DARPA Urban Challenge Technical Paper[‡]: Sydney-Berkeley Driving Team

Ben Upcroft[†], Michael Moser, Alexei Makarenko, David Johnson
University of Sydney

Ashod Donikian, Alen Alempijevic
University of Technology, Sydney

Robert Fitch, William Uther
National ICT Australia

Jan O. Biermeyer, Humberto Gonzalez, Esten Grotli,
Todd Templeton, Vason P. Srini, Jonathan Sprinkle
University of California, Berkeley

Team Members & Advisors—Eric Andres, Alexandre Bayen, Alex Brooks, Christopher Brooks, Iain Brown, Eric Chang, Wuyang Chen, Charlie Chiau, Jaechoon Chon, Hoam Chung, Gamini Dissayanake, Miguel Domingo, Bertrand Douillard, Vincent Duindam, Hugh Durrant-Whyte, J. Mikael Eklund, Ken Elkabany, Danish Farooq, Filip Furmanek, Mark Godwin, Michael Greenbaum, Jose Guivant, Karl Hedrick, Nicholas Hosein, Prasanth Jeevan, Roman Katz, Cheryl Killingsworth, Youngheub Kim, Sarath Kodagoda, Peter Lau, Kwong Leung, James Lin, Minh Van Ly, Rick Mann, Osman Massod, Janarbek Matai, John McCarthy, Kovid Mishra, Mansi Modi, Michael Ng, Brian Nguyen, Anindha Parthy, Terry Percival, Jacob Porter, Olek Proskurowski, Travis Pynn, Scott Robertson, Pannag Sanketi, S. Shankar Sastry, Olga Sawtell, David H.C. Shim, David Skellem, Stephan Sehestedt, Steven Shladover, Derek Speer, Aaron Staley, Jeff Tang, Masayoshi Tomizuka, Igor Tregub, Joel Veness, Ryan Waliany, Andrew Wan, Jacek Wnuk, Bailey Wu, Chaoying Wu, Boyang Zhang, Jia Zou.

[‡] DISCLAIMER: The information contained in this paper does not represent the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA) or the Department of Defense. DARPA does not guarantee the accuracy or reliability of the information in this paper.

This work was supported in part by Rio Tinto, Komatsu, Toyota, Chess at Berkeley, ZeroC, Advantech, PMD, Xilinx. Additional in-kind support was provided in the form of discounts on equipment from the following manufacturers: SICK, NovAtel, Honeywell.

[†] All correspondence should be directed to b.upcroft@cas.edu.au

***Executive Summary*—The Sydney-Berkeley Driving Team is a collaboration between academic and research personnel from (in alphabetical order) the National Information and Communication Technology of Australia, University of California, Berkeley, University of Sydney, and the University of Technology, Sydney. This document describes the planning, actuation, simulation, communication, theoretical tasks, advancements, and projections necessary for the team to compete in the DARPA Urban Challenge. Among our major accomplishments, we claim the ability for distributed code development through the use of our component-based middleware, a high-confidence testbed which was designed and implemented from the ground up by our engineers, prototype testing in months, and robust software design and development allowing a seamless transition between simulation and online testing.**

I. Introduction

THIS technical report describes the Sydney-Berkeley Driving Team's entry, 'Number 5', in the 2007 DARPA Urban Challenge. The DARPA Urban Challenge is the logical continuation of the much acclaimed and highly followed DARPA Grand Challenge I and II in 2004, and 2005. The focus of this competition is more on the humanization (not necessarily the remote operation) of autonomous vehicles than either of the previous DARPA Grand Challenges, and thus special efforts are required to achieve high-confidence behavior near humans, in the vicinity of other vehicles, and in concert with road traffic.

In this document we describe the various tasks which we have identified as paramount for completing the race, and our approach to each one of those tasks. Additional information regarding the formation and management of the team (and its distributed nature) is also provided.

A. *Sydney-Berkeley Driving Team*

The team was formed as a collaboration between the University of Sydney and the University of California, Berkeley, shortly after the announcement of an Urban Challenge competition, informally during May of 2006 and formally in June of 2006 through submission of a Track A proposal to the Urban Challenge. Our continued cooperation was anticipated during the submission of the Track A proposal, and the leaders from each institution committed to participation in the competition regardless of the outcome of the Track A submission results.

As work on the project has continued, additional institutions joined the team, namely the National ICT Australia (NICTA) and the University of Technology, Sydney.

The members of the team are undergraduate researchers, bona fide engineers, M.S. and Ph.D. students, postdoctoral researchers, and faculty and staff members. The cumulative team size since the beginning of the project is over 50 members, and the steady-state team size is around 15 persons. A majority of the team members are volunteers and are participating due to the charisma of the team and that of the competition.

B. *Technical Project Management*

The project work proceeded in three major phases which correspond to the three initial phases of the DARPA Urban Challenge: the initial design and strategy, the testbed development with DARPA video submission, and site visit participation. During these three phases the first strategic phase took place equally between the locations in Sydney and Berkeley, the second phase, where the testbed was developed and proved, took place mainly in Sydney with some participation from visiting Berkeley personnel, and the third phase mostly in Berkeley, with significant participation from Sydney personnel. In addition to technical teams, administrative teams were necessary to maintain safety during testing, conduct necessary public relations work, and other administrative tasks.

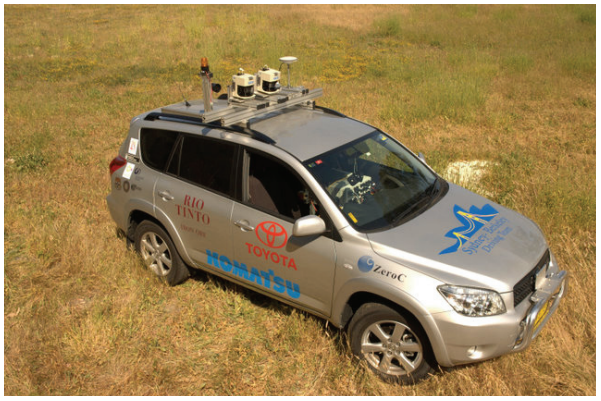


Figure 1. Our vehicle, Number 5

C. *The Vehicle*

Our vehicle was actuated from a stock Toyota RAV4 (2006) donated to the team in Sydney, Australia. The vehicle team was responsible for providing steer, throttle, and brake control inputs which could be computer controlled, and also an electronics interface for the various sensors and wireless emergency stop system. Modifications to the vehicle chassis were not required, but modifications to the external frame of the vehicle (including a nudge bar, roof racks, etc.) were performed.

II. System Architecture

The Urban Grand Challenge not only poses the problem of autonomy at the hardware level but also at abstract levels such as perception and decision making usually only performed by human operators. The SBDT has chosen a classic N-tier architecture [17] to address the large range of abstraction required in this competition Figure 2. The tiers, separated by the level of abstraction in control and perception, include modeling, planning, and execution elements.

The N-tier architecture encodes the following separation of concerns. The task of driving from point A to point B is first considered on the global level. Given the town map a trip plan is generated in terms of GPS-referenced waypoints. Travel between waypoints can only be performed by following streets (ignoring car parks for the moment).

Perception, modeling, and planning motion along a street segment in the presence of traffic and complying with traffic rules is handled by the symbolic level of the architecture.

Local environment modeling and action planning guarantees physical safety of the vehicle as it moves towards its goal. Information about the local environment and the vehicle itself is extracted from on-board sensors. The extent of modeling and planning at this level is limited by the sensor range and a finite (short) history of past observations.

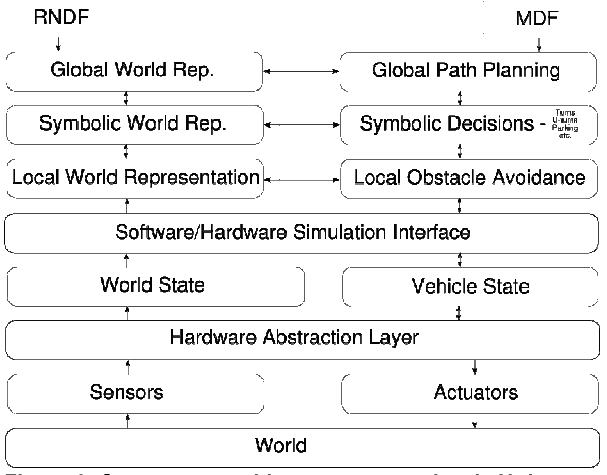


Figure 2. Our system architecture uses a classic N-tier approach.

Some of the representative sub-tasks at each of these levels include tracking the vehicle location relative to the map; identifying street boundaries, buildings and other cars; controlling vehicle motion; reasoning about traffic rules, and many others.

A. Computers and Networking

The on-board computer system currently has four hosts (not counting diagnostic laptops that are often connected to the system). The on-board computers use two operating systems: Linux (Kubuntu) and QNX. The computing hardware consists of off-the-shelf rack-mounted PCs with Intel dual-core processors. The hosts are connected with a standard 1-Gigabit Ethernet hub. Latency performance tests with this setup are shown in Figure 3.

Some but not all parts of our system require real-time features. For example, there is a strong need for accurate time-stamping of sensor data. A vehicle in the competition moves at speeds of up to 30mph (48km/h) and small sporadic delays in the standard Linux kernel can have significant negative impact, particularly in navigation. To address this issue, we use a dedicated host running the real-time QNX Neutrino operating system for all interactions with sensor and actuator hardware.

B. Software Architecture

Building reliable robotic hardware is a difficult task in itself, but the major challenge of this competition is in software. The main difficulties arise from the task's complexity and its dynamic real-time nature. Other contributing factors include the distributed and cross-platform computing environment, the large number of software contributors, and the need to use existing code.

Our software is built using a Component-Based Software Engineering (CBSE) approach. This offers modularity, software reuse, and flexibility in deployment, all of which are necessary to address the problems listed above. Applied to a robotic application, CBSE means that algorithms for perception and navigation are mapped to a set of components. Components run asynchronously and exchange information through communication.

We use ZeroC's Ice middleware [18] extensively in our system: for component interface definition, inter-component communication, component deployment, location, activation services, etc. We also use Orca—an open source project that customizes Ice to robotic applications and provides an on-line repository of reusable components [19].

