependency. The installation procedure given by [98] ("Standard install procedure") was:

- Download the latest Player source tarball (player-<version>.tgz) from Sourceforge.

- Uncompress and expand the tarball:

```
$ tar xzvf player-<version>.tgz
```

- 'cd' into Player's source directory:

```
$ cd player-<version>
```

- To configure Player with default settings:

```
$ ./configure
```

- Compile Player:

```
$ make
```

- Install Player. By default, Player will be installed in /usr/local so you need to become root for this step. Remember to return to your normal user ID afterwards.

```
$ make install
```

The installation procedures given by [99] ("Installation") and [100] ("Out-of-source Build") were essentially the same:

`$ cd player` (this step is omitted by [99])

```
$ mkdir build
$ cd build
$ cmake ../
```

`$ make` (this step is omitted by [99])

```
$ make install
```

The installation procedure given by [98] was incorrect. Player 3.0.0 was released on September 7, 2009. An announcement made to the playerstage-users mailing list stated: an “entirely new build system using CMake” was a feature of the new release. As a result, the author was aware the installation instructions for Player 3.0.0 had changed and concluded online documentation was not updated to document the use of `cmake` to configure Player 3.0.0.

Because online documentation was initially favored (see paragraph II.B.), the author considers `cmake` an undocumented dependency.

III.K.1.b. <u>Resolution: install `cmake`</u>

The author installed package `cmake 2.6.4-3.3` using YaST. During development of the installation procedure, the author also installed package `cmake-gui 2.6.4-3.3` using YaST to provide a more usable front-end for `cmake`. However, the author did not re-install `cmake-gui` during the verification of the installation procedure because it was less useful than anticipated.

III.K.2. Installation instructions

[98], [99], and [100] provided documented installation instructions. The installation procedure given by [98] was incorrect.

III.K.2.a. <u>Problem: the environment variables `PYTHON_INCLUDE_PATH` and `PYTHON_LIBRARY` were set to `NOTFOUND`</u>

Configuration of Player failed because the environment variables `PYTHON_INCLUDE_PATH` and `PYTHON_LIBRARY` were set to `NOTFOUND`. As noted in paragraph III.A. above, the author installed the “Base Development”and “C/C++ Development” package groups, but did not install the “Python Development” package group.

III.K.2.b. Resolution: Set `BUILD_PYTHONC_BINDINGS` to `OFF`

The author used `ccmake` to review the configuration options and set "`BUILD_PYTHONC_BINDINGS: Build the Python bindings for the C client library`" to `OFF`. The configuration option "`BUILD_PYTHONCPP_BINDINGS`" was then set to `OFF` by `ccmake` by default. The author considers this a configuration error. `./configure` should not have failed because an "optional" library was not installed. The author does not consider Python an undocumented dependency because building Python bindings for the C client library could be disabled.

III.K.3. Install Player

1. Install `cmake` (see paragraph III.K.1.b.).
2. Set `BUILD_PYTHONC_BINDINGS` to `OFF` (see paragraph III.K.2.b.).
3. Configure, compile, and install Player as follows:
   - `$ mkdir build`
   - `$ cd build`
   - `$ cmake ..`
   - `$ make`
   - `$ su`
   - `$ make install`
   - `$ exit`

III.L. Gazebo

[80] and [81] provide contradictory installation instructions. Neither is complete

or correct. The initial attempt to compile and install Gazebo failed with numerous errors. [80] states: “If things go wrong, please check the archives of the Gazebo mailing list.” The author found online documentation to be of limited utility. Review of available forums including the playerstage-gazebo mailing list reveals it is not uncommon for several people to ask the same question or describe the same problem, often with no recorded resolution. Several of the problems the author encountered during the initial attempt and later successful attempt were also identified by others before and after the development of this installation procedure, without resolution.

Package (source package name)	Description (Version)

`Gazebo`
`(-none-)`
	`Gazebo`
	`(0.9.0 rev. 8443)`

III.L.1. Problems encountered before installation

III.L.1.a. Problem: installation instructions included with online documentation are incorrect.

[80] states the following will extract Gazebo:

```
$ tar xvjf gazebo-<version>.tar.gz
```

However, Gazebo 0.9.0 is distributed as a “bz2” file (“gazebo-0.9.0.tar.bz2”). As a result, attempting to follow these instructions results in the following error:

```
gzip: stdin: not in gzip format
tar: Child returned status 1
tar: Error exit delayed from previous errors
```

III.L.1.b. Resolution: None.

All other releases of the Gazebo package hosted by [38] are ".tar.gz" files. The author concluded [80] was not updated to provide installation instructions for the current Gazebo 0.9.0 package. The author used Ark 2.10.999 to extract Gazebo.

III.L.1.c. Problem: installation instructions included with packaged documentation are incomplete.

[81] stated:

```
Installation
------------
Read the installation instructions in the online
manual for generic instructions.  For most people, the
following sequence will suffice:

$ mkdir build (inside the gazebo-trunk directory)
$ cd build
$ cmake ..
$ make

Uninstallation
--------------
Read the installation instructions in the online
manual for generic instructions.  For most people, the
following sequence will suffice:
```

```
$ make install (inside the gazebo-trunk/build
directory)
```

This appears to be an editorial error. To install Gazebo, it would be necessary for a user to read, then follow, the uninstallation instructions. No uninstallation instructions were provided.

III.L.1.d. Resolution: None.

The author concluded installation instructions provided by [81] were incomplete. Based on the installation instructions for Player, the author was familiar with the installation procedure using `cmake`.

III.L.1.e. Problem: installation instructions provided by online documentation do not match installation instructions provided by packaged documentation.

Installation instructions provided by [80] do not match the installation instructions provided by [81] (the Gazebo package contains no "INSTALL" file). [80] stated:

```
$ tar xvzf gazebo-<version>.tar.gz
$ cd gazebo-<version>
$ scons
Note that scons will fail if any of the required
packages are missing. Once Gazebo has been built, it
can be installed:
$ su
$ scons install
$ exit
```

Installation instructions provided by [81] are documented above.

III.L.1.f. Resolution: None.

Based on the result of trying to install Gazebo using `scons`, the author concluded [80] was incorrect. See paragraph III.L.3.a.

III.L.1.g. Problem: online documentation does not provide the latest installation instructions.

[81] also stated:

```
On-line installation instructions
---------------------------------
The latest installation instructions can be found on-
line, at
  http://playerstage.sourceforge.net/doc/
```

However, attempting to access this URL results in a "403 error".

III.L.1.h. Resolution: None.

The author concluded the latest installation instructions are included with packaged documentation, not online documentation.

III.L.1.i. Problem: Online documentation directs users to the socalwifi-iptv mailing list archive in lieu of the playerstage-gazebo mailing list archive.

[80] stated: "If things go wrong, please check the archives of the Gazebo mailing

list. Please read the instructions below carefully before reporting posting [*sic*] to the mailing list." However, the hyperlink to the Gazebo mailing list ("http://sourceforge.net/mailarchive/forum.php?forum_id=33909") is a hyperlink to the mailing list archive for the socalwifi-iptv project on SourceForge.net. The playerstage-gazebo mailing list ("http://sourceforge.net/mailarchive/forum.php?forum_id=27052") is accessible from the Player Project "Support" page.

III.L.2. Dependencies

[85] provided a list of documented dependencies.

Documented dependency	Description (Version)

`scons`	`Replacement for make`
	`(0.97 or greater)`

`fltk`	`Cross-platform GUI toolkit`
	`(1.1.7 or greater)`

`OGRE`	`Object-oriented Graphics Rendering Engine`
	`(1.4.4)`

`ODE`	`Open source library for simulating rigid body physics`
	`(0.8)`

`OIS`	`Cross-platform object-oriented library for handling input devices`
	`(1.0)`

`libxml2`	`A library to manipulate XML files`
	`(2.6.29 or greater)`

III.L.2.a. <u>Problem: a later version of OGRE was required to compile Gazebo than that documented by the installation instructions</u>

The author downloaded and installed a version of OGRE other than version 1.4.4 which [85] states is a prerequisite. The author concluded OGRE version 1.6.3 or greater is required to compile Gazebo based on the following output of the `cmake ../`

command:

```
-- checking for module 'OGRE>=1.6.3'
--   found OGRE, version 1.6.4
```

III.L.2.b. <u>Resolution: None</u>

The author concluded [85] was incorrect.

III.L.2.c. <u>Problem: `libxml2-devel` was not installed by default</u>

Package `libxml2` was installed as part of the base installation. However, the development (header and library) files were not installed. Attempts to compile Gazebo resulted in the following error:

```
Error: libxml2 and development files not found.
```

III.L.2.d. <u>Resolution: install `libxml2-devel`</u>

The author installed package `libxml2-devel 2.7.3-2.2` using YaST. Package `readline-devel 6.0-18.3` was installed by YaST to resolve a dependency.

Undocumented dependency	Description (Version)
`-none`	

III.L.2.e. <u>Problem: during the initial attempt the author concluded packages `freeglut` and `openal` were undocumented dependencies</u>

Based on errors encountered during the initial attempt, the author concluded `freeglut` and `openal` were undocumented dependencies of Gazebo.

As a result, the author installed packages `freeglut`, `freeglut-devel`, `openal-soft`, `openal-soft-devel`, and `libopenal1-soft`.

During development of the installation procedure, package `freeglut 090301-3.1` was installed as part of the base installation, and the author installed packages `freeglut-devel 090301-3.1`, `openal-soft 1.9.616-1.1.1`, `openal-soft-devel 1.9.616-1.1.1`, and `libopenal1-soft 1.9.616-1.1.1` using YaST.

III.L.2.f. Resolution: packages freeglut and openal are optional libraries, not undocumented dependencies

Based on the author's decisions to disposition errors on a case basis as either configuration errors or evidence of undocumented dependencies, and to limit the use of "optional" libraries to simplify the installation procedure to the extent possible, the author re-evaluated the installation of these packages, determined they were optional libraries, not undocumented dependencies, and did not install them during verification of the installation procedure.

III.L.2.g. Problem: `boost-devel` is an undocumented dependency.

During the initial attempt, attempts to compile Gazebo failed because package `boost-devel` was not installed.

III.L.2.h. Resolution: None.

During the initial attempt, the author installed package `boost-devel 1.36.0-9.5` using YaST. This problem did not recur during the development of the installation procedure because `boost-devel 1.39.0-3.4.1` is part of the openSUSE 11.2 "C/C++ Development" package group.

III.L.3. Installation instructions

[80] and [81] provided documented installation instructions.

III.L.3.a. Problem: `scons` is no longer used to configure or compile Gazebo

As noted in Chapter II., the author first attempted to follow documented installation instructions. [80] states `scons` is used to configure, make, and install Gazebo. The author installed package `scons 1.2.0-2.2` using YaST, then attempted to configure and compile Gazebo using `scons`, resulting in the following error:

```
scons: *** No SConstruct file found.
```

The SConstruct file is required.

III.L.3.b. Resolution: None.

[81] provided alternate installation instructions. The author concluded [80] was incorrect.

III.L.3.c. Problem: an attempt to compile Gazebo resulted in a "cannot convert" error

After successfully configuring the build of Gazebo, the author attempted to compile Gazebo, resulting in the following error:

```
gazebo/server/controllers/audio/Audio.cc: In member
function 'void
gazebo::AudioController::PutAudioData()':
gazebo/server/controllers/audio/Audio.cc:160: error:
cannot convert 'gazebo::Time' to 'double' in
assignment
```

III.L.3.d. Resolution: revise file `Audio.cc` to eliminate the source of the error

The author revised line 160 of file `Audio.cc`:

```
this->audioIface->data->head.time =
Simulator::Instance()->GetSimTime();
```

as follows:

```
this->audioIface->data->head.time =
Simulator::Instance()->GetSimTime().Double();
```

The author submitted bug report number 2909192 on December 5, 2009 to report this problem to the Player Project.

III.L.4. Install Gazebo

1. Install `libxml2-devel.`
2. Revise line 160 of file `Audio.cc`:

   ```
   this->audioIface->data->head.time =
   Simulator::Instance()->GetSimTime();
   ```

   as follows:

   ```
   this->audioIface->data->head.time =
   Simulator::Instance()->GetSimTime().Double();
   ```

3. Configure, compile, and install Gazebo as follows:
   - `$ mkdir build`
   - `$ cd build`
   - `$ cmake ..`

- `$ make`
- `$ su`
- `$ make install`
- `$ exit`

Appendix B: Installation Procedure

I. PATH ENVIRONMENT VARIABLES

Confirm the following path environment variables include the following paths, or export them as necessary:

```
export PATH=/usr/local/bin:$PATH
export CPATH=/usr/local/include:$CPATH
export LIBRARY_PATH=/usr/local/lib:$LIBRARY_PATH
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:
$PKG_CONFIG_PATH
```

II. INSTALL FREEIMAGE

Compile and install FreeImage as follows:

- `$ make`
- `$ su`
- `$ make install`
- `$ exit`

III. INSTALL OIS

Configure, compile, and install OIS as follows:

- `$ ./bootstrap`
- `$ ./configure --disable-joyevents`
- `$ make`
- `$ su`
- `$ make install`
- `$ exit`

IV. INSTALL ODE

Configure, compile, and install ODE as follows:

- `$ ./configure`
- `$ make`
- `$ su`
- `$ make install`
- `$ exit`

V. INSTALL FLTK

1. Install `Mesa-devel` from the openSUSE repository using YaST (see paragraph III.F.1.a.).
2. Install `fltk` and `fltk-devel` from the openSUSE repository using YaST.

VI. INSTALL CG

Install `cg` and `cg-devel` from the openSUSE repository using YaST.

VII. INSTALL OGRE

1. Install `zziplib` and `zziplib-devel` from the openSUSE repository using YaST.
2. Install `glew` and `glew-devel` from the openSUSE repository using YaST.
3. Configure, compile, and install OGRE as follows:
   - `$ ./bootstrap`
   - `$ ./configure --with-platform=GLX --disable-`

`ogre-demos`

- `$ make`
- `$ su`
- `$ make install`
- `$ exit`

VIII. INSTALL FFMPEG

Configure, compile, and install FFmpeg as follows:

- `$ ./configure --enable-shared`
- `$ make`
- `$ su`
- `$ make install`
- `$ exit`

IX. INSTALL PLAYER

1. Install `cmake` from the openSUSE repository using YaST (see paragraph III.K.1.b.).
2. Configure, compile, and install Player as follows:
   - `$ mkdir build`
   - `$ cd build`
   - `$ cmake ..`
   - `$ ccmake ..`
   - Set `BUILD_PYTHONC_BINDINGS` to `OFF` (see paragraph III.K.2.b.)
   - `$ cmake ..`

- `$ make`
- `$ su`
- `$ make install`
- `$ exit`

X. INSTALL GAZEBO

1. Install `libxml2-devel` from the openSUSE repository using YaST.
2. Revise line 160 of file `Audio.cc`:

   ```
   this->audioIface->data->head.time =
   Simulator::Instance()->GetSimTime();
   ```

   as follows:

   ```
   this->audioIface->data->head.time =
   Simulator::Instance()->GetSimTime().Double();
   ```

3. Configure, compile, and install Gazebo as follows:
   - `$ mkdir build`
   - `$ cd build`
   - `$ cmake ..`
   - `$ make`
   - `$ su`
   - `$ make install`
   - `$ exit`

Appendix C: Verification of the Installation Procedure

I. CONVENTIONS

During verification of the installation procedure, standard output and standard error from `./bootstrap`, `./configure`, `cmake`, `make`, and `make install` commands were re-directed to files to confirm successful installation. The installation procedure was verified as follows:

II. ARCHIVE THE BASE INSTALLATION

The author archived the base installation using the YaST "System Backup" utility before installing any additional applications from packages or source. See Appendix A.

III. DEVELOP THE INSTALLATION PROCEDURE

The author developed the installation procedure. See Appendix A.

IV. DOCUMENT THE INSTALLATION PROCEDURE

The author documented the installation procedure. See Appendix B.

V. RESTORE THE BASE INSTALLATION

The author restored the base installation from backup using the YaST "System Restoration" utility. The author decided not to re-install openSUSE because versions of packaged software may have changed from those installed during the base installation. However, the "System Restoration" utility functioned more or less as a new installation of openSUSE 11.2, "restoring" the system by installing updated versions of packages installed as part of the base installation package groups.

VI. UNINSTALL APPLICATIONS AND SOURCE INSTALLED IN ACCORDANCE WITH APPENDIX B

The author uninstalled applications and source installed in accordance with

Appendix B using the `make uninstall` and `make clean` commands, with the following exceptions:

VI.A. FreeImage

The attempt to `make uninstall` resulted in the following output:

```
make: *** No rule to make target 'uninstall'.  Stop.
```

As a result, the author reviewed the output of the previous `make install` command to determine which files were installed, then deleted the following files:

```
/usr/include/FreeImage.h
/usr/lib/libfreeimage.a
/usr/lib/libfreeimage-3.13.0.so
```

VI.B. FLTK

Packages `fltk` and `fltk-devel` were deleted when the base installation was restored from backup.

VI.C. Cg

Packages `cg` and `cg-devel` were not installed during development of the installation procedure.

VI.D. Player

The first attempt to run command `make uninstall` failed because `cmake` was deleted when the base installation was restored from backup. The author installed `cmake` from the openSUSE repository using YaST. Because `cmake` was installed during verification of the installation procedure, it was not necessary to install it later. The author did not revise the installation procedure to delete this step because the intent of the installation procedure is to provide instructions which will result in the successful installation of Player and Gazebo on the first attempt using the base installation as a baseline.

The command `make clean` resulted in no output.

VI.E. Gazebo

The attempt to `make uninstall` resulted in the following output:

```
make: *** No rule to make target 'uninstall'.  Stop.
```

As a result, the author archived the existing installation of Gazebo by renaming the containing directory and then downloading Revision 8443 of the Gazebo 0.9.0 source code using `svn`:

```
svn co https://playerstage.svn.sourceforge.net/
svnroot/playerstage/code/gazebo/trunk@8443 gazebo
```

File `.gazeborc` was deleted by the author.

The command `make clean` resulted in no output.

VII. RE-INSTALL APPLICATIONS AND SOURCE IN ACCORDANCE WITH APPENDIX B

The author re-installed applications and source in accordance with Appendix B, with the following exceptions:

VII.A. Step "Path environment variables"

The author exported CPATH, LIBRARY_PATH, and PKG_CONFIG_PATH. It was not necessary to export PATH, which included directory `/usr/local/bin`. The author notes that development of the installation procedure resulted in a successful installation of Gazebo without exporting additional paths.

VII.B. Step "Install Cg"

When attempting to install `cg` and `cg-devel` from the openSUSE repository using YaST, the author received the following warning:

```
nothing provides libGLU.so needed by cg-2.2-1.1.1-i586
```

The author confirmed package `Mesa` provides `libGLU.so.1`, and created the following symbolic link:

```
/usr/lib/libGLU.so -> libGLU.so.1
```

then forced installation to continue. Package `libstdc++33 3.3.3-15.3` was installed by YaST to resolve a dependency.

VII.C. Step "Install Player"

The first attempt to use `ccmake` to "Set BUILD_PYTHONC_BINDINGS to OFF" failed because `cmake` had not been run. As a result, there was no `CMakeCache.txt` file. The author revised step "Install Player" to require `cmake` to be run before using `ccmake` to complete this step.

VIII. VERIFY A WORKING INSTALLATION OF GAZEBO

The author confirmed a working installation of Player and Gazebo by constructing a simple Gazebo world file and Player configuration file, then executing the following commands from the gazebo directory:

```
$ gazebo worlds/test.world
$ player player_cfgs/test.cfg
$ playerv
```