The total number of components that comprises our current system is less than 20. However, this is expected to increase as the complexity and number of algorithms increase for future qualification events. (For reference, the winner of the previous competition had approximately thirty.) The software is written

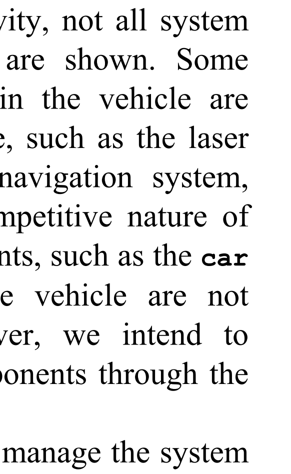
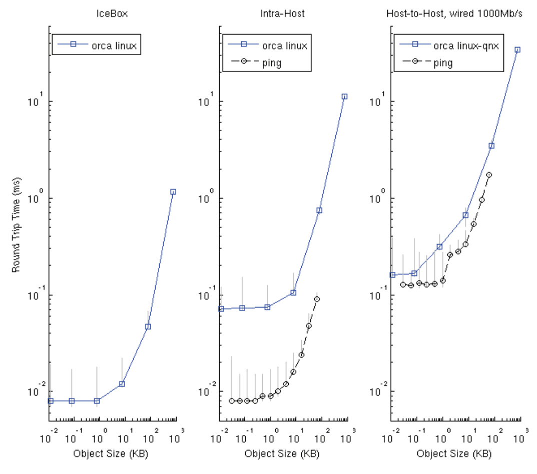



Figure 3.: The tests we perform are reflect the typical usage of our components: frequent transfer of small to medium size messages. Here we provide some numbers on round trip time (RTT) for sending messages of various sizes. The plots above show average RTT for sending 100 objects with an interval of 0.25s. Vertical bars show min and max values. Ping results are shown for comparison.

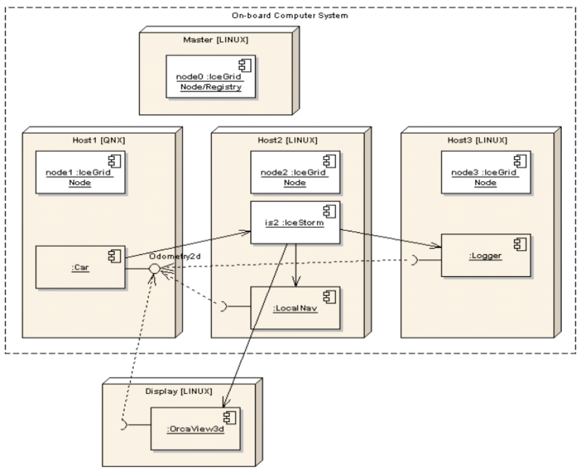


Figure 4.: Deployment diagram illustrating how the on-board computer system is configured.

by about a dozen people from four organizations (this number is higher if we count the authors of the existing components used directly in our system).

Figure 4 illustrates the current deployment strategy. The master host executes an instance of an IceGrid node with a collocated IceGrid registry. Every other host runs an IceGrid node. One host runs QNX Neutrino, the remaining hosts run Kubuntu Linux. The QNX host executes low-level actuator and sensor components (our own partial, unsupported port of Ice to QNX is available through the ZeroC developer forums. For brevity, not all system components (e.g., lasers) are shown. Some components that we use in the vehicle are currently publicly available, such as the laser range-finder, the inertial navigation system, and others; due to the competitive nature of this project, other components, such as the `car` component that drives the vehicle are not publicly available. However, we intend to release most of these components through the Orca project in the future.

Using Ice's tools we can manage the system from a centralized XML file, and distribute the system's components to computational nodes.

III. Control System Design

The lowest level of the control system (see Figure 2) consists of the vehicle actuation. This system was custom designed based on commercial systems while taking advantage of the existing drive-by-wire capabilities of the RAV4. Four components are critical to the control and operation of the vehicle; throttle, brake, steering, and transmission. A fifth component, the ignition, can be regarded as important for an autonomous vehicle start and for safety if an emergency cut-off to the engine is required.

For the modifications to each of these components, reliability, actuation speed, and control accuracy were considered. In addition it was clear from the experiences in the previous DARPA Grand Challenge that rigorous testing was vital for success of the project. Finally, in case of electrical, mechanical, or software failure, an emergency stop over-ride automatically shuts down all autonomous actuation and brings the vehicle to a quick stop without the need for external commands.

For autonomous testing, operations, and logistics in general, the vehicle is kept in a state which facilitates switching between robotic and human control without any need for mechanical alterations. Each actuation subsystem is controlled through independent switches accessible to a human observer sitting behind the front passenger seat. This allowed a great amount of flexibility for testing since some functionality of the system could be controlled by a human driver while other parts were completely autonomous. In addition, modifying the vehicle for normal driving and returning it to a state that is street legal can be performed within minutes.

A. Vehicle Actuation System

1) Steering Actuator

The RAV4 is equipped with electric power steering. In normal operation a sensor measures the torque applied to the steering by a human driver. The sensor outputs an analog signal (Figure 5) which the engine control unit (ECU) interprets as a torque and input to the assist motor which in turn applies an assisting torque to the steering column.

In autonomous operation, the torque sensor signal is replaced by a simulated signal provided by an analog output card (Advantech PCI-1721) installed on the QNX onboard control computer (Figure 7). Care was taken to ensure that the simulated signal was close to the expected signal so that it was accepted as genuine and fail-safe states weren't triggered.

This modification allowed the vehicle's inbuilt safety features (stability control, damping of too aggressive inputs) to remain operating. It also allowed the car to be returned to the original (street-legal) configuration by disconnecting the custom control and connecting the original torque sensor.

2) Throttle Actuator

Throttle control in the RAV4 is achieved with an analog position sensor located in the accelerator pedal. It provides a signal (Figure 6) to the throttle in a similar manner to that of the torque sensor for steering. As for the steering, the connector to the sensor is replaced with a custom adaptor for control input from the analog output card.

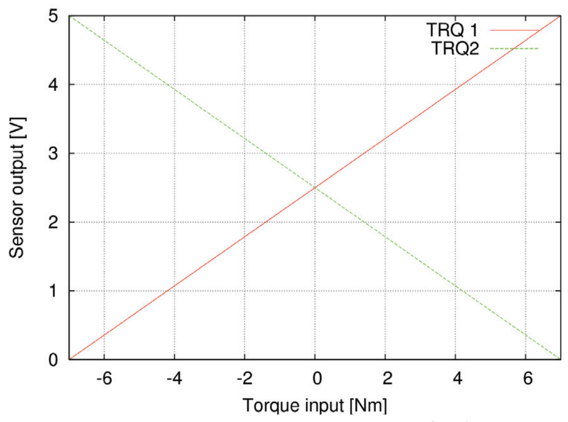


Figure 5: Mirrored signals output to the ECU for steering.

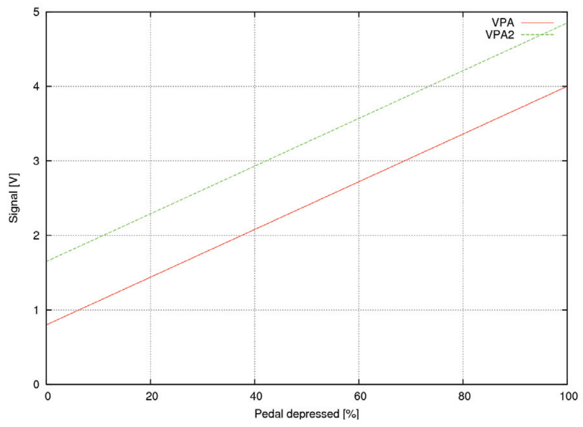


Figure 6: Offset signals output to the throttle.

3) *Brake Actuator*

Unlike the previous actuators, the brake does not have an existing electronic interface and thus required retrofitted actuation. In the previous Grand Challenge there were three main choices for brake actuation: linear electric actuators (majority of vehicles DGC 2005), (electro-) hydraulic actuators, and pneumatic actuators (a sizable minority, 2005). Linear electric actuators allow good position control. However, their major drawbacks are relatively slow speed and the lack of force control. Although hydraulic systems provide good position and force control, we dismissed this option in the early stages since there was little relevant experience in the use of these systems within the team.

We chose a pneumatic system as it is fast, allows good force control, and we had experience within the team using these systems. Note that one drawback with pneumatic actuators is that they are limited in position control.

We desired the ability for parallel brake actuation by a robotic and human driver as described in several designs for the 2005 Grand Challenge. However, to avoid space constraints in the driver's seat, the vehicle was converted to dual brake controls (as in driver training vehicles) and the front passenger seat removed. This conversion was performed by a certified mechanic ensuring the vehicle remained street-legal. The brake actuator could then be mounted to the existing strong points designed for the seat and attached to the dual brake guide cable (which would normally connect to an additional brake pedal). Note that for normal driving, the guide cable is mechanically detached from the actuator.

The pneumatic cylinder (Figure 8) is controlled using the same analog output card as the other actuators. The cylinder pulls the guide cable longitudinally to apply the brakes.

To ensure a failsafe system, we adopted the approach of the Austin Robot Technology Team from the 2005 DARPA Grand Challenge. The U-shaped assembly, termed the “sled” (constructed from Maytec rails), around the cylinder can move relative to the rest of the frame. The system is shown in failsafe mode (*i.e.* maximum brake applied) with the mechanical spring pushing the sled to

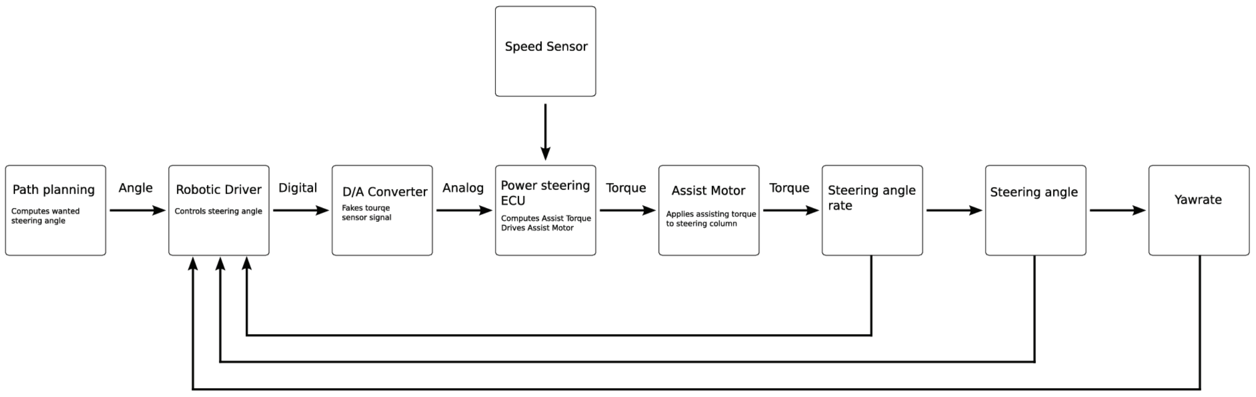


Figure 7: Overview of the steering system under robotic control

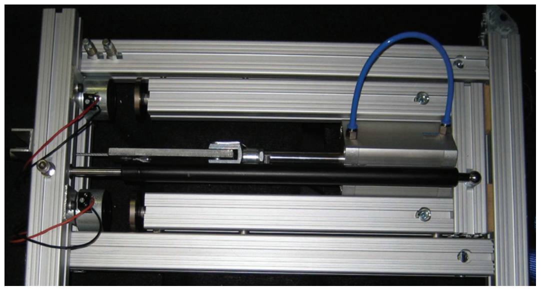


Figure 8: Retrofitted pneumatic brake & failsafe system.

the right and hence applying force on the brake. To reset the system from the failsafe state, the pneumatic cylinder overpowers the mechanical spring closing the gap between the sled and electromagnets. If power is supplied to the electromagnets, they hold the sled in the left position with the brakes released, giving the pneumatic actuator full control. This setup is failsafe in the sense that loss of power will result in the electromagnets releasing the sled and the mechanical spring applying maximum braking force.