Appendix D: Improved Steering Controller and Wheel

```
/*
 *  Gazebo - Outdoor Multi-Robot Simulator
 *  Copyright (C) 2003
 *     Nate Koenig & Andrew Howard
 *
 *  This program is free software; you can redistribute it and/or
modify
 *  it under the terms of the GNU General Public License as published
by
 *  the Free Software Foundation; either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program; if not, write to the Free Software
 *  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */
/*
 *  An improved steering controller for a four-wheeled vehicle
 *  Author: J. C. Allen
 *  Date: 26 March 2010
 *  Based on "General steering controller for any number of wheels and
configuration",
 *   Jordi Polo, dated 23 Dec 2007
 */

#include "Global.hh"
#include "XMLConfig.hh"
#include "Model.hh"
#include "Simulator.hh"
#include "gazebo.h"
#include "GazeboError.hh"
#include "ControllerFactory.hh"
#include "Steering_Position2d.hh"
#include "Wheel.hh"
#include "ODEBody.hh"
#include <string.h>


using namespace gazebo;

GZ_REGISTER_STATIC_CONTROLLER("steering_position2d",
Steering_Position2d);

enum {DRIVE, STEER, FULL};

//////////////////////////////////////////////////////////////////////
/////////
```

```
// Constructor
Steering_Position2d::Steering_Position2d(Entity *parent )
    : Controller(parent)
{
  this->myParent = dynamic_cast<Model*>(this->parent);

  if (!this->myParent)
    gzthrow("Steering_Position2d controller requires a Model as its
parent");

  Param::Begin(&this->parameters);
  // Load control parameters used by the steering controller
  this->velocityOffset = new ParamT<double>("velocityOffset", 0.0, 0);
  this->useSwaybars = new ParamT<bool>("useSwaybars", true, 0);
  this->swayForce = new ParamT<double>("swayForce", 300.0, 0);
  this->swayForceLimit = new ParamT<double>("swayForceLimit", 15.0, 0);
  this->useConstantVelocityMode = new
ParamT<bool>("useConstantVelocityMode", false, 0);
  this->useConstantSteeringAngleMode = new
ParamT<bool>("useConstantSteeringAngleMode", false, 0);
  this->constantSteeringAngle = new
ParamT<double>("constantSteeringAngle", 0.0, 0);
  this->useSafeVelocity = new ParamT<bool>("useSafeVelocity", true, 0);
  this->useTurnRadius = new ParamT<bool>("useTurnRadius", false, 0);
  this->turnRadius = new ParamT<double>("turnRadius", 46.1, 0);

  // Load default values for the steering controller
  this->defaultTorque = new ParamT<double>("torque", 1000.0, 0);
  this->defaultSteerTorque = new ParamT<double>("steerTorque", 1000.0,
0);

  // Load vehicle characteristics used by the steering controller
  this->tc = new ParamT<double>("turningCircle",11.491, 0);
  this->tw = new ParamT<double>("trackWidth", 1.580, 0);
  this->wb = new ParamT<double>("wheelBase", 2.619, 0);
  this->ssf = new ParamT<double>("ssf", 1.17, 0);
  this->vf = new ParamT<double>("velocityFinal", 27.778, 0);
  this->vt = new ParamT<double>("velocityFinalTime", 5.0, 0);
  this->tr = new ParamT<double>("tireRadius", 0.368, 0);
  this->sw = new ParamT<double>("sectionWidth", 0.235, 0);
  Param::End();

  this->enableMotors = true;

  this->prevUpdateTime = Simulator::Instance()->GetSimTime();

  this->sa=0;
  this->v=0;
}

//////////////////////////////////////////////////////////////////////
/////////
// Destructor
Steering_Position2d::~Steering_Position2d()
```

```
{
  delete this->velocityOffset;
  delete this->useSwaybars;
  delete this->swayForce;
  delete this->swayForceLimit;
  delete this->useConstantVelocityMode;
  delete this->useConstantSteeringAngleMode;
  delete this->constantSteeringAngle;
  delete this->useSafeVelocity;
  delete this->useTurnRadius;
  delete this->turnRadius;

  delete this->defaultTorque;
  delete this->defaultSteerTorque;

  delete this->tc;
  delete this->tw;
  delete this->wb;
  delete this->ssf;
  delete this->vf;
  delete this->vt;
  delete this->tr;
  delete this->sw;
}

////////////////////////////////////////////////////////////////////////
/////////
// Load the controller
void Steering_Position2d::LoadChild(XMLConfigNode *node)
{
  XMLConfigNode *childNode;
  std::string jointName, type;
  double torque, steerTorque;
  double g, r;

  this->myIface = dynamic_cast<PositionIface*>(this-
>GetIface("position"));

  // Load control parameters used by the steering controller
  this->velocityOffset->Load(node);
  this->useSwaybars->Load(node);
  this->swayForce->Load(node);
  this->swayForceLimit->Load(node);
  this->useConstantVelocityMode->Load(node);
  this->useConstantSteeringAngleMode->Load(node);
  this->constantSteeringAngle->Load(node);
  this->useSafeVelocity->Load(node);
  this->useTurnRadius->Load(node);
  this->turnRadius->Load(node);

  // Load default values for the steering controller
  this->defaultTorque->Load(node);
  this->defaultSteerTorque->Load(node);
  // Load vehicle characteristics used by the steering controller
  this->tc->Load(node);
  this->tw->Load(node);
  this->wb->Load(node);
  this->ssf->Load(node);
  this->vf->Load(node);
  this->vt->Load(node);
  this->tr->Load(node);
  this->sw->Load(node);

  std::cout<<"\n\nLoading the controller...\n";
  std::cout<<"  Load control parameters used by the steering
controller...\n";
  std::cout<<"    useSwaybars: "<<**this->useSwaybars<<"\n";
  if (**this->useSwaybars)
  {
  std::cout<<"      swayForce: "<<**this->swayForce<<"\n";
  std::cout<<"      swayForceLimit: "<<**this->swayForceLimit<<"\n";
  }
  std::cout<<"    useConstantVelocityMode: "<<**this-
>useConstantVelocityMode<<"\n";
  std::cout<<"    useConstantSteeringAngleMode: "<<**this-
>useConstantSteeringAngleMode<<"\n";
  if (**this->useConstantSteeringAngleMode)
    std::cout<<"      constantSteeringAngle: "<<**this-
>constantSteeringAngle<<"\n";
  std::cout<<"    useSafeVelocity: "<<**this->useSafeVelocity<<"\n";
  if (**this->useSafeVelocity)
    std::cout<<"      velocityOffset: "<<**this->velocityOffset<<"
m/s\n";
  std::cout<<"    useTurnRadius: "<<**this->useTurnRadius<<"\n";
  if (**this->useTurnRadius)
    std::cout<<"      turnRadius: "<<**this->turnRadius<<"\n";
  std::cout<<"  Load default values for the steering controller...\n";
  std::cout<<"    defaultTorque: "<<**this->defaultTorque<<"\n";
  std::cout<<"    defaultSteerTorque: "<<**this-
>defaultSteerTorque<<"\n";
  std::cout<<"  Load vehicle characteristics used by the steering
controller...\n";
  std::cout<<"    turningCircle: "<<**this->tc<<" m\n";
  std::cout<<"    trackWidth: "<<**this->tw<<" m\n";
  std::cout<<"    wheelBase: "<<**this->wb<<" m\n";
  std::cout<<"    ssf: "<<**this->ssf<<"\n";
  std::cout<<"    velocityFinal: "<<**this->vf<<" m/s\n";
  std::cout<<"    velocityFinalTime: "<<**this->vt<<" s\n";
  std::cout<<"    tireRadius: "<<**this->tr<<" m\n";
  std::cout<<"    sectionWidth: "<<**this->sw<<" m\n";
  std::cout<<"\nLoading the joints...\n";

  childNode = node->GetChild("wheel");

  while (childNode)
  {
    // Load default values for individual wheels.  These values
override the default values for the steering
    //  controller, above.
    jointName = childNode->GetString("jointName", "", 1);
    type = childNode->GetString("type", "", 1);
    torque = childNode->GetDouble("torque", **this->defaultTorque, 0);
    if (type != "drive")
      steerTorque = childNode->GetDouble("steerTorque", **this-
>defaultSteerTorque, 0);

    std::cout<<"  Loading: "<<jointName<<"\n";
    std::cout<<"    type: "<<type<<"\n";
    std::cout<<"    torque: "<<torque<<"\n";
    if (type != "drive")
      std::cout<<"    steerTorque: "<<steerTorque<<"\n";

    Wheel *wheel=new Wheel();

    if (type == "drive")
    {
      wheel->Connect(this->myParent->GetJoint(jointName), DRIVE);
      wheel->SetTorque(torque);
    }
    else
    {
      if (type == "steer")
      {
        wheel->Connect(this->myParent->GetJoint(jointName), STEER);
        wheel->SetTorque(0); // If the wheel is not full, FMax2 should
be 0 otherwise joint will lock
      }
      else
      {
        wheel->Connect(this->myParent->GetJoint(jointName), FULL);
        wheel->SetTorque(torque);
      }
      wheel->SetSteerTorque(steerTorque);
    }
    wheels.push_back(wheel);

    childNode= childNode->GetNext("wheel");
  }

  // Calculate vehicle characteristics used by the steering controller
  g = 9.80665; // acceleration due to gravity

  std::cout<<"\nCalculated vehicle characteristics used by the steering
controller...\n";

  // Calculate maximum velocity and maximum angular velocity at vehicle
center of gravity
  if (**this->useTurnRadius)
    r = **this->turnRadius + (**this->tw + **this->sw) / 2;
  else
    r = (**this->tc + **this->tw + **this->sw) / 2;
  std::cout<<"  Radius used to calculate maximum velocity, maximum
angular velocity, and maximum steering angle at vehicle center of
gravity: "<<r<<" m\n";

  wcgMax = sqrt(**this->ssf * g / r);

  if (**this->useSafeVelocity)
    vcgMax = sqrt(**this->ssf * r * g) + **this->velocityOffset;
  else
    vcgMax = **this->vf;

  std::cout<<"  Maximum velocity at vehicle center of gravity:
"<<vcgMax<<" m/s\n";
  std::cout<<"  Maximum angular velocity at vehicle center of gravity:
"<<wcgMax<<" rad/s\n";

  // Calculate maximum steering angle at vehicle center of gravity
  sacgMax = atan(**this->wb / r);

  std::cout<<"  Maximum steering angle at vehicle center of gravity:
"<<sacgMax<<" rad\n";

  // Calculate constant acceleration
  a = **this->vf / **this->vt;

  std::cout<<"  Acceleration: "<<a<<" m/s^2\n\n";

  vcg0 = 0;
  acg0 = 0;
}

//////////////////////////////////////////////////////////////////////
/////////
// Initialize the controller
void Steering_Position2d::InitChild()
{
  // Reset odometric pose
  this->odomPose[0] = 0.0;
  this->odomPose[1] = 0.0;
  this->odomPose[2] = 0.0;

  this->odomVel[0] = 0.0;
  this->odomVel[1] = 0.0;
  this->odomVel[2] = 0.0;
}

//////////////////////////////////////////////////////////////////////
/////////
// Reset the controller
void Steering_Position2d::ResetChild()
{
  // Reset odometric pose
  this->odomPose[0] = 0.0;
  this->odomPose[1] = 0.0;
  this->odomPose[2] = 0.0;

  this->odomVel[0] = 0.0;
  this->odomVel[1] = 0.0;
  this->odomVel[2] = 0.0;
}

///////////////////////////////////////////////////////////////////////
/////////
// Update the controller
void Steering_Position2d::UpdateChild()
{
  // local variables for tire radius, track width, and wheelbase
  double tr, tw, wb;
  // turning radius of wheels 1 through 4 and vehicle center of gravity
  double r1, r2, r3, r4, rcg;
  // linear distance traveled by wheels 1 through 4 and vehicle center
of gravity
  double d1 = 0.0, d2 = 0.0, d3, d4, dcg = 0.0;
  // linear distance traveled by wheels 1 and 2 (used to calculate
vehicle pose)
  double o1, o2;
  // linear velocity of wheels 1 through 4 and vehicle center of
gravity
  double v1, v2, v3, v4, vcg;
  // tangent angles of wheels 3 and 4 and vehicle center of gravity to
circles with turning radii of
  //  r3, r4, and rcg
  double a3, a4, acg = 0.0;
  // angular velocity of the STEER OR FULL wheel joints (wheels 3 and
4) or vehicle center of gravity if steering
  //  angle is zero
  double wa = 0.0;
  // angular velocity of the DRIVE wheel joints (wheels 1 and 2) or
FULL wheel joints (wheels 3 and 4) or vehicle
  //  center of gravity if steering angle is zero in the xz-plane (in
the direction of travel)
  double w = 0.0;

  Time dt;
  int count;

  tr = **this->tr;
  tw = **this->tw;
  wb = **this->wb;

  this->GetPositionCmd();

  dt = Simulator::Instance()->GetSimTime() - this->prevUpdateTime;
  this->prevUpdateTime = Simulator::Instance()->GetSimTime();

  std::vector<Wheel*>::iterator iter;

  // Calculate the current velocity at vehicle cg
  if (v >= 0)
  {
    if (v > 0.2) // we want to be able to "turn" the steering wheel
without acceleration
    {
      vcg = vcg0 + a * ( v - 0.2 ) / ( 0.5 - 0.2 ) * dt.Double();
      if (vcg > vcgMax)
        vcg = vcgMax;
    }
    else // useConstantVelocityMode controls whether the vehicle
"coasts" to a stop or maintains constant velocity
         //  when the "gas pedal" is not depressed
    {
      if (**this->useConstantVelocityMode) // maintain constant
velocity
        vcg = vcg0;
      else // "coast" to a stop
      {
        vcg = vcg0 - a * dt.Double();
        if (vcg < 0)
          vcg = 0;
      }
    }
  }
  else // v < 0
  {
    vcg = vcg0 - 3 * a * v / -0.1 * dt.Double();
    if (vcg < -vcgMax / 5)
      vcg = -vcgMax / 5;
  }

  // Calculate the distance at vehicle cg
  dcg = vcg * dt.Double();

  if (**this->useConstantSteeringAngleMode)
    sa = **this->constantSteeringAngle;

  // Calculate the angle at vehicle cg
  if (sa < -DTOR(10)) // right turn
  {
    if (sa < -sacgMax)
      sa = -sacgMax;
    if (acg > sa)
      acg = acg0 - wcgMax * dt.Double();
    if (acg <= sa)
      acg = sa;
  }
  else if (sa > DTOR(10)) // left turn
  {
    if (sa > +sacgMax)
      sa = +sacgMax;
    if (acg < sa)
      acg = acg0 + wcgMax * dt.Double();
    if (acg >= sa)
```

```
      acg = sa;
    }
    else // sa == 0
      acg = acg0;

    if ( fabs(acg) < 0.0001 ) // if fabs( acg ) < 0.0001, acg is
  effectively 0, so we set rcg greater than the diameter of the earth
      rcg = 999999999;
    else if ( fabs(acg) < 0.05 ) // if fabs( acg ) < 0.05, we use small
  angle approximation (alpha = tan(alpha))
      rcg = fabs( wb / acg );
    else
      rcg = fabs( wb / tan(acg) );

    count = 0;
    for (iter=this->wheels.begin(); iter!=this->wheels.end(); iter++)
    {
      if (this->enableMotors)
      {
        // Calculate the turning radius for each wheel
        if (sa < 0) // right turn
        {
          r1 = rcg + tw / 2;
          r2 = rcg - tw / 2;
          if (count == 0) // left_front_wheel_hinge
          {
            r3 = sqrt( r1 * r1 + wb * wb);
            a3 = acg * r3 / rcg;
            wa = a3;
            d3 = dcg * r3 / rcg;
            v3 = d3 / dt.Double();
            w = v3 / tr;
          }
          else if (count == 1) // right_front_wheel_hinge
          {
            r4 = sqrt( r2 * r2 + wb * wb);
            a4 = acg * r4 / rcg;
            wa = a4;
            d4 = dcg * r4 / rcg;
            v4 = d4 / dt.Double();
            w = v4 / tr;
          }
          else if (count == 2) // left_rear_wheel_hinge
          {
            wa = 0;
            d1 = dcg * r1 / rcg;
            v1 = d1 / dt.Double();
            w = v1 / tr;
          }
          else if (count == 3) // right_rear_wheel_hinge
          {
            wa = 0;
            d2 = dcg * r2 / rcg;
            v2 = d2 / dt.Double();
            w = v2 / tr;
          }
        }
        else if (sa > 0) // left turn
        {
          r1 = rcg - tw / 2;
          r2 = rcg + tw / 2;
          if (count == 0) // left_front_wheel_hinge
          {
            r3 = sqrt( r1 * r1 + wb * wb);
            a3 = acg * r3 / rcg;
            wa = a3;
            d3 = dcg * r3 / rcg;
            v3 = d3 / dt.Double();
            w = v3 / tr;
          }
          else if (count == 1) // right_front_wheel_hinge
          {
            r4 = sqrt( r2 * r2 + wb * wb);
            a4 = acg * r4 / rcg;
            wa = a4;
            d4 = dcg * r4 / rcg;
            v4 = d4 / dt.Double();
            w = v4 / tr;
          }
          else if (count == 2) // left_rear_wheel_hinge
          {
            wa = 0;
            d1 = dcg * r1 / rcg;
            v1 = d1 / dt.Double();
            w = v1 / tr;
          }
          else if (count == 3) // right_rear_wheel_hinge
          {
            wa = 0;
            d2 = dcg * r2 / rcg;
            v2 = d2 / dt.Double();
            w = v2 / tr;
          }
        }
        else // sa == 0
        {
          wa = 0;
          w = vcg / tr;
        }

        (*iter)->Update(-w, -wa, **this->updatePeriodP);

        if (**this->useSwaybars)
        {
          Swaybars();
        }

        count += 1;
```

```
    }
    else
    {
      (*iter)->Stop();
    }
  }

  vcg0 = vcg;
  acg0 = acg;

  o1 = dt.Double() * tr * this->myParent-
>GetJoint("left_rear_wheel_hinge")->GetVelocity(0);
  o2 = dt.Double() * tr * this->myParent-
>GetJoint("right_rear_wheel_hinge")->GetVelocity(0);

  // Compute odometric pose
  this->odomPose[0] += (o1 + o2) / 2 * cos( this->odomPose[2] );
  this->odomPose[1] += (o1 + o2) / 2 * sin( this->odomPose[2] );
  this->odomPose[2] += (o1 - o2) / tw;

  // Compute odometric instantaneous velocity
  this->odomVel[0] = (o1 + o2) / 2 / dt.Double();
  this->odomVel[1] = 0.0;
  this->odomVel[2] = (o1 - o2) / tw / dt.Double();

  this->PutPositionData();
}

//////////////////////////////////////////////////////////////////////
/////////
// Finalize the controller
void Steering_Position2d::FiniChild()
{
}

//////////////////////////////////////////////////////////////////////
///////
// Get commands from the external interface
void Steering_Position2d::GetPositionCmd()
{
  if (this->myIface->Lock(1))
  {
    this->v = this->myIface->data->cmdVelocity.pos.x;
    this->sa = this->myIface->data->cmdVelocity.yaw;

    this->enableMotors = this->myIface->data->cmdEnableMotors > 0;

    this-myIface->Unlock();
  }
}

//////////////////////////////////////////////////////////////////////
///////
// Update the data in the interface
```

```
void Steering_Position2d::PutPositionData()
{
  if (this->myIface->Lock(1))
  {
    // TODO: Data timestamp
    this->myIface->data->head.time = Simulator::Instance()-
>GetSimTime().Double();

    this->myIface->data->pose.pos.x = this->odomPose[0];
    this->myIface->data->pose.pos.y = this->odomPose[1];
    this->myIface->data->pose.yaw = NORMALIZE(this->odomPose[2]);

    this->myIface->data->velocity.pos.x = this->odomVel[0];
    this->myIface->data->velocity.yaw = this->odomVel[2];

    // TODO
    this->myIface->data->stall = 0;

    this-myIface->Unlock();
  }
}

//////////////////////////////////////////////////////////////////////
///////
// "Anti-sway bar" implementation
void Steering_Position2d::Swaybars()
{
  Vector3 wheelAnchor;
  Vector3 bodyAnchor;
  Vector3 axis;
  Vector3 force;
  double displacement, amt;

  std::string hinge[4];

  hinge[0] = "left_front_wheel_hinge";
  hinge[1] = "right_front_wheel_hinge";
  hinge[2] = "left_rear_wheel_hinge";
  hinge[3] = "right_rear_wheel_hinge";

  for (int i = 0; i < 4; i++)
  {
    bodyAnchor = this->myParent->GetJoint(hinge[i])->GetAnchor(0);
    wheelAnchor = this->myParent->GetJoint(hinge[i])->GetAnchor(1);
    axis = this->myParent->GetJoint(hinge[i])->GetAxis(1);

    displacement = (bodyAnchor.z - wheelAnchor.z) * axis.z;
    if (displacement > 0)
    {
      amt = displacement * **this->swayForce;

      if (amt > **this->swayForceLimit)
      {
        amt = **this->swayForceLimit;
      }
      // "downforce"
      force.Set(-axis.x * amt, -axis.y * amt, -axis.z * amt);
      this->myParent->GetJoint(hinge[i])->GetJointBody(1)-
>SetForce(force);
      // "upforce"
      force.Set(axis.x * amt, axis.y * amt, axis.z * amt);
      this->myParent->GetJoint(hinge[i^1])->GetJointBody(1)-
>SetForce(force);
    }
  }
}
```

```
/*
 *  Gazebo - Outdoor Multi-Robot Simulator
 *  Copyright (C) 2003
 *     Nate Koenig & Andrew Howard
 *
 *  This program is free software; you can redistribute it and/or
modify
 *  it under the terms of the GNU General Public License as published
by
 *  the Free Software Foundation; either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program; if not, write to the Free Software
 *  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-
1307  USA
 *
 */
/*
 *  An improved wheel for a four-wheeled vehicle
 *  Author: J. C. Allen
 *  Date: 26 March 2010
 *  Based on "Wheel that can not be steered",
 *   Jordi Polo, dated 18 Dec 2007
 */

#include "Global.hh"
#include "XMLConfig.hh"
#include "Model.hh"
#include "Body.hh"
#include "Joint.hh"
#include "World.hh"
#include "gazebo.h"
#include "GazeboError.hh"
#include "ControllerFactory.hh"
#include "Steering_Position2d.hh"
#include "Wheel.hh"
#include <string>

using namespace gazebo;

enum {DRIVE, STEER, FULL};

///////////////////////////////////////////////////////////////////////
/////////
// Constructor
Wheel::Wheel()
{
}
```

```
///////////////////////////////////////////////////////////////////////
/////////
// Destructor
Wheel::~Wheel()
{
  delete this->joint;
}

///////////////////////////////////////////////////////////////////////
/////////
// Connects the wheel to a given Joint
void Wheel::Connect(Joint *joint, int type)
{
  this->joint = joint;
  this->type = type;

  if (!this->joint)
  {
    std::ostringstream stream;
    stream << "The controller couldn't get the joint " <<this->joint-
>GetName();
    gzthrow(stream.str());
  }

  // avoid an initial impulse to the joints that would make the vehicle
flip
  this->joint->SetAttribute(Joint::FUDGE_FACTOR, 0, 0.1);
}

///////////////////////////////////////////////////////////////////////
/////////
// Stops the wheel
void Wheel::Stop()
{
  switch (this->type)
  {
    case DRIVE:
      this->joint->SetVelocity(0, 0);
      this->joint->SetMaxForce(0, 0);
      break;
    case STEER:
      this->joint->SetVelocity(0, 0);
      this->joint->SetMaxForce(0, 0);
      this->joint->SetVelocity(1, 0);
      this->joint->SetMaxForce(1, 0);
      break;
    default:
      this->joint->SetVelocity(0, 0);
      this->joint->SetMaxForce(0, 0);
      this->joint->SetVelocity(1, 0);
      this->joint->SetMaxForce(1, 0);
  }
}
```

```
///////////////////////////////////////////////////////////////////////
/////////
// Set the torque
void Wheel::SetTorque(double newTorque)
{
  this->torque = newTorque;

  switch (this->type)
  {
    case DRIVE:
      this->joint->SetMaxForce(0, this->torque);
      break;
    case STEER:
      this->joint->SetMaxForce(1, this->torque);
      break;
    default:
      this->joint->SetMaxForce(1, this->torque);
  }
}

///////////////////////////////////////////////////////////////////////
/////////
// Get the torque
double Wheel::GetTorque()
{
  return this->torque;
}

///////////////////////////////////////////////////////////////////////
/////////
// Set the steering torque
void Wheel::SetSteerTorque(double newTorque)
{
  this->steerTorque = newTorque;

  switch (this->type)
  {
    case DRIVE: // drive wheels have no steering axis
      break;
    case STEER:
      this->joint->SetMaxForce(0, this->steerTorque);
      break;
    default:
      this->joint->SetMaxForce(0, this->steerTorque);
  }
}

///////////////////////////////////////////////////////////////////////
/////////
// Get the steering torque
double Wheel::GetSteerTorque()
{
  return this->steerTorque;
}

//////////////////////////////////////////////////////////////////////
/////////
// Update the wheel
void Wheel::Update(double speed, double steer, double rate)
{
  switch (this->type)
  {
    case DRIVE:
      this->joint->SetVelocity(0, speed);
      break;
    case STEER:
      this->joint->SetVelocity(0, rate * (steer - this->joint-
>GetAngle(0).GetAsRadian()));
      break;
    default:
      this->joint->SetVelocity(0, rate * (steer - this->joint-
>GetAngle(0).GetAsRadian()));
      this->joint->SetVelocity(1, speed);
  }
}
```

Appendix E: Example Output Produced During Controller Validation

Only the output produced by the controller is included here. Output produced by Gazebo is not included, with the exceptions of the first line showing the version number and the line indicating Gazebo was successfully initialized:

```
Gazebo multi-robot simulator, version 0.10.0

Loading the controller...
  Load control parameters used by the steering controller...
    useSwaybars: 0
    useConstantVelocityMode: 1
    useConstantSteeringAngleMode: 1
      constantSteeringAngle: -0.376337
    useSafeVelocity: 1
      velocityOffset: 0 m/s
    useTurnRadius: 0
  Load default values for the steering controller...
    defaultTorque: 1000
    defaultSteerTorque: 1000
  Load vehicle characteristics used by the steering controller...
    turningCircle: 11.491 m
    trackWidth: 1.529 m
    wheelBase: 2.619 m
    ssf: 1.17
    velocityFinal: 27.778 m/s
    velocityFinalTime: 5 s
    tireRadius: 0.368 m
    sectionWidth: 0.235 m

Loading the joints...
  Loading: left_front_wheel_hinge
    type: full
    torque: 10000
    steerTorque: 10000
  Loading: right_front_wheel_hinge
    type: full
    torque: 10000
    steerTorque: 10000
  Loading: left_rear_wheel_hinge
    type: drive
    torque: 10000
  Loading: right_rear_wheel_hinge
    type: drive
    torque: 10000

Calculated vehicle characteristics used by the steering controller...
  Radius used to calculate maximum velocity, maximum angular velocity,
and maximum steering angle at vehicle center of gravity: 6.6275 m
  Maximum velocity at vehicle center of gravity: 8.72023 m/s
  Maximum angular velocity at vehicle center of gravity: 1.31577 rad/s
  Maximum steering angle at vehicle center of gravity: 0.376337 rad
  Acceleration: 5.5556 m/s^2
```

```
Gazebo successfully initialized
```

Appendix F: Example World File and Model File Used During Evaluation of 2004 GCE Course Segment 2570-2571-2572

```
<?xml version="1.0"?>

<gazebo:world
  xmlns:xi="http://www.w3.org/2001/XInclude"
  xmlns:gazebo="http://playerstage.sourceforge.net/gazebo/xmlschema/#gz
"
  xmlns:model="http://playerstage.sourceforge.net/gazebo/xmlschema/#mod
el"
  xmlns:sensor="http://playerstage.sourceforge.net/gazebo/xmlschema/#se
nsor"
  xmlns:window="http://playerstage.sourceforge.net/gazebo/xmlschema/#wi
ndow"
  xmlns:param="http://playerstage.sourceforge.net/gazebo/xmlschema/#par
am"
  xmlns:body="http://playerstage.sourceforge.net/gazebo/xmlschema/#body
"
  xmlns:geom="http://playerstage.sourceforge.net/gazebo/xmlschema/#geom
"
  xmlns:joint="http://playerstage.sourceforge.net/gazebo/xmlschema/#joi
nt"
  xmlns:interface="http://playerstage.sourceforge.net/gazebo/xmlschema/
#interface"
  xmlns:ui="http://playerstage.sourceforge.net/gazebo/xmlschema/#ui"
  xmlns:rendering="http://playerstage.sourceforge.net/gazebo/xmlschema/
#rendering"
  xmlns:controller="http://playerstage.sourceforge.net/gazebo/xmlschema
/#controller"
  xmlns:physics="http://playerstage.sourceforge.net/gazebo/xmlschema/#p
hysics" >

  <verbosity>5</verbosity>

  <physics:ode>
    <stepTime>0.001</stepTime>
    <gravity>0 0 -9.80665</gravity>
    <cfm>10e-5</cfm>
    <erp>0.8</erp>
    <!-- updateRate: <0 == throttle simTime to match realTime.
                      0 == No throttling
                     >0 == Frequency at which to throttle the sim -->
    <updateRate>0</updateRate>
  </physics:ode>

  <rendering:gui>
    <type>fltk</type>
    <size>640 480</size>
    <pos>0 0</pos>
  </rendering:gui>

  <rendering:ogre>
    <ambient>0.4 0.4 0.4 1.0</ambient>
    <sky>
      <material>Gazebo/CloudySky</material>
    </sky>
    </rendering:ogre>

   <!-- Ground Plane -->
  <model:physical name="plane1_model">
    <xyz>0 0 0</xyz>
    <rpy>0 0 0</rpy>
    <static>true</static>
    <body:plane name="plane1_body">
      <geom:plane name="plane1_geom">
        <normal>0 0 1</normal>
        <size>2000 2000</size>
        <segments>10 10</segments>
        <uvTile>100 100</uvTile>
        <material>Gazebo/GrassFloor</material>
        <visual>
          <rpy>0 0 0</rpy>
          <mesh>models/02.7.924.mesh</mesh>
          <scale>1 1 1</scale>
          <material>Gazebo/Grey</material>
        </visual>
      </geom:plane>
    </body:plane>
  </model:physical>

  <!-- The camera -->
  <model:physical name="cam1_model">
    <xyz>0 0 20</xyz>
    <rpy>0 0 180</rpy>
    <static>true</static>
    <body:empty name="cam1_body">
      <sensor:camera name="cam1_sensor">
        <nearClip>0.1</nearClip>
        <farClip>100</farClip>
<!-- not in use, but named parameter
        <saveFrames>false</saveFrames>
        <saveFramePath>frames</saveFramePath>
-->
        <imageSize>640 480</imageSize>
<!-- not in use, but named parameter
        <mask></mask>
-->
        <hfov>60</hfov>
<!-- allowed image formats are: L8, R8G8B8, B8G8R8 ref: OgreCamera.cc
-->
        <imageFormat>R8G8B8</imageFormat>
<!-- not in use, but named parameter
        <updateRate></updateRate>
-->
        <controller:generic_camera name="camera_controller">
          <interface:camera name="camera_iface_0"/>
        </controller:generic_camera>
      </sensor:camera>
    </body:empty>
  </model:physical>
  <model:physical name="cv_model">
    <xyz>0.0 3.072 0.1</xyz>
    <rpy>0.0 0.0 0.0</rpy>
    <static>false</static>
<!--
The include should be last within a model. All previous statements
will override those in the included file
-->
    <include embedded="true">
      <xi:include href="models/cv.model" />
    </include>
  </model:physical>

  <!-- White Directional light -->
  <model:renderable name="directional_white">
    <static>true</static>
    <light>
      <type>directional</type>
      <direction>0 -0.8 -0.3</direction>
      <diffuseColor>0.9 0.9 0.9</diffuseColor>
      <specularColor>0.0 0.0 0.0</specularColor>
      <range>100</range>
      <!-- Constant(0-1) Linear(0-1) Quadratic -->
      <attenuation>0.0 1.0 0.4</attenuation>
    </light>
  </model:renderable>

</gazebo:world>
```

```
<?xml version="1.0"?>

<!-- Challenge Vehicle Model -->
<model:physical name="cv_model"
  xmlns:model="http://playerstage.sourceforge.net/gazebo/xmlschema/#mod
el"
  xmlns:sensor="http://playerstage.sourceforge.net/gazebo/xmlschema/#se
nsor"
  xmlns:body="http://playerstage.sourceforge.net/gazebo/xmlschema/#body
"
  xmlns:geom="http://playerstage.sourceforge.net/gazebo/xmlschema/#geom
"
  xmlns:joint="http://playerstage.sourceforge.net/gazebo/xmlschema/#joi
nt"
  xmlns:controller="http://playerstage.sourceforge.net/gazebo/xmlschema
/#controller"
  xmlns:interface="http://playerstage.sourceforge.net/gazebo/xmlschema/
#interface"
  xmlns:visual="http://playerstage.sourceforge.net/gazebo/xmlschema/#vi
sual"
>

<!--
  The following conventions are used herein:
   SI units were used to model the challenge vehicle and create the
mesh.
   "Length" refers to dimensions along the x-axis.
   "Width" refers to dimensions along the y-axis.
   "Height" refers to dimensions along the z-axis.

  The overall dimensions of the Team 2005-06 challenge vehicle were:
    Length:                   174.9 in (4.442 m)
    Width:                     70.1 in (1.780 m)
    Height:                    70.4 in (1.788 m)
    Track width (front):       61.1 in (1.552 m)
    Track width (rear):        60.2 in (1.529 m)
    Bumper to front axle:      34.1 in (0.866 m)
    Wheelbase:                103.1 in (2.619 m)
    Rear axle to end of frame:  37.7 in (0.958 m)
    Ground clearance:          10.0 in (0.254 m)
      The chassis_body visual is located 0.495 m from the ground, to
compensate for ground clearance and an error
        of 0.894 - 0.653 = 0.241 m in distance from the ground
    Curb weight:             3792.0 lb (1720.0 kg)

  The stock tires on the Team 2005-06 challenge vehicle were
"P235/70TR16.0 BSW AS" tires.  Team 2005-06
   replaced the stock tires on their challenge vehicle with: "off-road
tires that provide an extra inch of
   clearance.  The new tires also have reinforced sidewalls and thicker
tread to help prevent flat tires
   due to the rocky terrain."  However, Team 2005-06 provided no
additional identifying information for
   the tires in use by the team.  The author used the dimensions of the
```

Team 2005-06 challenge vehicle
stock tires herein, and did not alter the ground clearance of the challenge vehicle.

The overall dimensions of the Team 2005-06 challenge vehicle tires were:

Section width:	9.3 in (0.235 m)
Sidewall height (from rim to tread):	6.5 in (0.165 m)
Rim diameter:	16.0 in (0.406 m)
Tire radius:	14.5 in (0.368 m)

Sidewall height (from rim to tread) is equal to 70 percent of the section width. The sidewall aspect ratio
for the Team 2005-06 challenge vehicle tires was 70.

Tire radius is equal to one-half the rim diameter plus the sidewall height (from rim to tread).