4) Transmission Actuator

The transmission is controlled by a linear electric actuator. It is applied using digital output signals from the card used for analog output. The actuator can be mechanically disengaged quickly to return the vehicle to human control. We replaced the original lever guide so that only 1D movement was required to change gears.

5) Emergency Stop

The primary concern of the e-stop system is the safety not only of the team members but also of the general public, with an additional concern of ensuring that damage to property and equipment does not occur. Furthermore, it was required that the vehicle be maintained in a manner in which it can be switched back to a “road legal” condition. Thus, the e-stop system implementation and control of the actuators should never be allowed to interfere with normal driving on public roads. To guarantee zero interference during normal driving the e-stop system and actuator control circuitry are physically disconnected from the vehicle controls ensuring that the two systems are isolated. The current wireless e-stop (Hetronic ERGO V4, operation frequency = 434MHz), is designed to be easily interchangeable with the DARPA provided e-stop via a single connector.

To maintain simplicity our implementation of the e-stop system is limited to the three e-stop modes as specified by the DARPA guidelines. For safety and reliability it was decided to implement the e-stop modes and their transition completely in hardware. This prevents potential software bugs from resulting in an unknown condition or state of operation. For this reason a microcontroller-based embedded system was not used.

To meet the e-stop requirements, a state machine was devised with the corresponding state diagram as depicted in Figure 9. The state machine allows free transition between the RUN and PAUSE modes and a direct transition to DISABLE from either of these modes. The e-stop DISABLE mode is latched and requires manual intervention for recovery and subsequent transition to the e-stop PAUSE mode. The system always starts up in the e-stop DISABLE mode which is the default passive mode.

During both PAUSE and DISABLE modes, braking is actively applied. In PAUSE mode this is achieved by the e-stop system

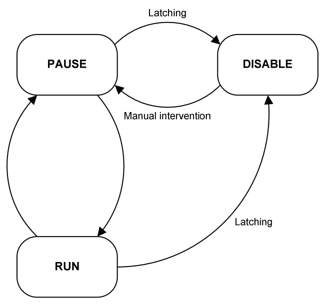


Figure 9: E-stop state machine

overriding the controls and applying the brakes at a predetermined threshold calibrated during testing. In DISABLE mode this is achieved by cutting power to the electromagnets which controls the failsafe braking system and results in maximum braking pressure. As the DISABLE mode is a latched mode, this condition, along with the engine cut-out and disconnection of computer actuation cannot be recovered from without human intervention.

6) Electrical Design

The vehicle electrical design can be broken down into two key components:

- Power design –Secondary battery system, charging and isolation, wiring and protection, power supplies
- Circuit design – Analog and digital circuit design for vehicle actuator control, sensing, and emergency stop system

a) Power design

The vehicle power design is based on a dual-battery system which includes a pair of secondary batteries connected in parallel to the primary vehicle battery and charging system via an automatic battery isolator. To utilize the original vehicle charging system, the design incorporated a 12V input.

To make room for the secondary battery system, all related components, and power supplies, the front passenger seat was removed and replaced with a 19" rack mount enclosure. Installed in the front passenger compartment are two 12V secondary sealed lead-acid (SLA), deep cycle batteries rated at 100Ah. These batteries act as a buffer between the vehicle primary battery/charging system and all custom electronics.

To ensure that all power requirements were met, a conservatively rated power budget was calculated taking into account all required electrical equipment (including 4x computers, 4x lasers, IMU, and GPS). Power consumption was (conservatively) estimated at 510W (42.56A from the 12V system) including de-rating for losses and efficiency, etc. This doesn't include the 200W load from the air compressor which although a fairly strong load has a very low duty-cycle. The secondary battery available capacity was de-rated for utilization within conservatively chosen parameters such as minimum charge and discharge states. Finally, run-time estimation was calculated using the Peukert equation to characterize battery capacity under load. Under these calculations it was decided to ensure that the secondary batteries are always maintained in a positively charged state via the vehicle charging system, and the required battery capacity was selected based on this decision.

To meet the additional load placed on the vehicle charging system the original alternator was replaced with a 150A alternator providing a 50% increase in capacity. To increase reliability, passive protection for voltage surges was installed therefore clamping any serious and potentially damaging voltage spikes on the noisy vehicle power system. Circuit breakers were used for the high current portion of the wiring to reduce voltage drops typically resulting from the use of high current fuses. Additionally, circuit breakers protecting the primary battery and each secondary battery provide the convenience of selectively isolating the batteries for servicing and maintenance.

As the vehicle is required to operate without human intervention, special attention was paid towards maintaining a reliable electrical design optimized for reducing power dissipation and minimizing failures. A number of key areas of the design were investigated. All high power items were adequately over-rated as a safety margin and the vehicle air conditioning is maintained at a temperature to reduce heat-induced stress of components mounted within the vehicle.

DC-DC converters (Amtex Electronics) are used for all circuitry powering sensitive equipment and were chosen with attention to rating and efficiency. Efficiency was a key point of focus in the reduction of the system power consumption. The dominant source of power consumption in our power budget is the computers and typical computer power supplies follow poor efficiency ratings. For example, power supplies meeting the ATX standard require only a minimum efficiency of 70% at full-load. To significantly decrease our power consumption we used fan-less power supplies (Minibox MT-ATX) with 94% efficiency.

All electronics are housed in a power/control box which acts as a power distribution and protection box. A master on/off switch controls the flow of current to the power/control box which in turn distributes power to all electronics. Essentially, this provides a single point for disabling all non-standard equipment such as computers, sensors, and vehicle actuation circuitry. This is important as it guarantees that power is disabled to all electrical systems if maintenance is required or if the secondary electrical system should be turned off in an emergency.

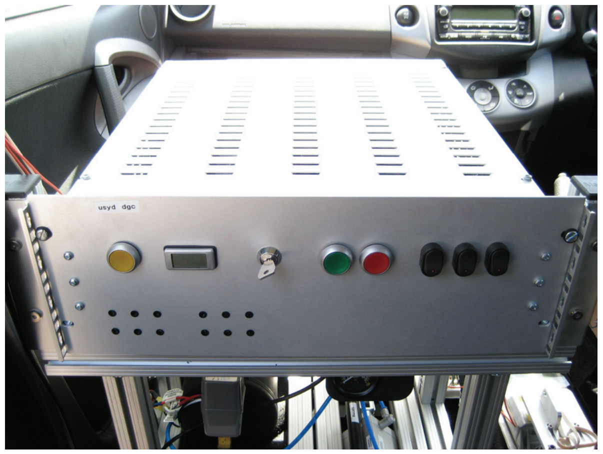


Figure 10: Main power/control box. The yellow (left) button indicates DISABLED mode, the green (just right of the key) RUN mode, the red (left of the three switches) PAUSE mode. The three switches to the right independently provide power to the individual actuators. The system is disabled without the key, and voltage to the system is displayed on the LED.

b) Circuit Design

The core circuit design is centered on control of the vehicle actuators enabling autonomous control from the computer, followed by the design of the vehicle emergency stop system.

A computer data acquisition card (Advantech PCI-1721) supports analog outputs and digital I/O which was used for vehicle control. The computer outputs an analog signal for each actuator which is translated by analog circuitry in the power/control box to output voltages required by the actuators. Each actuator control circuit contains a differential op-amp which is connected to the computer using shielded twisted pair wiring. Differential signals were used to ensure that any induced noise in the signal wiring is common-mode and is rejected. Additionally signal wiring and power wiring were routed on separate sides of the vehicle to minimize any induced noise. The actuator control circuitry translates the computer-generated differential signal to two single-ended signals for both the throttle and steering, mimicking the standard vehicle throttle and steering control signals. For the brake actuator, the circuitry translates the differential signal into an industrial standard 0 – 20mA current loop which is used to interface to the brake control regulator. This current loop is used as it can detect an open-loop condition and perform an emergency override.

Digital circuit design consists of the e-stop system and computer digital I/O control. This includes watchdog monitoring of the control computer and management of the e-stop system modes. Additional digital circuitry also interfaces the computer digital I/O lines for monitoring and control of various aspects such as detecting the e-stop system state and controlling items such as the vehicle indicators, actuation of the transmission, and resetting sensors such as lasers. The digital I/O design includes a combination of passive and active circuitry such as optical isolation over-voltage clamping.

B. Actuator Controllers

Since the trajectory planning is handled at the motion planning level, our low level control interface consists primarily of vehicle speed (throttle and brake) and steering angle, both executed using PID feedback loops. The controllers also consider secondary state information (e.g. which gear is currently active, car error states, etc.).

1) State Information

Using the state information provided by the vehicle's CAN-Bus and the Inertial Navigation System, we were able to accurately characterize the car's parameters. Most importantly, actuator positions could be mapped to actual states such as acceleration/deceleration, speed, and steer angle for precise low level control.

We are using an optoisolated CAN-controller card by PEAK System (PCAN-PCI) to interface with the vehicle's CAN-Bus. This card can be used under the real-time operating system QNX, using a third-party driver by BitCtrl Systems. With this setup we achieve high data rates (500kb/s for the vehicle's CAN-bus), high frequency (up to 80Hz, depending on message priority), and excellent timing accuracy.

2) Steering Controller

Steering was the simplest controller and just required a PID loop around the error in steering angle. The built in power steering motor provides the capability to turn the wheels from lock to lock in 1.2 seconds. However, the PID gains that we are currently using have been tuned for smooth operation and take approximately 2.0 seconds.