Miscellaneous dimensions calculated to model the Team 2005-06 challenge vehicle:

Chassis body width:	47.9 in (1.218 m)
Chassis body height:	12.0 in (0.305 m)
anchorOffset (rear):	1.5 in (0.038 m)
anchorOffset (front):	2.0 in (0.050 m)
Height of the center of gravity:	25.7 in (0.653 m)
Front axle to center of gravity x-dimension:	53.4 in (1.356 m)
Center of gravity to rear axle x-dimension:	49.8 in (1.265 m)
Front and rear axle to center of gravity z-dimension:	20.7 in (0.526 m)
Wheel well radius:	20.0 in (0.508 m)
xy-origin to base of hood z-dimension (three-sevenths overall height):	(0.766 m)
xy-origin to base of windshield z-dimension (four-sevenths overall height):	(1.022 m)
xy-origin to front and rear axle z-dimension:	6.0 in (0.152 m)
yz-origin to base of windshield x-dimension:	54.1 in (1.374 m)
yz-origin to base of roof x-dimension:	74.1 in (1.882 m)
yz-origin to rear axle x-dimension:	140.8 in (3.576 m)
Roof length:	100.8 in (2.560 m)
Front axle to end of frame x-dimension:	140.8 in (3.576 m)

The width of the chassis body is equal to the track width (rear) minus section width minus twice the anchorOffset
(rear). An arbitrary anchorOffset (rear) of 1.5 inches (0.0381 m) was selected.

```
  The anchorOffset (front) is equal to one-half the track width (front)
minus one-half the width of the chassis
   body, minus one-half the tire width.
  The z-dimension of the anchorOffset is equal to the height of the
center of gravity minus tire radius.
  The center of gravity of the model is located at the center of mass
of the chassis body.  Tires have been no mass
   as a result.
  The height of the center of gravity is equal to track width (rear)
divided by twice the vehicle's static stability
   factor.
  Roof length is equal to chassis length minus the yz-origin to base of
roof x-dimension.
  The xy-origin to front and rear axle z-dimension is equal to tire
radius minus ground clearance.
  Front axle to end of frame x-dimension is equal to wheelbase plus
rear axle to end of frame.
  Bumper to axle is equal to the chassis length minus wheelbase minus
rear axle to end of frame, or
   chassis length minus front axle to end of frame x-dimension.
-->

  <xyz>0 0 0</xyz>
  <rpy>0 0 0</rpy>
  <canonicalBody>chassis_body</canonicalBody>
  <controller:steering_position2d name="steering_controller">
    <updateRate>50</updateRate>
    <wheel>
      <jointName>left_front_wheel_hinge</jointName>
      <type>full</type>
      <torque>10000</torque>
      <steerTorque>10000</steerTorque>
    </wheel>
    <wheel>
      <jointName>right_front_wheel_hinge</jointName>
      <type>full</type>
      <torque>10000</torque>
      <steerTorque>10000</steerTorque>
    </wheel>
    <wheel>
      <jointName>left_rear_wheel_hinge</jointName>
      <type>drive</type>
      <torque>10000</torque>
    </wheel>
    <wheel>
      <jointName>right_rear_wheel_hinge</jointName>
      <type>drive</type>
      <torque>10000</torque>
    </wheel>
    <useSwaybars>false</useSwaybars>
    <swayForce>300</swayForce>
    <swayForceLimit>15</swayForceLimit>
    <useConstantVelocityMode>true</useConstantVelocityMode>
    <useConstantSteeringAngleMode>false</useConstantSteeringAngleMode>
    <constantSteeringAngle>0.314852</constantSteeringAngle>
    <useSafeVelocity>true</useSafeVelocity>
    <velocityOffset>2.5</velocityOffset>
    <useTurnRadius>true</useTurnRadius>
    <turnRadius>46.1</turnRadius>
    <turningCircle>11.491</turningCircle>
    <trackWidth>1.529</trackWidth>
    <wheelBase>2.619</wheelBase>
    <ssf>1.17</ssf>
    <velocityFinal>26.822</velocityFinal>
    <velocityFinalTime>5.0</velocityFinalTime>
    <tireRadius>0.368</tireRadius>
    <sectionWidth>0.235</sectionWidth>
    <interface:position name="position_iface_0"/>
  </controller:steering_position2d>
  <body:box name="chassis_body">
    <geom:box name="chassis_geom">
      <xyz>0 0 0.653</xyz>
      <size>4.442 1.218 0.305</size>
      <mass>1720</mass>
      <visual>
        <mesh>unit_box</mesh>
        <scale>4.242 1.218 0.305</scale>
        <material>Gazebo/Green</material>
      </visual>
      <visual>
        <xyz>0 0 0.495</xyz>
        <rpy>0 0 0</rpy>
        <mesh>../../Media/models/cv.mesh</mesh>
        <material>Gazebo/Pioneer2Body</material>
        <scale>1 1 1</scale>
      </visual>
    </geom:box>
  </body:box>
  <body:cylinder name="left_front_wheel">
    <xyz>1.355 0.776 0.368</xyz>
    <rpy>90 0 0</rpy>
    <geom:cylinder name="left_front_wheel_geom">
      <size>0.368 0.235</size>
      <visual>
        <mesh>../../Media/models/Pioneer2at/tire.mesh</mesh>
        <rpy>-90 0 0</rpy>
        <size>0.736 0.235 0.736</size>
        <material>Gazebo/Black</material>
      </visual>
      <visual>
        <mesh>../../Media/models/Pioneer2at/wheel.mesh</mesh>
        <rpy>-90 0 0</rpy>
        <size>0.736 0.235 0.736</size>
        <material>Gazebo/Gold</material>
      </visual>
    </geom:cylinder>
  </body:cylinder>
  <body:cylinder name="right_front_wheel">
```

```
    <xyz>1.355 -0.776 0.368</xyz>
    <rpy>-90 0 0</rpy>
    <geom:cylinder name="right_front_wheel_geom">
      <size>0.368 0.235</size>
      <visual>
        <mesh>../../Media/models/Pioneer2at/tire.mesh</mesh>
        <rpy>-90 0 0</rpy>
        <size>0.736 0.235 0.736</size>
        <material>Gazebo/Black</material>
      </visual>
      <visual>
        <mesh>../../Media/models/Pioneer2at/wheel.mesh</mesh>
        <rpy>-90 0 0</rpy>
        <size>0.736 0.235 0.736</size>
        <material>Gazebo/Gold</material>
      </visual>
    </geom:cylinder>
  </body:cylinder>
  <body:cylinder name="left_rear_wheel">
    <xyz>-1.264 0.765 0.368</xyz>
    <rpy>90 0 0</rpy>
    <finiteRotationMode>0</finiteRotationMode>
    <finiteRotationAxis>0 1 0</finiteRotationAxis>
    <geom:cylinder name="left_rear_wheel_geom">
      <size>0.368 0.235</size>
      <visual>
        <mesh>../../Media/models/Pioneer2at/tire.mesh</mesh>
        <rpy>-90 0 0</rpy>
        <size>0.736 0.235 0.736</size>
        <material>Gazebo/Black</material>
      </visual>
      <visual>
        <mesh>../../Media/models/Pioneer2at/wheel.mesh</mesh>
        <rpy>-90 0 0</rpy>
        <size>0.736 0.235 0.736</size>
        <material>Gazebo/Gold</material>
      </visual>
    </geom:cylinder>
  </body:cylinder>
  <body:cylinder name="right_rear_wheel">
    <xyz>-1.264 -0.765 0.368</xyz>
    <rpy>-90 0 0</rpy>
    <finiteRotationMode>0</finiteRotationMode>
    <finiteRotationAxis>0 1 0</finiteRotationAxis>
    <geom:cylinder name="right_rear_wheel_geom">
      <size>0.368 0.235</size>
      <visual>
        <mesh>../../Media/models/Pioneer2at/tire.mesh</mesh>
        <rpy>-90 0 0</rpy>
        <size>0.736 0.235 0.736</size>
        <material>Gazebo/Black</material>
      </visual>
      <visual>
        <mesh>../../Media/models/Pioneer2at/wheel.mesh</mesh>
        <rpy>-90 0 0</rpy>
        <size>0.736 0.235 0.736</size>
        <material>Gazebo/Gold</material>
      </visual>
    </geom:cylinder>
  </body:cylinder>
  <joint:hinge2 name="left_front_wheel_hinge">
    <body1>chassis_body</body1>
    <body2>left_front_wheel</body2>
    <anchor>left_front_wheel</anchor>
    <axis1>0 0 1</axis1>
    <axis2>0 1 0</axis2>
    <erp>0.8</erp>
    <cfm>10e-5</cfm>
  </joint:hinge2>
  <joint:hinge2 name="right_front_wheel_hinge">
    <body1>chassis_body</body1>
    <body2>right_front_wheel</body2>
    <anchor>right_front_wheel</anchor>
    <axis1>0 0 1</axis1>
    <axis2>0 1 0</axis2>
    <erp>0.8</erp>
    <cfm>10e-5</cfm>
  </joint:hinge2>
  <joint:hinge name="left_rear_wheel_hinge">
    <body1>chassis_body</body1>
    <body2>left_rear_wheel</body2>
    <anchor>left_rear_wheel</anchor>
    <axis>0 1 0</axis>
    <erp>0.8</erp>
    <cfm>10e-5</cfm>
  </joint:hinge>
  <joint:hinge name="right_rear_wheel_hinge">
    <body1>chassis_body</body1>
    <body2>right_rear_wheel</body2>
    <anchor>right_rear_wheel</anchor>
    <axis>0 1 0</axis>
    <erp>0.8</erp>
    <cfm>10e-5</cfm>
  </joint:hinge>
</model:physical>
```

Appendix G: Miscellaneous Problems Encountered

I. PROBLEMS ENCOUNTERED WHILE VERIFYING PLAYER AND GAZEBO

I.A. Gazebo "`Namespace prefix ... is not defined`" errors

While attempting to verify Player and Gazebo using the packaged "simplecar" model, the author encountered a number of "`Namespace prefix ... is not defined`" errors.

Through review of the Gazebo mailing list archives, the author determined these errors were caused by missing `<xmlns>` declarations at the beginning of the file defining the packaged simplecar model, file `simplecar.model`. Including these declarations resolved the problem.

The author was unable to determine why some users of Gazebo were able to load this model without modification, as reported on the Gazebo mailing list, or why the packaged simplecar model was not revised to correct the problem when it was first reported on September 9, 2009. The author submitted patch number 2934729 to resolve the "`Namespace prefix ... is not defined`" errors on January 29, 2010.

I.B. Player "`Unhandled message for driver device`" error

While attempting to verify Player and Gazebo using the packaged examples, the author encountered an "`Unhandled message for driver device`" error. While this error occurred, the simplecar model would move slightly, then suddenly come to a stop, with unpredictable results. Sometimes the rear end of the vehicle would bounce into the air, and sometimes the vehicle would simply stop. After coming to rest, the model would no longer respond to commands. The author eventually identified the cause of this problem. See paragraph I.E.

While searching the Gazebo source code to determine the source of the error, the author noted the svn version of Gazebo revision 8443 included two copies of an 11.8 MB file in directories "`examples/player/ptz/.player`" and "`examples/player/ptz/.svn/text-base/.player.svn-base`" in which this error is repeated thousands of times. The compressed size of the Gazebo source distribution ("`gazebo-0.9.0.tar.bz2`") originally downloaded by the author, and which would not compile, was 17.4 MB. The copy of this file in directory "`examples/player/ptz/.svn/text-base/.player.svn-base`" was not included in the source distribution, but the copy in directory "`examples/player/ptz/.player`" was included in the source distribution.

I.C. ODE "`bNormalizationResult`" error

While attempting to verify Player and Gazebo using the packaged simplecar model, the author encountered the following ODE error:

```
ODE INTERNAL ERROR 1: assertion "bNormalizationResult"
failed in _dNormalize3()
[../../../include/ode/odemath.h] Aborted
```

Based on a review of ODE documentation ([44]), the author proposed this problem was related to the order of bodies in "hinge2" `<joint>` declarations in file `simplecar.model`, specifically a mismatch between bodies and the `<anchor>` declaration of the joint. ODE documentation states function `dJointGetHinge2Anchor`, corresponding to the file `simplecar.model` `<anchor>` declaration "returns the point on body 1", and that function `dJointGetHinge2Anchor2` "returns the point on body 2". However, in the packaged version of file `simplecar.model`, the `<anchor>` declaration refers to the "left_front_wheel" and the `<body1>` declaration refers to the "chassis_body". The author therefore reversed the order of the bodies in the `<joint>` declaration.

When the order of the bodies was reversed, the model would load successfully. The author submitted patch number 2934693 to reverse the order of the bodies in the `<joint>` declaration on January 19, 2010. However, the author was unable to interact with the model using the `playerv` utility at this time.

While attempting to resolve this problem, the author posted a message reporting the above to the Gazebo mailing list, and received the following reply: "I've seen bNormalizationResult in the past due to float/double inconsistency between ode and gazebo." ([101]). As a result, the author uninstalled ODE in accordance with Appendix C and re-installed ODE in accordance with Appendix B using the "`--enable-double-precision`" and "`--enable-demos`" flags. The use of the "`--enable-double-precision`" flag to enable double precision in ODE resulted in errors when attempting to run Gazebo. The author then uninstalled ODE in accordance with Appendix C and re-installed ODE in accordance with Appendix B using the "`--enable-demos`" flag and confirmed that the ODE demos would run without errors.

Finally, the author uninstalled ODE in accordance with Appendix C and re-installed ODE in accordance with Appendix B using the "`--enable-demos`" and "`--disable-asserts`" flags to disable assertion checking. The author was unable to determine why installation instructions for Gazebo do not require the use of the "`--disable-asserts`" flag when compiling ODE for use with Gazebo. As a result, the author's only conclusion is that a default installation of Gazebo using a default installation of ODE will fail due to ODE assertions when attempting to load the packaged simplecar model.

I.D. Playerv "Devices>position2d" menu

As noted in paragraph I.C., the author was unable to interact with the packaged simplecar model using the `playerv` utility. The simplecar model would move slightly, then suddenly come to a stop, with unpredictable results. Sometimes the rear end of the vehicle would bounce into the air, and sometimes the vehicle would simply stop. After coming to rest, the model would no longer respond to commands.

The author initially proposed the problem was due to the interaction between the disabling of a menu item via function `rtk_menuitem_isactivated` in file `rtk_menu.c` and function `position2d_create` in file `pv_dev_position2d.c`. File `pv_dev_position2d.c` creates menu items "Enable" and "Disable" on the "Devices>position2d" menu in the `playerv` utility which appeared to be disabled by default (i.e., no checkbox is available to enable or disable the position2d interface), and file `rtk_menu.c` appeared to disable menu items as soon as they were enabled via the following lines:

```
if (item->activated)
{
  item->activated = FALSE;
  return TRUE;
}
```

The author revised file `pv_dev_position2d.c` to enable the menu items to be enabled (i.e., to display a checkbox indicating their status). The author was then able to verify the device was enabled as expected and remained enabled while attempting to command the simplecar model using the `playerv` utility.

Following resolution of the Gazebo `ODEHingeJoint.cc` error (see paragraph I.E.), the author discovered the menu items “Enable” and “Disable” on the “Devices>position2d” menu in the `playerv` utility can be enabled despite not having an associated checkbox and that the `playerv` utility cannot be used to command a model using the position2d interface unless the space next to the “Enable” interface option in the “Devices>position2d” menu, where a checkbox would be if the “Enable” interface option were enabled, is first clicked on. The author was unable to determine why the “Enable” and “Disable” options appear to be disabled.

I.E. Gazebo “`ODEHingeJoint.cc`” error

As noted in paragraph I.C., the author was unable to interact with the simplecar model using the `playerv` utility. Initially, the author proposed the cause of the problem was the disabling of the position2d interface as soon as it was enabled. The author determined that although a problem exists (to enable the position2d interface, users must click a non-existent checkbox next to “Enable” in the “Devices>position2d” menu), the problem did not cause the observed behavior. See paragraph I.D.

The author has been monitoring the “playerstage-developers”, “playerstage-gazebo", and "playerstage-users" mailing list since beginning this research. Another user identified one of the two causes of this problem the author has been able to verify, and submitted bug report number 2933700 on January 17, 2010 to report it to the Player Project. The order of parameters passed to function "SetParam" in file `ODEHingeJoint.cc`, function:

```
void ODEHingeJoint::SetVelocity(int /*index*/, double
angle)
{
  this->SetParam(angle, dParamVel);
}
```

was reversed. The function should be:

```
void ODEHingeJoint::SetVelocity(int /*index*/, double
angle)
{
  this->SetParam(dParamVel, angle);
}
```

The author reversed the order of parameters as described and was able to verify Player and Gazebo using the packaged examples, specifically the simplecar model.

II. PROBLEMS ENCOUNTERED WHILE VALIDATING THE IMPROVED CONTROLLER

II.A. Gazebo `ODEHinge2Joint::GetAngle` and `GetVelocity` problem

While validating the improved controller, the author encountered a problem when setting the angular velocity of the steering wheels around the steering axis. Under certain circumstances, the angular velocity of a steering wheel, typically the outside wheel, would increase suddenly due to a large difference between the target steering angle and current steering angle. This caused the steering wheel to suddenly turn at high speed around the steering axis. The effect of friction would then cause the wheel to appear to "dig in" and throw the rear of the model into the air.

Although this behavior is consistent with expectations from the perspective of physical realism, the purpose of the improved controller was to effectively limit the maximum angular velocity at model CG to prevent the model from being able to turn at a rate faster than allowed by representative challenge vehicle and course geometry and thereby prevent rollover. The purpose of the improved controller was to prevent exactly this problem.

The angular velocity of each steering wheel around the steering axis is determined by the difference between the target steering angle and current steering angle of the wheel and the update rate of the improved controller. Multiplying the difference by the update rate yields the angular velocity which must be applied to reach the target steering angle in one controller time step. The controller limits the angular velocity to the calculated angular velocity or maximum angular velocity at model CG, whichever is less.

The author originally proposed this problem was caused by Gazebo function `ODEHinge2Joint::GetAngle`, which is a wrapper around ODE function `dJointGetHinge2Angle1`, which itself is a wrapper around ODE function `dxJointHinge2::measureAngle`. This function returns the value of the C programming language function `atan2`, which is undefined if both arguments are equal to zero. The author noted that the problem occurred at approximately the same time and position in each trial, and proposed it was related to the difference between the target steering angle and current steering angle of the steering wheel approaching zero because the value returned by function `ODEHinge2Joint::GetAngle` increased suddenly as the current steering angle approached the target steering angle in each trial.

The author attempted several solutions to this problem: revising the time step, reducing the mass of the representative challenge vehicle, and setting the velocity of the chassis body directly during each time step. Each proposed solution was rejected as unrealistic or problematic.

The author then revised improved controller function `Wheel::Update` to store the last known value returned by function `ODEHinge2Joint::GetAngle` in a private class member variable and use it in lieu of the value returned by the function itself when the current steering angle approached the target steering angle, but this did not resolve the problem. As a result, the author concluded function `ODEHinge2Joint::GetAngle` was not the cause of the problem and that the sudden increase in angular velocity observed was an indirect effect of another problem.

The author determined the problem was caused by a combination of inaccurate maximum angular velocity determination and insufficient torque. Diagnosing the problem was made more difficult by lack of a return value from Gazebo function `ODEHinge2Joint::GetVelocity`.

The packaged controller subtracted the angular velocity returned by function `ODEHinge2Joint::GetVelocity` from the target velocity determined by the controller every time the controller was updated. However, it is unclear if this was ever successful because this function had no return value.

After revising function `ODEHinge2Joint::GetVelocity` to return the angular velocity of the joint the author was able to use this function to determine that the wheel joints were unable to reach the target angular velocities set by the improved controller before the next controller update. As a result, the difference between the target angle and current angle (returned by function `ODEHinge2Joint::GetAngle`) increased as the simulation ran. The controller was setting the target angular velocity based on the difference between the target steering angle and current steering angle, which was large. Multiplying this difference by the update rate of the controller further increased the resulting value. The controller was therefore attempting to set a target velocity that far exceeded the current angular velocity, but was unable to achieve the target steering angle in one controller time step.

The ODE Manual states: “The preferred method of setting body velocities during the simulation is to use joint motors. They can set body velocities to a desired value in one time step, provided that the force/torque limit is high enough.” ([44]). When the author increased the torque applied to the model's joints to 10,000, the joints were able to reach their target steering angle in one controller time step, eliminating the large difference between their current steering angle and target steering angle. However, although this resolved the problem by making it possible for the joints to reach their target steering angle in one controller time step, this resolution did not address the root cause of the problem, which was inaccurate maximum angular velocity determination.

When determining the maximum angular velocity at model CG in a turn of arbitrary radius, the author failed to include a factor of 2π in the denominator. As a result, the maximum angular velocity was approximately six times larger than it should have been. This problem did not become apparent until the second test, because the improved controller correctly limited the maximum steering angle of the model during the first test to that allowed by the vehicle's turning circle, and because the author allowed the steering wheels to turn to their maximum right extent before accelerating the model. Because the controller limits the angular velocity of each steering wheel around the steering axis to the calculated angular velocity or maximum angular velocity at model CG, whichever is greater, this had the effect of allowing the controller to set an angular velocity which exceeded the maximum angular velocity which would have prevented this problem.

In combination with increasing the torque applied to the steering axis so that the steering wheels were able to reach their target steering angles in one improved controller time step, correctly calculating the maximum angular velocity effectively resolved this problem. The author eliminated the use of function `ODEHinge2Joint::GetVelocity` entirely after noting that it occasionally returns "`nan`" (literally "not a number"), and proposes this may be the reason the function had no return value originally.

II.B. Swaybar implementation

While validating the improved controller, the author encountered a problem with rollover at moderate speeds, when the controller was correctly setting the velocity of each wheel so that rollover should have been prevented. Based on a review of the ODE

Manual, the author noted that similar problems were reported by other users, and that a proposed solution was to implement “anti-sway” bars to limit the back-and-forth rotation of a model around the x-axis.

The author revised the improved controller to calculate and apply a force to each joint during each controller update to compensate for some of this motion. However, the calculated displacement for each joint during each controller update was near zero, and the force applied to each joint insignificant. The author concluded the swaybar implementation was not a solution to the problem observed.

Through review of the model, the author determined that two of the representative challenge vehicle characteristics in use by the controller were in error.

Prior to deciding to develop an improved controller to achieve stability at high speeds, the author used the vehicle characteristics for a 2009 Honda Accord as the basis for a model using the packaged controller. The mesh for this model was also used as the “Car Obstacle”. See Figure 8.

The author was originally unable to determine the SSF of the Team 2005-06 challenge vehicle while researching the vehicle through commercial used car search services ([102] and [103]). As a result, the author estimated the Team 2005-06 challenge vehicle SSF using available information about the vehicle and general information about the class of vehicle.

When revising the 2009 Honda Accord model to simulate the representative challenge vehicle, the author calculated the height of vehicle CG using the alternate SSF and placed a geom having a mass of 1720 kg at a height of 0.894 m as the chassis body of the model. This was an error. The author later determined the Team 2005-06 challenge vehicle had an SSF of 1.17 ([104] and [105]).

To eliminate other potential errors, the author reviewed all representative challenge vehicle characteristics in use by the improved controller, and noted one additional error: when the author converted from English to Metric units, the author incorrectly calculated the rear track width of the model as 1.580 m (62.2 in), in lieu of 1.529 m (60.2 in), due to an error when entering data.

As a result of these errors, the effective SSF of the model was 0.88, but the improved controller was calculating the maximum velocity of the model using a SSF of 1.17, and accelerating the model to a velocity which predictably resulted in rollover. The author re-calculated the height of model CG using a SSF of 1.17 (0.653 m), and revised the model XML file to correct these errors. The model was then able to successfully complete the turn.

The author considers this incident, more than any other, highlights the ability of ODE as an accurate physics simulation to help identify problems with world files, models, or controller logic which are intended to model real-world interaction, and to help model real-world interaction which cannot be realistically evaluated, e.g., the risk of rollover.

III. PROBLEMS ENCOUNTERED DURING EVALUATION OF THE SIMULATION TARGETS

III.A. "`ODE Message 3`" error

The author encountered the following ODE error when running the simulations:

```
ODE Message 3: LCP internal error, s <= 0 (s=...)
```

with different values for `s`. The ODE Manual states this error "is usually caused by an object ramming into another with too much force (or just the right force)." ([44]), and recommends decreasing the mass of the object or changing the simulation timestep to eliminate the error, but also states: "this [error] won't crash your simulation".

The mass of the model was based on the mass of the representative challenge vehicle and the simulation timestep was selected through a process of trial and error to produce a stable simulation. As a result, the author decided not to change the mass of the model or the simulation timestep, and chose to ignore the error as a warning.

III.B. Angular unit inconsistencies between Player and Gazebo

The Player Project "World File Syntax" states: "Unless otherwise specified, the world file uses SI units (meters, seconds, kilograms, etc). One exception is angles and angular velocities, which are measured in degrees and degrees/sec, respectively." ([106]).

However, the `playerv` utility returns angular values in radians and angular velocities in radians/s. In addition, ODE functions in use by the improved controller such as `SetVelocity` for ODEHingeJoint and ODEHinge2Joint joints expect radians/s.

The ODE Manual states: "...ODE doesn't use specific units. You can use anything you wish, as long as you stay consistent with yourself." ([44]). Although ODE is unit agnostic, SI units are recommended.

For consistency with Gazebo world files, the improved controller reads Euler angles (included in `<rpy>` declarations) and the maximum steering angle (included in `<steerMaxAngle>` declarations) from an XML file which are measured in degrees, but internally uses angles and angular velocities measured in radians and radians/s.

III.C. `ODEHinge2Joint <anchorOffset>` problem

The author encountered a number of problems when evaluating the packaged steering controller. Specifically, the author was unable to interact with the simulation until the `AutoDisableFlag` and `SetParameters` problems were resolved. However, while attempting to resolve these problems, the author reviewed the ODE Manual and determined that the order of the chassis and wheel bodies defined by the `<joint>` declarations in each joint's `<body1>` and `<body2>` declarations was reversed in some joints but not others.

Specifically, "body1" of joints "left_front_wheel_hinge" and "right_front_wheel_hinge" was "chassis_body", but "body1" of joints "left_rear_wheel_hinge" and "right_rear_wheel_hinge" was the corresponding wheel. In the example included with the ODE Manual, "body1" is the chassis and "body2" is the wheel. The author attempted to resolve the problem by revising the order of the body declarations so that "body1" was "chassis_body" and "body2" the corresponding wheel for all joints. This was unsuccessful.

When the order was reversed so that "body1" was the chassis of the vehicle in lieu of the wheel for all joints, the author observed a "wobble" when the "simplecar" model was driven around in simulation. The wobble had the effect of causing the wheels of the model to leave the ground at high speed. Because the wobble of the wheels was not synchronized, the model was very unstable and would readily roll over. Without a stable model capable of traveling at speeds typical of vehicles participating in the 2004 and 2005 GCE in simulation, the author would not be able to test the rollover condition or effectively evaluate LIDAR.

Initially, the author believed the wobble was due to a mismatch between the `<anchorOffset>` declaration and the body to which it referred, and attempted to resolve this problem by revising the `<anchorOffset>` declaration for each wheel to be relative to the chassis of the vehicle, not the wheel. This was unsuccessful.

At this point, the author requested clarification from the playerstage-gazebo mailing list and received conflicting reports that the "simplecar" model and steering controller were and were not working. Specifically, that another user was able to load the model, but had not confirmed the model could be controlled using the `playerv` utility. Based on a response, the author downloaded Robot Operating System (ROS) ([107]) using the `svn` utility for the purposes of evaluating it for use.

The author was unable to locate the equivalent ODE source files in the ROS package "core code", and did not want to download and install software which might interfere with research to date. Rather than return to the beginning, install ROS, and resolve problems similar to those encountered when installing Player and Gazebo, the author elected not to continue with ROS.

While attempting to determine the cause of the wobble, the author reviewed the ODE Manual and noted that similar problems due to off-axis rotation at high-speed have been reported. ODE provides functions to limit the off-axis rotation of a body: `dBodySetFiniteRotationMode` and `dBodySetFiniteRotationAxis`. The author implemented these functions in files `Body.cc` and `ODEBody.cc` (and corresponding header files) to utilize the ODE-provided functions in an attempt to limit off-axis rotation: `SetFiniteRotationMode` and `SetFiniteRotationAxis`.

As a result of these changes, the author was able to read the values of two parameters from the model XML file: `finiteRotationMode` and `finiteRotationAxis` and set the finite rotation mode and axis of a body. However, setting the finite rotation mode and axis of the wheel bodies did not eliminate the wobble.