3) Speed Controller

The speed controller consists of two PID controllers (brake and accelerator) and an arbiter to choose the controller for a given situation. The average speed of the rear wheels (calculated from the wheel encoder information on the vehicle's CAN-Bus) provide the feedback with corrections at a frequency of 80Hz. The arbiter allows only one controller to run at the same time *i.e.* braking and accelerating cannot occur simultaneously. It also considers the vehicle state, so that the maximum RPM cannot be exceeded and the throttle can only be applied while in gear.

Control of the brake was more difficult than the throttle due to stiction in the mechanical system. As a result, brake force was not proportional to the applied actuator pressure. Instead, the effective braking force would remain relatively constant and at irregular intervals jump to match the actuator force. Further testing showed that small variations in actuator force generally resulted in unpredictable and jumpy responses in effective braking force. However, step inputs of varying size resulted in accurate proportional brake force response.

To solve the stiction problems in the control system, we quantize the output of the brake controller. Whenever the output of the

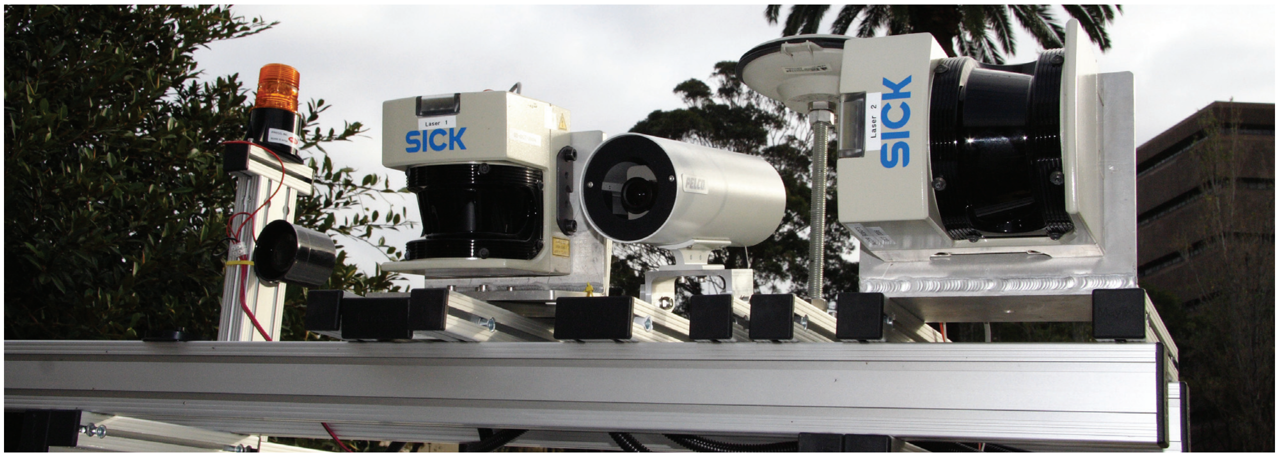


Figure 11: Current sensor suite – SICK lasers, INS, and camera.

controller is commanded to change, the actuator is set to zero force for 60 milliseconds and then returns to the quantized set point. This ensures the effective brake force matches the controller's output by transforming a small correction into a large one and eliminating the effects of stiction.

IV. Perception

Perception of the environment and the vehicle state is performed by on-board sensors. Most of the vehicle sensors have been mounted on the custom built roof rack (Maytec rails) shown in Figure 11.

Visibility of the terrain is greatest on the roof and obstruction of signals received by the global positioning sensor (GPS) is at a minimum in this position. Fusion of GPS and output from an Inertial Measurement Unit (IMU) provides the vehicle navigation solution. The IMU is mounted to the chassis above the transaxle. Currently, perception of the environment is performed using two SICK LMS291 lasers and a single camera (Hitachi HV-F31F, 3CCD progressive scan). It is planned that extra lasers, radars, flash ladar, and infrared cameras will be added to the sensor suite in the near future.

One laser is pointed forwards at a pitch of 7^{o}. This laser records the cross-section of the terrain at a range of 16m in front of the vehicle. The second laser is mounted vertically and is used for estimating the relative pitch of the road out to approximately 25m depending on the reflectivity of the road. The lasers are connected to a Bluestorm/Express PCI-Express 4-port serial card and communicate using RS-422 in order to run at a 500k baudrate. The clock on-board the serial card was replaced with a clock that could provide this non-standard speed.

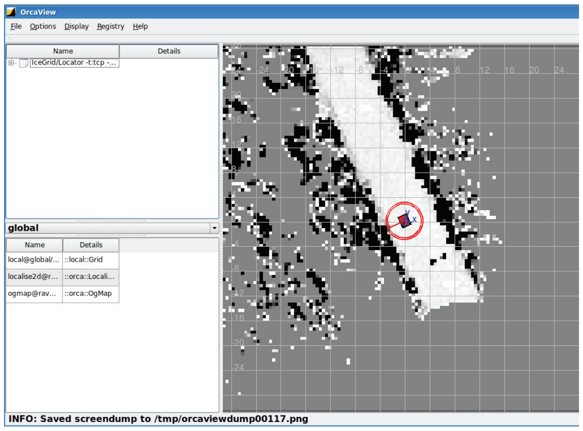
The camera is used for detection of lane markings out to approximately 40m limited by resolution of the camera. It is connected to the computers via firewire.

A. *Navigation and State Estimation*

Estimating the vehicle state accurately is a key requirement for sensible path-planning and representation of obstacles. Errors in the pose estimate can result in poor terrain maps with phantom obstacles leading to seemingly poor control decisions. Thus it is important that the navigation solution, at least at the local level, is smooth and locally accurate.

Navigation is the central sensor system. All other sensors are registered with the navigation and fused using this information.

Therefore, we chose to acquire a quality navigation system, a Novatel ProPak-V3 GPS receiver, combined with a Honeywell HG-



Name	Details
IceGrid/Locator -t:tcp -...	

Name	Details
local@global/...	::local::Grid
localise2d@r...	::orca::Locali...
ogmap@rav...	::orca::OgMap

```
INFO: Saved screendump to /tmp/orcaviewdump00117.png
```

Figure 12.: Snapshot from the `orcaview` component illustrating the traversability map. Note the GUI tools to the left, which indicate which components are running, and what content they provide. White indicates traversable, black indicates an obstacle larger than 30 [cm], and shades of grey: objects 0-30 [cm]. Cell sizes are 50 [cm].

1700. We use OmniSTAR HP global GPS corrections, and we are currently developing a navigation filter to fuse IMU data with wheelspeed based odometry.

B. *Static Obstacle Detection and Representation*

For reliability and robustness, we have chosen to use a standard grid map data structure often used in robotics to represent obstacles in the environment [20][21]. We closely follow the obstacle detection technique used by Thrun *et al.* [21] in the last challenge in which the scans from downwards looking lasers are projected into the vehicle's local coordinate frame resulting in a 3D point cloud for each laser. Laser points are grouped into 2D cells which combine to make a surface grid. The traversability of a cell is classified by vertical height differences $\left|z_i - z_j\right|$ between laser points within a cell. A cell with a height difference greater than some threshold, δ, is deemed not traversable. Unlike Thrun *et al.* [21] we did not need the additional machine learning layer required to compensate for small inaccuracies in pose. We found that accurate time stamps recorded using the hard real-time QNX operating system for both the navigation solution and laser scans resulted in no such error.

An example of a traversability map is shown in Figure 12. This map is a display version of the grid's data structure which is provided to the navigation and path-planning components.

C. *Dynamic Obstacles*

A major challenge for unmanned vehicles to navigate in urban traffic is the reliable detection of potentially dynamic obstacles such as other vehicles, whether they are oncoming, parked, slow, etc. Detection of vehicles from stationary positions is relatively well understood and addressed using a combination of vision and machine learning techniques (see for example [22][23][24]), such as the ones included in the popular open source vision package OpenCV [25]. However, the Urban Challenge requires detection of dynamic obstacles with close to 100% reliability and purely using vision can be problematic due to lighting changes, previously unseen vehicle types which could possibly be misclassified, field of view, etc. Thus we have chosen to use a combination of lasers, radars, and vision to address these problems.

1) *Lasers and Ray-Tracing*

The traversability map as described is not suitable for dynamic obstacles. Each cell *permanently* records height differences. So if a moving vehicle passes within the field of view of the lasers, the height at which the laser intersected the vehicle will be recorded at a corresponding cell. Once the observed vehicle continues, the height difference at the previously occupied cells will remain the same, leaving a non-traversable trail in the map. This poses a problem for the path planner as it will decide to avoid this trail treating it as an obstacle in the local world.

To avoid this issue, we chose to include "negative information" conveyed by a ray passing *through* cells between the laser itself and from where the laser return was recorded. To ensure that we have not received a spurious return, if two rays pass through a cell and the maximum height has been reduced, we update the height difference in two ways. If the cell is on the road, the height difference is reduced to zero as it is a reasonable assumption that the object was a vehicle, while a cell elsewhere is updated to the new recorded height.

This technique will fail if the vehicle is reversing as the laser returns from an object will reduce in height even if the object is static. Thus the map is not updated using forward pointing lasers while the vehicle transmission is in reverse.

D. *Experimental Sensors*

Although SICK laser range finders and visible spectrum cameras are our current primary perception sensors, these sensors will not guarantee long range static and dynamic obstacle detection. Briefly described in the following sections are a number of sensors and techniques we plan to integrate with the vehicle for increased reliability for obstacle detection.

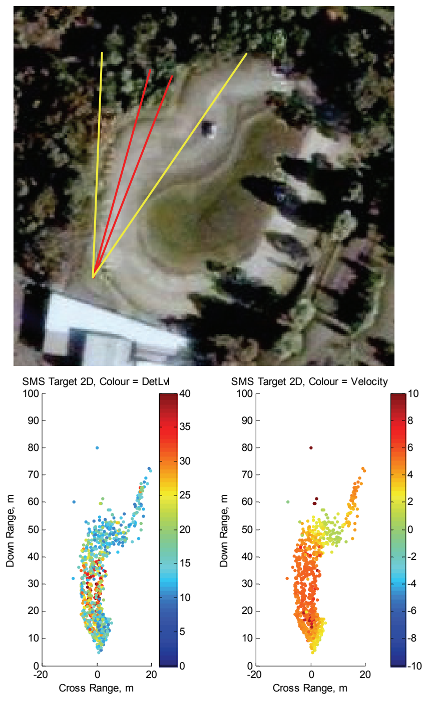


Figure 13: Combined data from 3 clockwise laps with the car in a different lane on each lap. LHS: an aerial image of the test track overlaid with the position and beam patterns of the two radars (TRW in red, SMS in yellow), RHS: scatter plots for the car traveling in each direction. The graphs depict the variation in SNR & radial velocity over the area of interest for the SMS radar.