At this point, the author hypothesized that the problem was caused by miniscule errors in calculation over thousands of simulation cycles of the physics engine due to the weight of the chassis (1720 kg), as ODE attempted to maintain the position of the chassis body relative to each wheel. To eliminate the possibility that the length of the joints was contributing to the problem observed, the author revised the model to eliminate the z-dimension of the anchor offset by loading the model with the wheels at the same height as the chassis body. As a result of this change, the author was able to identify the cause of the wobble by experimentation.

The body of each wheel rotated in a plane at a fixed distance from its defined axis (y-axis for the rear wheels and z-axis for the front wheels). By eliminating the z-dimension anchor offset, the author changed the distance from the axis around which the body rotated, resulting in clear rotation around the y-axis. By increasing the distance from the axis, the effect was more pronounced.

As a result, the author reviewed the ODE Manual to determine what the intended effect of the anchor offset was, and discovered that ODE does not provide a function to set or return an anchor offset parameter. The anchor offset is used by file `Joint.cc` (and corresponding header file) for “setting anchor relative to gazebo body frame origin”. However, the anchor offset is applied to the body when the model is loaded. As a result, subsequent attempts to rotate the body around an axis rotate the offset body.

ODE attempts to keep the bodies of a joint together. Small values for anchor offset had less effect on the “simplecar” model than larger values, and extremely small values had no observable effect. When the values for anchor offset were too large, and one axis of rotation was eliminated, it became clear ODE was no longer able to compensate for the forces being placed on the axis by the controller, and was allowing the wheel bodies to freely rotate at a fixed distance from their axes, as defined by the `<anchorOffset>` declaration for each joint in the “simplecar” model.

In addition, the author determined the effect of changes to the x-, y-, and z-dimensions of the anchor offset in each joint by experimentation. The author concluded the orientation of the anchor offset was not preserved when the `<anchorOffset>` declaration was used. The wheels of the model were created from a cylinder, a built-in type of geom. Gazebo's built-in cylinder geom is defined by a radius and height. The height of the cylinder extends along the positive z-axis. To create a wheel, the cylinders modeling the left wheels were rotated around the x-axis by 90 degrees by the wheel body's `<rpy>` declaration when the model was loaded. Based on the behavior observed, if an anchor offset is declared for the wheel's corresponding joint this rotation also rotates the axis around which the wheels rotate by 90 degrees so that the anchor offset's positive y-dimension becomes the positive z-dimension, and positive z-dimension becomes the negative y-dimension. A similar rotation was observed for the cylinders modeling the right wheels. As a result, a front wheel would revolve around the z-axis by the anchor offset's positive y-dimension, and around the y-axis by the anchor offset's negative z-dimension. This made the problem more difficult to resolve.

The author considers this may be the cause of the failure of the front wheels of the "simplecar" model to turn when a type of "full" was declared and the packaged steering controller was in use. The revolution of the wheel bodies around the y- and z-axes may effectively "bind" the wheels, preventing rotation. The author eliminated the `<anchorOffset>` declaration from each joint, and the use of anchor offset entirely. This greatly increased stability at high speed by eliminating the wobble, and made it possible for the author to implement four-wheel drive at speeds typical of vehicles participating in the 2004 or 2005 GCE.

The author considers this and other similar issues, such as the reversed order of parameters described above, to be evidence of a "meandering direction of development" evident through review of the Gazebo codebase. The Gazebo codebase is being actively developed. Some features have been abandoned, others were never fully implemented, and the purpose of some functions is unclear from function declarations.

For example, ODE has no intrinsic function to set a second hinge anchor in either a `Hinge` or a `Hinge2` joint. However, Gazebo functions `ODEHingeJoint::SetAnchor` and `ODEHinge2Joint::SetAnchor` both include a parameter `index`, which is commented out. Parameter `index` is commented out in several other `ODEHingeJoint` functions.

In addition, the ODE Manual states: "These parameter names can be optionally followed by a digit (2 or 3) to indicate the second or third set of parameters, e.g. for the second axis in a hinge-2 joint, or the third axis in an AMotor joint." ([44]). However, the list of parameters to which the ODE Manual refers does not include parameter `axis`. Functions for setting or getting the axis of a `Hinge2` joint are documented by the ODE Manual, but not parameter `axis`. As a result, the author concluded ODE may have, at one time, used parameter names followed by a digit to indicate a second or third set of parameters, and that Gazebo functions `ODEHingeJoint::SetAnchor` and `ODEHinge2Joint::SetAnchor` may be referring to these numbers as the "index", and that "index" has since been commented out because Gazebo has not been updated to remove these references.

Diagnosing the `ODEHinge2Joint anchorOffset` problem required several days, during which more productive research was delayed.

III.D. `OGRE::AxisAlignedBox` error

The author encountered the following error while attempting to resolve the `ODEHinge2Joint <anchorOffset>` problem reported above (only a portion of the actual error message is included herein):

```
Assertion `(min.x <= max.x && min.y <= max.y && min.z
<= max.z) && "The minimum corner of the box must be
less than or equal to maximum corner"' failed.
```

In general, this error occurred immediately before a segmentation fault which terminated the running simulation. The author did not encounter this error after resolving the `ODEHinge2Joint <anchorOffset>` problem.

REFERENCES

1 DARPA, *DARPA Grand Challenge 2004 Rules*, version April 1.2, dated April 2, 2003

2 DARPA, *DARPA Grand Challenge 2005 Rules*, dated October 8, 2004

3 DARPA, *Grand Challenge 2004 Final Report*, July 30, 2004

4 House Report 106-945, *Enactment of Provisions of H. R. 5408, The Floyd D. Spence National Defense Authorization Act for Fiscal Year 2001*, Library of Congress, dated October 6, 2000

5 News Release, *A Huge Leap Forward for Robotics R&D*, dated October 9, 2005

6 DARPA, *DARPA Grand Challenge 2004 Rules*, dated January 5, 2004

7 DARPA, *Report to Congress: DARPA Prize Authority*, dated March, 2006

8 Press Release, *Organizers of Autonomous Robotic Ground Vehicle Challenge Announce Initial Team Selection*, dated November 13, 2003

9 Press Release, *Final Data from DARPA Grand Challenge*, dated March 13, 2004

10 DARPA, *QID Process Description*, dated January 2, 2004

11 DARPA, Archived Grand Challenge 2004 Website, http://www.darpa.mil/grandchallenge04/index.html

12 Carl D. Crane III, David G. Armstrong II, Robert Touchton, et al., *Team CIMAR's NaviGATOR: An Unmanned Ground Vehicle for the 2005 DARPA Grand Challenge*, Journal of Field Robotics, Vol. 23, No. 8, Wiley Periodicals, Inc., 2006

13 Desert Buckeyes, *ION: The Intelligent Off-Road Navigator The Desert Buckeyes'*

entry in the DARPA Grand Challenge 2005, no date (2005)

14 Qi Chen and Ümit Özgüner, *Intelligent Off-Road Navigation Algorithms and Strategies of Team Desert Buckeyes in the DARPA Grand Challenge 2005*, Journal of Field Robotics, Vol. 23, No. 9, Wiley Periodicals, Inc., 2006

15 Richard Mason, Jim Radford, Deepak Kumar, et al., *The Golem Group/University of California at Los Angeles Autonomous Ground Vehicle in the DARPA Grand Challenge*, Journal of Field Robotics, Vol. 23, No. 8, Wiley Periodicals, Inc., 2006

16 MITRE Meteorites, *2005 DARPA Grand Challenge Entry*, no date (2005)

17 Robert Grabowski, Richard Weatherly, Robert Bolling, et al., *MITRE Meteor: An Off-Road Autonomous Vehicle for DARPA's Grand Challenge*, Journal of Field Robotics, Vol. 23, No. 9, Wiley Periodicals, Inc., 2006

18 MonsterMoto, *Technical Paper*, Revision A, dated August 29, 2005

19 Red Team, *DARPA Grand Challenge 2005 Technical Paper*, dated August 24, 2005

20 Red Team Too, *DARPA Grand Challenge 2005 Technical Paper*, dated August 24, 2005

21 Chris Urmson, Charlie Ragusa, David Ray, et al., *A Robust Approach to the High-Speed Navigation for Unrehearsed Desert Terrain*, Journal of Field Robotics, Vol. 23, No. 8, Wiley Periodicals, Inc., 2006

22 SciAutonics/Auburn Engineering, *The Autonomous Ground Vehicle RASCAL: Team SciAutonics/Auburn Engineering in the DARPA Grand Challenge 2005*, no date

23 William Travis, Robert Daily, David M. Bevly, et al., *SciAutonics-Auburn Engineering's Low-Cost High-Speed ATV for the 2005 DARPA Grand Challenge*, Journal of Field Robotics, Vol. 23, No. 8, Wiley Periodicals, Inc., 2006

24 Team Cajunbot, *Technical Overview of CajunBot (2005)*, no date (2005)

25 Arun Lakhotia, Suresh Golconda, and Anthony Maida, et al., *CajunBot: Architecture and Algorithms*, Journal of Field Robotics, Vol. 23, No. 8, Wiley Periodicals, Inc., 2006

26 Team Caltech, *DARPA Technical Paper: Team Caltech*, dated August 29, 2005

27 Team Cornell, *Technical Review of Team Cornell's Spider*, no date (2005)

28 Isaac Miller, Sergei Lupashin, Noah Zych, et al., *Cornell University's 2005 DARPA Grand Challenge Entry*, Journal of Field Robotics, Vol. 23, No. 8, Wiley Periodicals, Inc., 2006

29 Team ENSCO, *Team ENSCO's DEXTER*, no date (2005)

30 Team TerraMax, *DARPA Grand Challenge 2005*, no date (2005)

31 Deborah Braid, Alberto Broggi, and Gary Schmiedel, *The TerraMax Autonomous Vehicle*, Journal of Field Robotics, Vol. 23, No. 9, Wiley Periodicals, Inc., 2006

32 Red Team, *DARPA Grand Challenge Technical Paper*, Revision 6.1, dated April 8, 2004

33 Rob Meyer Productions, *Technical Paper for DARPA Grand Challenge*, no date (2004)

34 Oshkosh Truck Co. and The Ohio State University (Team TerraMax), *Technical Paper for TerraMax*, no date (2004)

35 Virginia Tech Grand Challenge Team, *DARPA Grand Challenge 2005*, no date (2005)

36 Brett M. Leedy, Joseph S. Putney, Cheryl Bauman, et al., *Virginia Tech's Twin Contenders: A Comparative Study of Reactive and Deliberative Navigation*, Journal of Field Robotics, Vol. 23, No. 9, Wiley Periodicals, Inc., 2006

37 Virginia Tech Team Rocky, *DARPA Grand Challenge 2005*, no date (2005)

38 The Player Project, http://playerstage.sourceforge.net (last accessed January 12, 2010)

39 CMake, *CMake FAQ*, http://www.cmake.org/Wiki/CMake_FAQ, dated January 28, 2010 (last accessed February 26, 2010)

40 The Player Project, *Model Creation Tutorial*, http://playerstage.sourceforge.net/doc/Gazebo-manual-svn-html/tutorial_model.html, dated August 4, 2007 (last accessed February 26, 2010)

41 The Player Project, *Mesh Creation Tutorial*, http://playerstage.sourceforge.net/doc/Gazebo-manual-svn-html/tutorial_mesh.html, dated August 4, 2007 (last accessed February 26, 2010)

42 The Blender Foundation, http://www.blender.org/ (last accessed February 26, 2010)

43 Torus Knot Software, Ltd., *Blender Exporter* http://www.ogre3d.org/wiki/index.php/OGRE_Meshes_Exporter (last accessed February 26, 2010)

44 Open Dynamics Engine, *ODE Manual*,

http://opende.sourceforge.net/wiki/index.php/Manual, dated July 22, 2008 (last accessed February 26, 2010)

45 SICK AG, *Technical Description - LMS200/211/221/291 Laser Measurement Systems*, dated December, 2006

46 SICK AG, *LMS 200/LMS 211/LMS 220/LMS 221/LMS 291 Laser Measurement Systems*, June, 2003

47 Paul G. Trepagnier, Jorge Nagel, Powell M. Kinney, et al., *KAT-5: Robust Systems for Autonomous Vehicle Navigation in Challenging and Unknown Terrain*, Journal of Field Robotics, Vol. 23, No. 8, Wiley Periodicals, Inc., 2006

48 The Golem Group / UCLA, *DARPA Grand Challenge Technical Paper*, no date (2005)

49 Stanford Racing Team, *Stanford Racing Team's Entry In The 2005 DARPA Grand Challenge*, no date (2005)

50 DARPA, *DARPA Grand Challenge 2005 Route Data Definition File*, August 3, 2005

51 Sebastian Thrun, Mike Montemerio, Hendrick Dahlkamp, et al., *Stanley: The Robot that Won the DARPA Grand Challenge*, Journal of Field Robotics, Vol. 23, No. 9, Wiley Periodicals, Inc., 2006

52 Team Overbot, *Team Overbot*, Revision 3, dated September 22, 2003 (the pages of this reference are dated February 13, 2004)

53 The Gray Team, *Team Gray Technical Paper*, date August 28, 2005

54 The Golem Group, *The Golem Group*, no date (2004)

55 Palos Verdes High School Warriors, *DARPA Grand Challenge Technical Paper*, March 1, 2004

56 Intelligent Vehicle Safety Technologies 1, *Technical Description*, dated August 29, 2005

57 Mojavaton, *Technical Paper*, dated August 28, 2005

58 A. I. Motorvators, *AI Motorvators Technical paper*, dated March 4, 2004

59 Axion Racing, *Technical Paper*, dated February 29, 2004

60 Princeton University, *Technical Paper*, no date (2005)

61 Team Caltech, *Technical Paper*, dated February 23, 2004

62 Virginia Department of Education in cooperation with the Virginia Department of Motor Vehicles, *Curriculum and Administrative Guide for Driver Education in Virginia*, 2001

63 Virginia Department of Education in cooperation with the Virginia Department of Motor Vehicles, *Curriculum Scope and Sequence Modules for Driver Education in Virginia, Module Eleven: Behind-the-Wheel and In-Car Observation*, dated August, 2001

64 SciAutonics I, *Technical Paper for DARPA Grand Challenge*, no date

65 AVID-ET and SciAutonics, *Technical Paper Addendum for DARPA Grand Challenge*, no date (2004)

66 Axion Racing, *DARPA Grand Challenge 2005 Technical Paper*, dated August 11, 2005

67 Team Arctic Tortoise, *Arctic Tortoise Technical Paper*, Revision 1, no date (2004)

68 Chris Urmson, Joshua Anhalt, Michael Clark, et al., *High-Speed Navigation of Unrehearsed Terrain: Red Team Technology for Grand Challenge 2004*, CMU-RI-TR-04-37, The Robotics Institute, Carnegie-Mellon University, June, 2004.