1) *Radar and Tracking*

Many of the requirements for autonomous sensing of dynamic obstacles match those for automatic cruise control (ACC) radars: robust range, velocity and angle resolution of multiple targets over a wide dynamic range. But why use radar over alternate sensors such as cameras and laser scanners? Feature extraction from visual sensors is difficult and works best for well-behaved features such as lane markings and road signs. Laser scans provide a very good snapshot of the local environment, but their range is limited, target reflectivity is variable and velocity must be inferred from repeated measurements, assuming the same targets are seen on each scan.

Radar reflectivity measurements are also highly variable, but have the benefit of determining range, velocity and angle simultaneously, although the angular resolution and coverage is usually poorer than that of a laser scanner. Solid state radar sensors also have a much lower volume cost than mechanically scanned lasers, and have the added advantage of an all-weather capability, hence are likely to provide a better long term solution. At the current state (and cost) of technology, given that the volume cost of ACC radars isn't given to University departments, the optimal solution is therefore a combination of sensors, using each to their greatest strengths, and for radar that means detecting other moving vehicles. The question then becomes, is ACC radar sufficiently robust to perform this task?

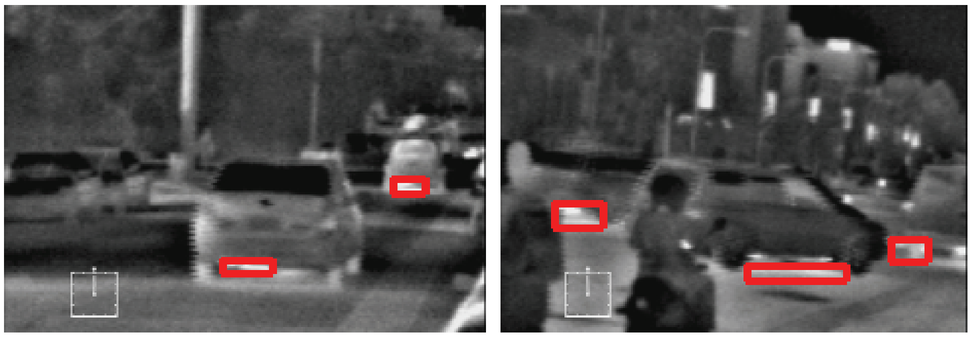


Figure 14. Example infrared images of vehicles in an urban environment. Red rectangles are thresholded regions indicating hot areas of a vehicle. The cross-hairs in the left hand corner of the images indicate the panning angle of the camera.

Two ACC radars at different operating frequencies (24 & 77GHz) were compared in both static and dynamic environments. The two radars were mounted on a pole and were positioned overlooking a 4-lane vehicle test-track, around which a single car was driven in both directions. An occupancy grid was created showing the strength of target returns over a section of the circuit. An example result from the dynamic environment tests is shown in Figure 14.

These results illustrate that ACC radars are clearly good sensors for this application and can detect vehicles over a very large range compared to that of a SICK laser. It is our aim to include at least two SMS radars on the vehicle.

2) *Thermal Infrared Stereo and Fusion with Laser Information*

All vehicles have some hot area that can be detected using thermal infrared (TIR) sensors. It is proposed that fusing information from a TIR camera with a laser range finder would greatly increase the reliability of detecting vehicles.

Figure 14 shows two examples of white-heat thermal infrared photos of moving vehicles, taken while driving around urban environments. Red squares indicate thresholded areas in the image that are classified as a part of a car. Fusion of thresholded regions with laser range data has shown that distances to hot objects with centimeter accuracy can be achieved at 25 frames per second. Using only this sensor and rudimentary algorithms, 85% of cars driving in the same direction are detected, cars perpendicular to our vehicle are detected 80% of the time, while oncoming cars are detected 70% of the time. False positives arise due to reflections from metallic objects but these will be filtered out based on expected position of the road, as well as feedback from laser scans.

3) *Stereo Vision in the Visible Spectrum*

The goal of this task is to detect moving objects using a stereo camera system mounted on a car. The case of multiple moving objects seen from two perspective views has been studied in [1][2][3][4], but all of these are either specialized to the case of estimating fundamental matrices (assuming non-planar objects), or are also capable of estimating homographies but cannot choose between the two. The approach closest to our two-view algorithm is that of Torr in [5], although his solution is not invariant to planarity, and his algorithm is not suitable for real-time use.

Following Longuet-Higgins in [6], we envisage the vehicle as being in motion relative to an object in the scene. Note that since we always know the image coordinates (x, y) and that we can approximate $(\dot{x}, \dot{y})$ using optical flow, then this system is linear in angular and linear velocity (W, V) when we

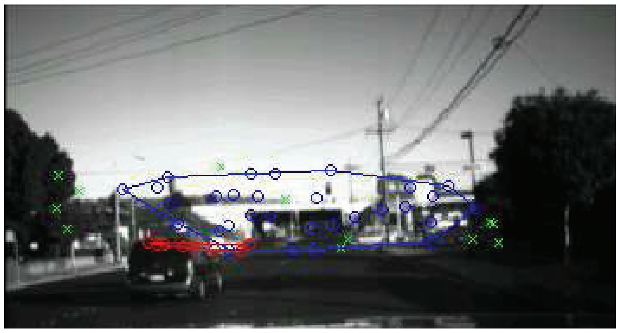


Figure 15. Being passed by another vehicle. The blue region is the largest labeled independently-moving object, and the red areas are other labeled independently-moving objects. The green points are unlabeled.

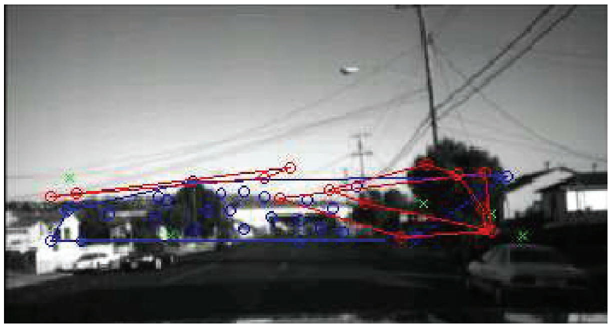


Figure 16. A sample experimental result for the combined system. The blue region is the largest labeled independently-moving object, and the red areas are other labeled independently-moving objects. The green points are unlabeled.

know Z (distance).

We have separately developed a suitable two-view motion segmentation algorithm, and an efficient segmentation-maintenance procedure. Unfortunately, when we simply combine the two-view segmentation with the segmentation maintenance procedure, we often get results like those shown in Figure 16. This is not surprising, since we are performing anomaly detection despite many of those anomalies being due to errors implicit in a real-time feature tracker.

Our approach is a dense algorithm that takes advantage of all of the information available in the image sequence instead of attempting to differentiate independently-moving objects based on only a few (often incorrectly tracked) features. Although we do not expect an entire object to appear to move identically in the image sequence, we do expect neighboring pixels on the same object to move much more similarly than those on the border between independently-moving objects. This immediately suggests a graph segmentation approach (as opposed to traditional image segmentation), to which end we construct a graph with edges between neighboring pixels that have weights given by the similarity between the pixels' (image) movements over approximately 10 frames to estimate a value for each pixel in a reference frame.

Once we have built the adjacent-pixel motion similarity graph, we use the graph segmentation algorithm of Flake *et al.* [7] to determine the independently-moving objects. This method makes no assumption about the number of independently-moving object groups, and is controlled by a single parameter. We use the heuristic version of this method, which provides the same guarantee while reducing the number of required max-flow/min-cut operations as equal to the number of groups (observed independently-moving objects). For the max-flow/min-cut subroutine, we use Boykov and Kolmogorov's algorithm [8]. This algorithm removes the theoretical runtime guarantee of many max-flow/mincut algorithms, and in [8] is shown to run an order of magnitude faster than the best theoretically-guaranteed (practical) computer vision algorithms.

4) *Flash Ladar*

3D scanning ladars have been proposed for ground vehicle navigation, obstacle avoidance, and detection of other vehicles present in the path of an autonomous vehicle [26]. Unlike a SICK laser or a 3D scanning ladar that scans using a collimated narrow laser beam, flash ladars illuminate an entire scene with a laser pulse. A pixel array measures intensity and optical time of flight to objects in the environment resulting in a 3D model of the scene Figure 17.

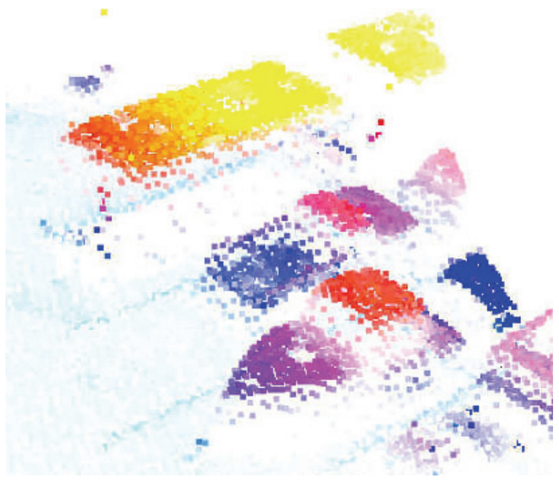


Figure 17: Vehicles detected at a >100m range using the flash ladar. The camera was mounted above the vehicles.

We are developing detection, tracking, and representation techniques for a flash ladar by Advanced Scientific [27][28]. A combination of point tracking, kernel tracking with multi-view appearance, and matching shapes will be used [29][30].

E. Lane Detection

Unlike the first two grand challenges, abstract objects such as lane markings must be detected and represented for high level decisions. Lanes only make sense in a world where traffic rules are specified. Thus lane markings must not only be detected but reasoned about for this third DARPA challenge. Although, reflective paint can assist lasers in detecting lane markings, we chose to use visual sensors in case of the scenario in which poor or non-reflective lane markings are on the road. However, detection of lane markings in image data is not a trivial task. That is mainly due to the non existence of unique models, poor quality of lane markings due to wear, occlusions due to the presence of traffic and complex road geometry.

The potential pitfalls when using image data leads us to the conclusion that any method using a strong model of the road (lane), will eventually fail. Thus, weak models must be applied. As a result of this, particle filters are a good choice for the task of lane marking detection due to their ability to handle poor process models. The proposed method is to use an inverse perspective mapped image (IPM image) and identify all pixels that could belong to a geometrical model of a lane marking. After initial segments are identified an exhaustive search using lane width is used to identify seeding points for a particle set. The particles are propagated from the bottom to the top of the IPM image consolidating the pixels belonging to lane markings, thus, identifying lanes in the image which are stored as a trail.

1) Inverse Perspective Mapping

Lane detection is generally based on the localization of a specific pattern (the lane markings) in the acquired image and can be performed with the analysis of a single still image. The method for lane marking detection employed in our paper is based on the Generic Obstacle and Lane Detection (GOLD) implementation [31]. In order to fit the lane model to the acquired road images geometrical image warping is performed with Inverse Perspective Mapping (IPM). The IPM technique projects each pixel of a 2D perspective view of a 3D object and re-maps it to a new position constructing a new image on an inverse 2D plane. Conversely, this will result in a bird's eye view of the road, removing the perspective effect. Each pixel represents the same portion of the road, allowing a homogeneous distribution of the information among all image pixels. To remove the perspective effect, it is necessary to know *a priori* the specific acquisition conditions (camera position, orientation, optics) and the scene represented in the image (the road).

The resolution of the remapped image has been chosen as a trade-off between information loss and processing time. At the moment we are using the blue channel of an RGB image only as it provides the highest contrast between the lane and tarmac color. Another potential space to use is the C1C2C3 space which gives marginally better results but at the cost of computational effort.

As mentioned above, the mapping assumes that the road is flat which does not hold in most cases. Therefore, we use a vertically mounted laser to detect the relative pitch of the camera and the road. The result can be seen in Figure 18.

Lane marking detection is performed on the remapped image applying a lane model which stipulates that a road marking is represented by a predominantly vertical bright line (lane marking) of constant width surrounded by a darker region (the road). Thus, the pixels belonging to a road marking have a brightness value higher than their left and right neighbors at a given horizontal distance. A vertical edge in an image conforms similarly to the same principle; the intensity difference between neighboring pixels must be over a threshold to be validated. Therefore, an exhaustive search across each row of the image will produce potential lane marking candidates where the match probability can be measured with the edge quality *i.e.* the difference in intensity.

Each candidate pixel is classified as an observation. If it conforms to a fixed lane

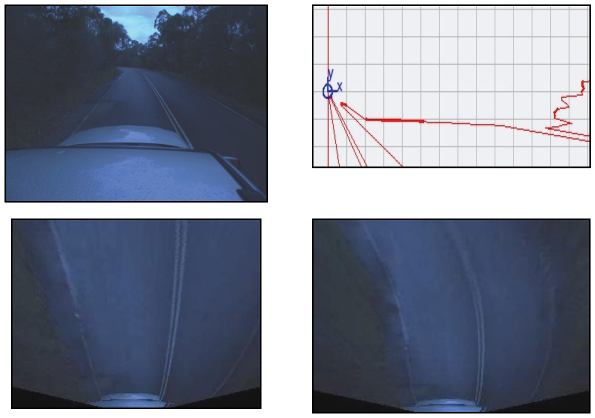


Figure 18: a) Original image, b) laser scan used to correct pitch, c) incorrect IPM Image, d) image with correct pitch.

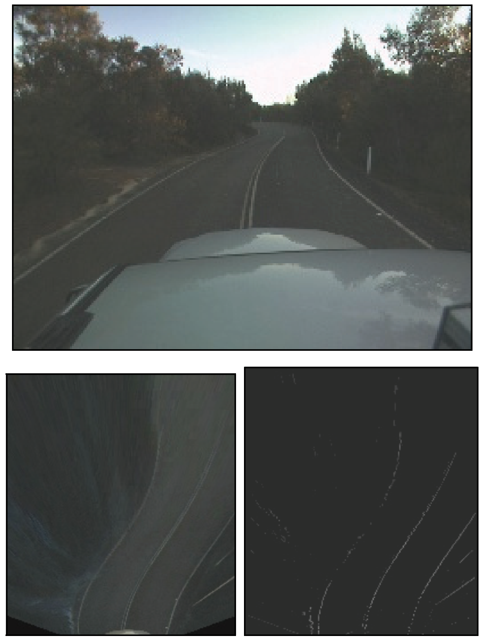


Figure 19: a) original image, b) IPM image, c) detected lane marking segments

width (defined by the RNDF) with candidate pixels on the same row in the image, it is fused into a particle filter which propagates vertically through the image starting at the car position. The process model is Markovian incorporating general knowledge about road design. In this model a straight lane is more likely than curved. Furthermore, a low degree of curvature is more likely than a high degree of curvature. This property is derived from the observation that larger degrees of curvature are usually only present for relatively short times. Finally, we also take into account that there is a certain maximum degree of curvature, which will by definition not be exceeded. An example of detected lanes is shown in Figure 19.

2) *Centerline calculations*

Finally the virtual centre line is calculated from the lane markings and represented as a polyline. It is then communicated to the path-planner and incorporated as a constraint.

V. NAVIGATION AND PLANNING

We use a layered approach to planning and

Algorithm 1 Route-planning overview.

```
1: Preprocess RNDF to build graph $G$
2: **while** running **do**
3:     Localize robot in $G$
4:     Get obstacle (blockage) data
5:     Add blockages to $G$
6:     Search $G$ for best path to next checkpoint
7:     Send initial path segment to street-level planner
8: **end while**
```

control. Our approach consists of a route planner, a street-level planner, and a reactive local planner. This layering achieves a number of benefits. First, it is more efficient to plan in a layered manner. We only plan in high detail through a region about which we have information. Also, it is an important design principle of the team's entry that driving is performed at a local level. We do not need to know where we are in a global space to retain safe control of the vehicle. Finally, although both the route and reactive planners are in metric spaces, the layering allows these metric spaces to be decoupled. This enables us to use two navigation solutions: one smooth, but perhaps not globally accurate (from the IMU), and another one globally accurate, but usually not smooth (from the GPS unit). The local obstacle detection and tracking uses the smooth navigation solution, and the global positioning uses the globally accurate positioning. We describe the three layers of our planning system (as shown in Figure 2) in this section

A. Route Planner

The route planner reads in the RNDF and MDF files supplied to the team, and uses them to plan a high-level path that fulfills the mission defined in the MDF. The planning proceeds as follows. On start-up, the RNDF is processed to produce a connected graph. While driving, this graph is searched each second to discover a route to the next checkpoint, and then the route is processed to determine where we should next stop or exit the current street. That next exit or stopping point is passed to the street-level planner, along with any available information regarding the current road segment such as speed limits, estimated Euclidean distance to the exit point, and road curvature. The algorithm is summarized in pseudocode in Algorithm 1 and discussed below.

1) Planning our Route

Once the graph is constructed, we can plan over the graph using the standard $A*$ search algorithm [32]. In this search, links have a cost proportional to their length. Lane change links have a linear penalty. Exits also have a small penalty. The search starts at the closest location in the RNDF to our current global position, as supplied by our GPS/INS unit. It runs to the next checkpoint the vehicle is required to pass through. The resulting path is then traversed looking for the first stop sign or exit. That exit or stopping point is passed to the street-level planner.

During a run, it is possible to modify the graph that is being planned. New waypoints can be added, increasing the accuracy of the RNDF and so the optimality of the search. Lanes can also be marked as blocked, which would cause a different plan to be found during subsequent iterations.

B. Street-Level Planner

The street-level planner is the component in our system that plans for traffic. The role of this layer is to plan ahead along the current street, avoiding other cars, until the next stop sign or exit (which was passed down by the route planner). Plans are constructed in a 1.5 dimensional, discrete space. That space has distance down the current segment in three-meter increments as one dimension, and lane number as a second dimension. Each discrete cell can contain an obstacle passed up from the sensor fusion. That obstacle is assumed to have a discrete velocity along the road.

Algorithm 2 Street-level planning overview.

```
1: while running do
2:     Get street segment and goal from route planner
3:     Get obstacle data
4:     Build Nagel-Schreckenberg CA model NS
5:     Build MDP from NS and road rules
6:     Solve MDP using Real-Time Dynamic Programming
7:     Transform actions into waypoints in global coordinate frame
8:     Send waypoints to motion planner
9: end while
```

Sensors also pass up the rough shape of the road ahead in terms of the curvature of each segment. This can be used to reduce speed ahead of sharp corners. Figure 20 illustrates this problem formulation.

Planning among moving obstacles is challenging, but we can exploit the fact that vehicles are generally constrained to driving within lanes to develop an elegant solution to this problem. In particular, we use a real-time dynamic programming (RTDP) algorithm [33] to solve a Markov decision process (MDP) with a *Nagel-Schreckenberg* traffic model [34]. We then output a list of actions to the low-level reactive planner such as: 'go slowly', 'go quickly', 'stop', 'change lanes left' or 'change lanes right'. We list pseudocode for our approach in Algorithm 2 and detail important components in the following subsections.

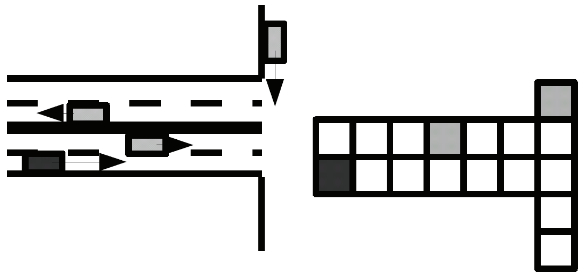


Figure 20. The street-level planner finds a sequence of actions to navigate the current street segment, terminated by an intersection, stop sign, or turn. It takes as input road information from the route planner, and current position/velocity of obstacles from sensor data (left), and represents this in cellular-automata form (right). Cell update rules describe how moving obstacles (cars) interact with each other over time. This system is solved using dynamic programming, resulting in a sequence of actions that move our car through the current segment to a goal position (provided by the route planner) while avoiding collisions. These actions are transformed into metric coordinates using sensed lane geometry and output to the motion planner layer for execution. The entire process runs at >2Hz.

1) *Real-Time Dynamic Programming*

Our street model is a discrete stochastic system, which we assume to be fully observable at this level in our system. This type of system is known as a *Markov Decision Process* (MDP) [35]. Since Bellman in the 1960's it has been known that these systems can be solved using Dynamic Programming [33]. This requires building a table that maps states to the value of being in that state: the *value function*.

Since our state space is too large to exhaustively explore, even for a single street, the state space is sub-sampled. Rather than keep an array of all possible states, we keep a hash table [36] over the states. We then start our agent in the current state and simulate forward, only updating the values of states that we reach. The forward simulation is stochastic, and so many runs are required to achieve a reasonable coverage of the relevant space. This relevant region is significantly smaller than the entire state space, however, which makes the technique tractable.