69 DARPA, *DARPA Grand Challenge Team Newsletter #1*, dated August 27, 2003

70 Department of Defense, *DARPA Grand Challenge: Defense Advanced Research Projects Agency Competition for Autonomous Robotic Ground Vehicles; 2004, 2005, and 2007 Events, Urban Challenge, Reports, Movies (Two CD-ROM Set)*, Progressive Management, dated May 2, 2007

71 Martin Buehler, Karl Iagnemma, and Sanjiv Singh, *The 2005 DARPA Grand Challenge: The Great Robot Race*, Springer-Verlag Berlin Heidelberg, 1st ed., dated October 23, 2007

72 Amazon.com, Inc., http://www.amazon.com (last accessed August 10, 2009)

73 Anand R. Atreya, Bryan C. Cattle, Brendan M. Collins, et al., *Prospect Eleven: Princeton University's Entry in the 2005 DARPA Grand Challenge*, Journal of Field Robotics, Vol. 23, No. 9, Wiley Periodicals, Inc., 2006

74 Alain Kornhauser, *About Prospect Eleven and Princeton University's "DARPA Project"*, Princeton University, dated November 29, 2005

75 Press Release, *DARPA Grand Challenge Finalizes Field for Qualification, Inspection and Demonstration Event*, dated December 19, 2003

76 Press Release, *Finalists Selected for DARPA Grand Challenge*, dated October 5, 2005

77 DARPA, Archived Grand Challenge 2005 Website,

http://www.darpa.mil/grandchallenge05/index.html

78 irobotics.org, http://www.irobotics.org (last accessed January 5, 2010)

79 The Player Project, *Installation (Linux)*, http://playerstage.sourceforge.net/doc/Gazebo-manual-cvs-html/install.html, dated September 12, 2005

80 The Player Project, *Installation (Linux)*, http://playerstage.sourceforge.net/doc/Gazebo-manual-svn-html/install.html, dated August 4, 2007

81 Gazebo, *README*, version 0.9.0

82 irobotics.org, *Player 2.1.1 / Gazebo 0.8 SVN rev. 6886 Installation on Fedora 9*, http://www.irobotics.org/gazebo08.f8.html, dated August 19, 2008

83 The Player Project, *Installation Instructions for Gazebo*, http://playerstage.sourceforge.net/wiki/Install, dated March 3, 2009

84 The Player Project, *Installation Instructions for Gazebo*, http://playerstage.sourceforge.net/wiki/Install, dated January 12, 2010 (last accessed January 12, 2010)

85 The Player Project, *Prerequisites*, http://playerstage.sourceforge.net/doc/Gazebo-manual-svn-html/prerequisites.html, dated August 4, 2007

86 NVIDIA Corporation, *Unix Drivers Portal Page*, http://www.nvidia.com/object/unix.html (last accessed December 18, 2009)

87 Novell, Inc., *NVIDIA Installer HOWTO for SUSE LINUX users*, http://www.suse.de/~sndirsch/nvidia-installer-HOWTO.html (last accessed

December 18, 2009)

88 FreeImage, *README.linux*, version 3.13.0

89 Object-oriented Input System, *ReadMe.txt*, version 1.2.0

90 Open Dynamics Engine, *INSTALL.txt*, version 0.11.1

91 Fast Light ToolKit, *README*, version 1.1.9

92 CrazyEddie's GUI System, *README*, version 0.6.2b

93 Crazy Eddie's GUI System Project, http://www.cegui.org.uk (last accessed September 11, 2009)

94 Free Software Foundation, Inc., *Coreutils - GNU core utilities*, http://www.gnu.org/software/coreutils/, dated February 21, 2009 (last accessed December 18, 2009)

95 Torus Knot Software, Ltd., http://www.ogre3d.org (last accessed September 10, 2009)

96 NVIDIA Corporation, http://developer.nvidia.com (last accessed September 9, 2009)

97 FFmpeg, *INSTALL*, version 0.5

98 The Player Project, *Standard Install Procedure*, http://playerstage.sourceforge.net/doc/Player-cvs/player/install.html, dated September 12, 2005

99 Player, *README*, version 3.0.0

100 Player, *INSTALL*, version 3.0.0

101 John Hsu, *Re: [PlayerStage-Gazebo] Problem with simplecar.world*, playerstage-

gazebo mailing list, dated January 19, 2010

102 Edmunds, http://www.edmunds.com/

103 Cars.com, http://www.cars.com/

104 Motor Trend, http://www.motortrend.com/

105 Autobuyguide.com, http://www.autobuyguide.com (last accessed March 25, 2010)

106 The Player Project, *World File Syntax*, http://playerstage.sourceforge.net/doc/Gazebo-manual-svn-html/config_syntax.html, dated August 4, 2007 (last accessed March 15, 2010)

107 ROS.org, http://www.ros.org/wiki/ (last accessed March 15, 2010)

1. The text of the footnote reported by DARPA differs slightly from the text of the Fiscal Year 2001 National Defense Authorization Act, which states: “It shall be a goal of the Armed Forces to achieve the fielding of unmanned, remotely controlled technology such that... by 2015, one-third of the operational ground combat vehicles are unmanned.” ([4], p. 46).
2. Several teams participating in the 2004 and 2005 GCE made extensive use of pre-planning or pre-mapping prior to the race to effectively eliminate from consideration for the controlling intelligence all terrain but the actual course defined by the RDDF. The task of the controlling intelligence was therefore made simpler, and became one of distinguishing the course from terrain which had been eliminated from consideration by the team, and avoiding unintended obstacles.
3. Ostensibly, the criteria used to determine which vehicles were of interest to the DOD were reported by DARPA. However, only 25 of the 86 technical proposals received by DARPA are available for review, and it is unclear why 41 teams submitted technical proposals describing vehicles which did not satisfy these criteria. The author proposes the reported criteria were not the only criteria used by DARPA to determine which vehicles were of interest to the DOD, but concluded the published record does not provide enough information to be able to determine what deficiencies or weaknesses caused 41 of 86 teams to be eliminated.
4. DARPA stated: “...DARPA selected 19 teams for advancement to the next phase of the Grand Challenge and established a site visit process to determine the final 6

teams." ([3], p. 4). However, DARPA selected 18 teams to participate in the 2004 QID, in lieu of the 19 claimed. The technical proposal submitted by the ION Team was one of 19 technical proposals described by DARPA as "completely acceptable" on November 13, 2003, approximately four months prior to the 2004 GCE ([8]), but the ION Team was not selected to participate in the 2004 QID.

5. DARPA stated the purpose of the technical inspection was to ensure each challenge vehicle "complied with all rules and was safe to operate". Published records indicate the technical inspection did not identify challenge vehicles which were not safe to operate. For example, DARPA stated ([9]):

```
[Team 2004-02] - Vehicle circled the wrong way in the
start area.  Vehicle was removed from the course.
[Team 2004-09] - Vehicle hit a wall in the start area.
Vehicle was removed from the course.
[Team 2004-16] - Vehicle brushed a wall on its way out
of the chute.  Vehicle has been removed from the
course.
```

Although some time elapsed between the technical inspection and the 2004 GCE, it is unreasonable to conclude changes made by Teams 2004-02, 2004-09, and 2004-16 were responsible for their disqualification.

As a result, the author concluded the purpose of the technical inspection was not to ensure each challenge vehicle "was safe to operate", but that it "complied with all rules" DARPA established concerning devices emitting radiation, warning devices, e-stop requirements, etc.

6. DARPA identified obstacles selected as “representative” ([10]), however published records support a conclusion that obstacles selected as representative were not comprehensive. Several teams selected to participate in the 2004 GCE were eliminated by obstacles not identified as representative, and which teams apparently did not encounter during the 2004 QID, such as wire, fence, brush, or obstacles too small to detect. For example, DARPA stated ([9]):

   ```
   [Team 2004-04] - At mile 0.45, vehicle ran into some
   wire and got totally wrapped up in it.
   [Team 2004-06] - At mile 6.0, vehicle was paused to
   allow a wrecker to get through, and, upon resuming
   motion, vehicle was hung up on a football-sized rock.
   [Team 2004-17] - At mile 1.3, vehicle veered off
   course, went through a fence, tried to come back on
   the road, but couldn’t get through the fence again.
   [Team 2004-23] - Several times, the vehicle sensed
   some bushes near the road, backed up and corrected
   itself.  At mile 1.2, it was not able to proceed
   further.
   ```

7. DARPA did not publish the point deductions in use during the 2004 QID. It is unclear if, for example, more points were deducted for exceeding the speed limit by 20 mph versus five mph, or if collisions with obstacles were “weighed” by assigning a severity to the collision.

8. DARPA did not request teams participating in the 2004 GCE respond to a

question similar to 2005 GCE SQ 2.5.1.

9. Teams 2004-10 and 2004-11 referred to the use of "simulation" but not a simulation environment similar to the Player Project:

- Team 2004-10 stated: "Post-processing synthesized goodness-maps and facilitated run-simulations in addition to physical testing." ([32], p. 6).
- Team 2004-11 stated: "Based on the earlier simulation we had written, we are still busy at this date (24 Feb) arriving at an optimum arrangement and timing for the peripherals to 'talk' to each other." ([33], p. 8).

10. Teams 2005-13 and 2005-14 originally proposed using seven LIDAR sensors during the 2005 GCE.

11. Other body and geom primitives supported by Gazebo include: `box`, `cylinder`, `sphere`, `trimesh`, `cone`, `heightmap`, and `plane`.

12. This was not possible with the representative challenge vehicle. By default, Gazebo places the CG of a body at its center. Use of a trimesh geom primitive for the representative challenge vehicle model would have placed the model CG higher than the representative challenge vehicle CG, resulting in simulated vehicle dynamics which inaccurately model real world vehicle dynamics.

13. The field-of-view of the Navtech DS2000 RADAR is 360 degrees. However, Teams 2005-13 and 2005-14 stated: "Most of the RADAR's scan is obscured by the vehicle or brush guard. Its effective field of view is 70 degrees 40-70 meters in front of the vehicle." ([19], p. 8 and [20], p. 8).

14. Sensors with a maximum effective range greater than 20 m (corresponding to a

stopping distance of 19.3 m at a maximum velocity of 25 mph) were not required for a team challenge vehicle to complete the 2004 or 2005 GCE course in less than ten hours. However, the effective use of complementary sensors to extend obstacle detection range and allow driving at higher speeds provided a competitive advantage to teams with significant experience and several potentially disruptive teams. Team 2005-16 was the most successful team to use complementary sensors.

15. DARPA did not identify which team proposed this. However, Teams 2004-11 and 2004-20 reported technologies which were similar in concept:

- Team 2004-11 stated: "In addition, we have kept the odometer on the axle with no brakes. This allows us to sense a skid or inadequate braking impulse. We have no algorithm for the former." ([33], p. 2).
- Team 2004-20 stated: "Vehicle speed as measured by the radar speedometer is compared with vehicle speed as measured at the driveshaft to detect slippage." ([52], p. 6).

16. Ironically, DARPA did not request teams participating in the 2004 QID or GCE or 2005 GCE describe how they would handle the loss of any other sensors.

17. Although not directly related to navigation sensors, several teams reported the potential military deployment of autonomous ground vehicles was a consideration in their selection of obstacle and path detection sensors. For example, Team 2005-12 stated: "Passive sensing offers advantages in both a military context, where undetectable sensors are crucial for effective operation and in a civilian

context, in which multiple autonomous vehicles must not interfere with one another." ([60], p. 2).

18. Teams 2005-16 and 2005-21 described calibration of sensors which was similar in concept. However, calibration of sensors was performed by the team. The challenge vehicle controlling intelligence was not taught to learn to interpret sensor data.

- Team 2005-16 stated: "...the sensors are periodically calibrated using data of dedicated obstacles of known dimensions. Calibration is an offline process which involves human labeling of sensor data. The calibration process adjusts the exact pointing directions of the individual sensors by minimizing a quadratic error, defined through multiple sightings of the same calibration obstacle." ([49], p. 8).
- Team 2005-21 stated: "Thanks to a precise calibration of the cameras – performed on a graduated grid – the three degrees of freedom specifying cameras orientation are fixed to known values, and in particular – in order to ease the subsequent processing – the yaw and roll angles are fixed to zero for all cameras." ([30], p. 10).

19. Team 2005-12 later stated: "...[The challenge vehicle] suffered a communications failure between the GPS unit and the guidance computer just before Beer Bottle Pass, a mountain pass near the end of the course, that would have ended a fully autonomous attempt." ([73], p. 753). Obviously, a similar communications failure during the 2005 GCE would have resulted in a similar outcome. However, the author is here attempting to distinguish between potentially-disruptive teams

which proposed competent system integration at a reasonable procurement cost and other teams, and there is no evidence supporting a conclusion a similar communications failure would have occurred during the 2005 GCE if Team 2005-12 had not failed to complete the 2005 GCE due to the programming error reported.

20. Teams are listed in alphabetical order, based on the alphabetizing scheme utilized by DARPA, in which the words "a", "an", or "the" are not considered significant ([75]). DARPA established an alternate alphabetizing scheme which treats the word "team" as "a", "an" or "the" ([76]). The original alphabetizing scheme was retained so that the teams would appear in the same order when referenced herein.

    In addition, to preserve alphabetical order, the occurrence of the team names "MonsterMoto" and "Mojavaton", in the order presented by [76], was reversed.