We also make a change to the method described above for updating values. We have assumed that the reward and transition functions are known, and while we have assumed specific functions here for our system, those functions are not efficient to iterate over. Again, our response is to sample. The update rule, known in the reinforcement learning community as the Q-learning update rule [37], can be applied at the same time we sample our state space. We are also able to cache information between planning runs simply by not clearing the hash table between runs. If the hash table becomes over-full, we use a "least-recently-used" ejection scheme to provide space.

2) *Transition and Reward Functions*

In an MDP, the transition function describes how the state of the system evolves over time. Formally, we have a function $P_{s,a}(s')$ that yields the probability of transitioning into state s' from state s after taking action a. The system state in our case includes dynamic obstacles, so the transition probabilities must model the position and movement of opponent cars in relation to ours. In effect, this requires the transition function to act as a traffic simulator.

Although many complex traffic simulation packages are available, it is possible to model traffic with surprising accuracy using a simple cellular automata-based approach known as the Nagel-Schreckenberg (NS) model [34][38]. NS stochastically predicts the behavior of individual cars in a road network It models a road segment as an array of cells, where the cells can be either occupied by a vehicle or unoccupied. Cells are updated using a small rule set: 1) accelerate if possible, 2) decelerate if nearing an obstacle, and 3) randomly decelerate with probability p.

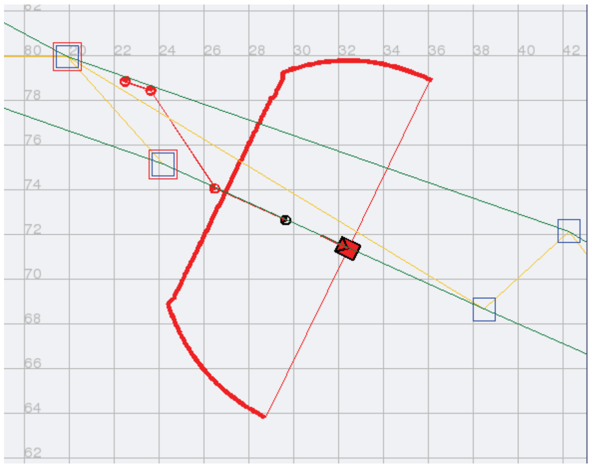


Figure 21. Screen capture of our visualisation tool as the car approaches a lane change. The blue boxes are RNDF waypoints. The red boxes are RNDF waypoints that the car plans to travel through. The small red circles are low-level waypoints for the local navigation component to follow. The large red D shape is the laser return. Note that once the local waypoints are past the last RNDF waypoint in the current lane, they are offset into the neighboring lane to cause the car to change lanes

The macroscopic behavior of NS has been empirically validated and it is popular in the traffic engineering community due to its computational efficiency. This suitably models traffic behavior for our purposes, but in order to plan our car's behavior we must define a reward function. The reward function assigns an instantaneous numeric reward to state-action pairs. Actions that result in undesirable outcomes such as collisions are punished with a strong negative value, whereas desirable actions such as making appropriate lane changes and proper behavior at intersections are rewarded with a positive value. The reward function allows us to encode any other necessary road rules as well, such as activating turn signals and obeying speed limits.

3) *Street-Level to Low-Level Mapping*

The local reactive planner in our system takes commands in the form of waypoints that the vehicle treats as goals. The street-level planner gives commands in terms of where to change lanes and a maximum speed. We have a simple system that maps between these two types of commands.

The mapping uses the detected centerline of the lanes. In particular, we loop forward along the detected line until the distance for the next street-level command has been reached. We then place another waypoint on the centerline at that point. If a lane change is detected, we offset the waypoints so as to place them in the appropriate lane. Figure 21 illustrates the waypoints generated by the street level planner during a lane change.

C. *Motion Planner*

The waypoints generated by the street-planning level provide goal information to the final component of our planning system, the low-level motion planner. The responsibility of this component is to control the car to visit the goal waypoint sequence while avoiding

Algorithm 3 RRT-based motion planning.

```
1:  while running do
2:    Get new waypoints
3:    if no waypoints then
4:      Send stop action to actuator controllers
5:    end if
6:    Get obstacle data
7:    Get lanes data
8:    Initialize tree to previous best path
9:    for each iteration = 1 to NUM_EXPANSIONS do
10:     With probability p:
11:       Randomly choose node n from tree
12:       Randomly choose safe control values
13:       Predict new state s
14:     With probability 1-p:
15:       Randomly choose node n from tree
16:       Use heuristics to choose safe control values that
            guide robot toward goal
17:       Predict new state s
18:     Compute distance to nearest obstacle from s
19:     Compute weighted cost of leaf-to-root path from s
20:     if cost  is unfavorable then
21:       Prune path
22:     else
23:       Add s to tree
24:     end if
25:   end for
26:   Store best-cost path
27:   Send initial control action in best path to
        actuator controllers
28: end while
```

obstacles and staying in its lane. This task is a standard type of problem in robot motion planning, and this useful decomposition is a strength of our layered planning system design. However, in our case we are planning for a car-like robot, and this involves differential constraints. Motion planning under differential constraints remains an extremely challenging problem in the literature.

We developed path planning components in parallel, each of which utilized a predictive control technique. One relies on heuristically encouraged random behaviors, while the other performs online optimization. Since each of these planners has implicit weaknesses, we designed a supervisory controller which selects the set of best inputs submitted by each component during each control step. This provides extra robustness to the system, reducing the probability of falling into a local minimum solution.

Two main categories of algorithms for motion planning are the well-established combinatorial methods, and the more recent sampling-based methods [39][40]. Because combinatorial methods in general do not consider nonholonomic constraints or any type of dynamics, the current best options are sampling-based methods. We chose to use a variant of the popular Rapidly-Exploring Random Trees (RRT) algorithm [40][41]. RRTs have been used successfully with real robots and demonstrate acceptable real-time performance [42][43]. We modified the basic algorithm to incorporate lane following and to respect velocity and acceleration bounds. We list pseudocode as Algorithm 3 and discuss the algorithm in detail in this section.

1) Rapidly Expanding Random Trees

The standard RRT algorithm searches for a collision-free path between start and goal states. It searches by iteratively expanding a search tree until it finds the goal. Here the start state is current robot position, and goal states are waypoints provided by the street-level planner. We expand the tree by forwards integration of control values from a randomly-chosen node in the tree. The control values respect velocity and acceleration bounds and are either determined by heuristics that lead towards the goal or are chosen randomly. We prune unfavorable leaves in the tree using a weighted cost function that takes into account Euclidean distance to goal, distance to obstacles, proximity to lane centre, and path length. Because of this pruning, our sampling is biased towards favorable regions of configuration space.

Obstacles are sensed using the laser scanner and provided to the planner in a grid representation. Lanes are processed separately by the car's vision system and provided in the form of lane centerlines, represented in piecewise-linear form.

The algorithm performs a fixed number of expansions (or a fixed time-distance into the future) and then returns the best-valued control action (desired speed, desired steering angle), which is then executed by the low-level hardware controllers. We interleave this planning with a check for new waypoints and current sensor data. Since waypoints do not generally change radically between calls to the planner, we store the last best path for use in initializing the tree in the next iteration. In fact, the vehicle rarely reaches any given waypoint before the street-level planner provides a new set (receding-horizon control).

We performed preliminary evaluation in simulation using the Player/Gazebo environment [44]. The CPU time spent choosing a control action ranged from 20 to 100 milliseconds on a standard workstation running both the simulator and the planner. Similar performance was observed during all our on-line hardware testing with the vehicle.

Because we expand the RRT tree using 'safe' control values, the resulting path should not violate the vehicle's dynamic constraints. An obvious downside to this biased sampling is that we explore a restricted region of configuration space. We have not observed problems in finding good paths in our simulation experiments, however, most likely due to useful heuristics and relatively simple obstacles.

2) *Optimized Nonlinear Model-Predictive Control*

In contrast to the RRT implementation, this technique takes advantage of known geometric properties of the vehicle model (as well as those assigned to obstacles and other vehicles) and performs geometric optimization of the optimal control problem to produce a suitably safe (and smooth) control sequence.

a) *Cost Function*

We seek to find the optimal control sequence u^* that minimizes a cost function subject to the vehicle model $x_{k+1} = f(x_k, u_k)$. The approach taken here for obstacle avoidance is similar to the one found in [12]. We only consider obstacles closer than a distance δ_3 from the vehicle. For each waypoint there is assigned a direction in which we want the car to move as it passes through, thus defining a “wayline” in the opposite direction of travel. Using that direction and our relative position, we use the distance from the vehicle to the wayline to make the vehicle

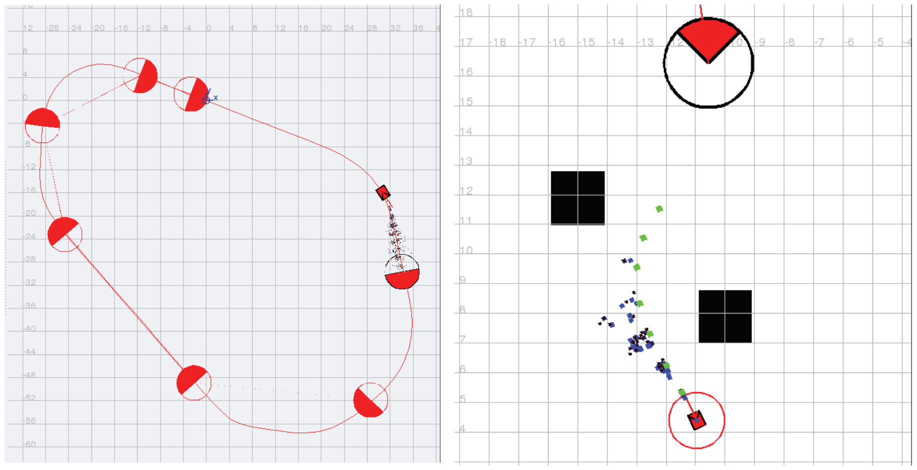


Figure 22: LHS: Simulation of motion planner execution in `orcaview`, a utility program provided by the Orca software package. The vehicle is drawn as a rectangle, search tree nodes are drawn as pixels emanating from the vehicle, large circles represent waypoints, and the curved red line represents the lane centerline. RHS: Example of reactive obstacle avoidance. Obstacles are black while traversable areas are white. The path chosen by the motion planner in order to safely reach the waypoint is represented by green squares and the possible paths that it searched over represented as blue squares.

smoothly arrive at the waypoint in the desired direction. The cost associated with the states and inputs favor some inputs and state behaviors over others, and also helps prevent saturation of the actuators over predefined values.

b) Optimization

Given the previous facts, the algorithm used in order to minimize the objective function is one that maintains the feasibility of the solution at all times, making successive approximations to a local minimum during each iteration. These properties are achieved by using a steepest descent method [9][10] together with the Armijo projected gradient method to compute the step size [11]. The output of the optimization process is the next *N* control inputs, and it is generally seeded with the previous set of control inputs. The optimization loop is "any time", meaning it can be stopped if a solution is not found in the time given by a real-time supervisor, thus providing an input which is suboptimal but still feasible.

c) Parameter identification

Finding the appropriate gains and weighting parameters for the cost function is quite often done based on experience or repeated simulation until the desired behavior is achieved. Upon proper identification of parameters we can assign weights to the cost function such that the autonomous behavior of the vehicle mimics the behavior of the vehicle with a human driver. We perform this by discovering the human driver's weights for the same function through analysis of choices made. Similar types of optimization problems have been successfully applied to a variety of problems, ranging from identifying parameters in biological systems [13], [15], solving air traffic flow problems [14] and to applications within aerodynamic design [16].

VI. Analysis

The nature of the Urban Grand Challenge requires a tight schedule for development and vehicle testing. Emphasis was placed on low level reliability i.e. robust hardware, controllers, and software drivers. This ensured error testing for higher level algorithms was decoupled from the base platform. The system was tested often, initially at our test facility in Sydney, Australia and recently at the mock urban environment facility in Berkeley, USA. Through simulation and continual online testing, we were able to complete an end-to-end system within 6 months which was capable of completing the tasks required for the video submission.

Although individual software and hardware components were tested independently for reliability through simulation and online testing, it was very difficult to measure the system's capability as a whole. Currently we are following the method of Thrun *et al.* [21] of mean time between failures, where a failure requires human intervention. We also currently monitor individual components and their health online. For each online and simulation test, we can log the performance of components allowing us to post-process the data analyzing failure points. We plan to develop a more quantitative approach for the upcoming qualification events.

In the later stages of testing, failure is virtually isolated to software and sensors. Problems with sensors have been addressed by hard reboots while software issues are addressed by rigorous and continual online testing with the vehicle. We are now able to autonomously travel at speeds approaching 20mph over multiple hours along a typical RNDF without human intervention.

Since the race is largely about the humanization of driving we draw our comparisons to those of a human driver in

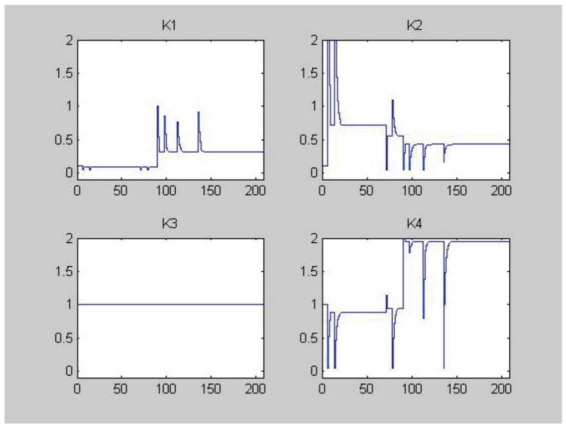


Figure 23. Parameter identification of human drivers in contexts similar to our path planner can help us assign weights to parameters that give smooth driving behaviors while satisfying the hard constraints of safety.

similar scenarios described in the urban challenge i.e. derive the characteristics from human behavior and analyze the difference with robotic behavior.

Thus, analysis and testing is now primarily focused on planning and navigation. For example a high confidence system should behave like a human driver for both observers and any riders in the vehicle. We are using emerging research to discover these parameters by observing human driver reactions to similar situations to those encountered in the urban challenge Figure 23.

VII. Conclusions

From the experience of the previous DARPA Grand Challenge, although only a few teams crossed the finish line, many of the other participants had unique and perhaps profound contributions in at least one respect, though they may lack the testbed, software, or theory to realize that contribution in the challenge.

Our team's approach from the beginning was to devise a system where individuals in our team could provide proof of concept of their ideas, and could then perform software development in parallel with other team members. Since integration of the runtime code is an integral part of our simulation procedure, our robotics middleware infrastructure enforces good systems practices from the beginning of implementation. And since our team and efforts are widely distributed, our ability to reintegrate our testbed vehicle in another country shows the flexibility of our basic design.

Finally, we believe that our novel approaches in high-precision sensing, real-time computer vision, and parallel planning will provide robustness, while simultaneously exploring research topics which will be of interest to the automotive and robotics sectors.

Acknowledgment

Each institution's administrative staff has provided support above and beyond the call of duty, given the harsh time constraints of the competition, the short timelines for most administrative emergencies, and the monumental efforts required to import the vehicle into the United States for participation in the Site Event and anticipated participation in the National Qualifying Event and Final Event. We are grateful for their services.

References

[1] R. Vidal, Y. Ma, S. Soatto, and S. Sastry, "Two-view multibody structure from motion," *International Journal of Computer Vision*, vol. 68, no. 1, 2006.
[2] R. Vidal and S. Sastry, "Optimal segmentation of dynamic scenes from two perspective views," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 281–286, 2003.
[3] R. Vidal, S. Soatto, Y. Ma, and S. Sastry, "Segmentation of dynamic scenes from the multibody fundamental matrix," in *Proceedings of ECCV Workshop on Visual Modeling of Dynamic Scenes*, 2002.
[4] R. Vidal and Y. Ma, "A unified algebraic approach to 2-d and 3-d motion segmentation," *Journal of Mathematical Imaging and Vision*, 2006. (In press).
[5] P. Torr, "Geometric motion segmentation and model selection," Phil. Trans. Royal Society of London, vol. 356, no. 1740, 1998.
[6] H. C. Longuet-Higgins, "The visual ambiguity of a moving plane," in *Proceedings of the Royal Society of London, Series B, Biological Sciences*, 1984.
[7] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis, "Graph clustering and minimum cut trees," *Internet Mathematics*, vol. 1, no. 4, pp. 355–378, 2004.
[8] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in

vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[9] Bryson & Ho, "Applied Optimal Control", Waltham, Mass., 1969.

[10] H.J. Kim, H.C. Shim, and S.S. Sastry, "Nonlinear Model Predictive Tracking Control for Rotorcraft-based Unmanned Aerial Vehicles", *Proc. of the American Control Conference*, 2002.

[11] *Optimization: Algorithms and Consistent Approximations*, Polak, Springer, 1997.

[12] *Model Predictive Control*, Camacho & Bordons, Springer, 2003.

[13] E. Lobaton and A. M. Bayen, "Modeling an optimization analysis of single flagellum bacterial motion", in *Proceedings of the American Control Conference*, 2007, (to appear).

[14] A. M. Bayen, R. Raffard, and C. J. Tomlin, "Adjoint-based control of a new Eulerian network model of Air Traffic Flow", *IEEE Transactions on Control Systems Technology, 14(5): 804-818, September 2006.*

[15] K. Amonlirdviman, N. A. Khare, D. R. P. Tree, W.-S. Chen, J. D. Axelrod, and C. J. Tomlin, "Mathematical Modeling of Planar Cell Polarity to Understand Domineering Nonautonomy", *Science 307 5708:423-426, 21 Jan 2005.*

[16] A. Jameson, "Aerodynamic design via control theory", *Journal of Scientific Computing,* 3(3): 233-260, 1988.

[17] J.S Albus, 4D/RCS: A reference model architecture for intelligent unmanned ground vehicles, In *SPIE 16th Annual International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, Orlando, FL, USA, 2002

[18] http://zeroc.com/, 2007

[19] http://orca-robotics.sourceforge.net/, 2007

[20] A. Elfes, Occupancy Grids: A Stochastic Spatial Representation for Active Robot Perception, Proc. 6th Conference on Uncertainty in AI, 1989

[21] S. Thrun et al., Stanley: The Robot that Won the DARPA Grand Challenge, Journal of Field Robotics, 23, p.661-692, 2006

[22] A. Ferencz, E. Learned-Miller, and J. Malik, Learning Hyper-Features for Visual Identification, Neural Information Processing Systems (NIPS), 2004

[23] Andras Ferencz, Erik G. Learned-Miller, and Jitendra Malik, Building a Classification Cascade for Visual Identification from One Example, ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, 2005

[24] C. P. Papageorgiou, and T. Poggio, A Trainable Object Detection System: Car Detection in Static Images, 1999

[25] http://opencvlibrary.sourceforge.net/, 2007

[26] http://www.ghostriderrobot.com/hdlidar/specifications.html, 2007

[27] R. Stettner, H. Bailey, and S. Silverman, Three dimensional Flash Ladar Focal Planes and Time Dependent Imaging, 2006

[28] http://www.advancedscientificconcepts.com/camera1.html,

[29] J. Au, Z. N. Li, and M. S. Drew, Object Segmentation and Tracking Using Video Locales, ICPR, 2002

[30] A. Yilmaz, O. Javed, and M. Shah, Object Tracking: A Survey, ACM computing surveys, 38, p., 2006

[31] M. Bertozzi and A. Broggi, GOLD: A parallel real-time stereo vision system for generic obstacle and lane detection, IEEE Trans. on Image Processing, 7, p.62-81, 1998

[32] P.E. Hart, N.J. Nilsson, and B. Raphael, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics SSC4, 2, p.100-107, 1968

[33] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh, Learning to act using real-time dynamic programming, Artificial Intelligence, 72, p.81-138, 1995

[34] K. Nagel and M. Schreckenberg, A cellular automaton model for freeway traffi, Journal de Physique I, 2, p.2221-2229, 1992

[35] R. Sutton and A. Barto, Reinforcement Learning: An Introduction, MIT Pres, 1998

[36] A. L. Zobrist, A New Hashing Method with Applications for Game Playing, ICCA Journal, 13, p.69-73, 1990

[37] Christopher J. C. H. Watkins and Peter Dayan, Q-Learning, Machine Learning, 8, p.279-292, 1992

[38] K. Nagel, D.E. Wolf, P. Wagner, and P. Simon, Two-lane traffic rules for cellular automata: Asystematic approach, arXiv:cond-mat/9712196, 58, p.1425-1437, 1998

[39] J.C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, 1991

[40] S.M. LaValle, Planning Algorithms, Cambridge University Press, 2006

[41] H. Choset et al., Principles of Robot Motion: Theory, Algorithms, and Implementations, MIT Press, 2005

[42] James Bruce and Manuela Veloso, Real-Time Randomized Path Planning for Robot Navigation, IROS, 2002

[43] A.M. Ladd and L.E. Kavraki, Fast Tree-based Exploration of State Space for Robots with Dynamics, Alrogithmic Foundations of Robotics VI, Springer STAR, 17, p.297, 2005

[44] http://player-stage.sourceforge.net/