REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
08-02-2017	Final Report	1-Aug-2012 - 30-Jun-2015

4. TITLE AND SUBTITLE
Final Report: A Scaled Automotive Platform for Validation and Testing of Perception and Control Algorithms for Unmanned Ground Vehicles Operating Under Extreme Driving Conditions

5a. CONTRACT NUMBER
W911NF-12-1-0377

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
611103

6. AUTHORS
Panagiotis Tsiotras

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
Georgia Tech Research Corporation
505 Tenth Street NW
Atlanta, GA 30332 -0420

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
61461-NS-RIP.1

12. DISTRIBUTION AVAILIBILITY STATEMENT
2 Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
This report summarizes efforts to create a new high-speed scaled autonomous vehicle and accompanying outdoor test facility at Georgia Tech via DURIP Award W911NF-12-1-0377 to test and validate newly developed perception and control algorithms for autonomous unmanned ground robotic vehicles (UGVs) operating in uncertain and unstructured environments, and under extreme driving conditions (i.e., off-road, at high speed, while skidding/slipping, etc). Two vehicles were constructed, each approximately 1 m long, 0.6 m wide, 0.4 m high, weighing 20.5 kg, with a top speed of 90 kph. The platform can drive itself fully autonomously using only on-board

15. SUBJECT TERMS
high-speed scaled autonomous vehicle, experiments, test track, hardware, mobile platform, UGV

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Panagiotis Tsiotras
UU	UU	UU	UU		19b. TELEPHONE NUMBER 404-894-9526

RPPR

as of 02-Feb-2023

Agency Code:

Proposal Number: **Agreement Number:**

Organization:
Address: , ,
Country:
DUNS Number: EIN:
Report Date: Date Received:
for Period Beginning and Ending
Title:
Begin Performance Period: **End Performance Period:**
Report Term: -
Submitted By: Email:
Phone:

Distribution Statement: -

STEM Degrees: **STEM Participants:**

Major Goals:

Accomplishments:

Training Opportunities:

Results Dissemination:

Plans Next Period:

Honors and Awards:

Protocol Activity Status:

Technology Transfer:

I certify that the information in the report is complete and accurate:
Signature:
Signature Date:

A Scaled Automotive Platform for Validation and Testing of Perception and Control Algorithms for Unmanned Ground Vehicles Operating Under Extreme Driving Conditions

by

Panagiotis Tsiotras
Principal Investigator
School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0150

FINAL REPORT

DURIP W911NF-12-1-0377

Submitted to

Army Research Office
Research Triangle Park, NC 27709-2211

October 2015

School of Aerospace Engineering
Atlanta, Georgia 30332-0150 U.S.A.
PHONE 404-894-9526
FAX 404-894-2760

1 Summary

This report summarizes efforts to create a new high-speed scaled autonomous vehicle and accompanying outdoor test facility at Georgia Tech via DURIP Award W911NF-12-1-0377 to test and validate newly developed perception and control algorithms for autonomous unmanned ground robotic vehicles (UGVs) operating in uncertain and unstructured environments, and under extreme driving conditions (i.e., off-road, at high speed, while skidding/slipping, etc).

Two vehicles were constructed, each approximately 1 m long, 0.6 m wide, 0.4 m high, weighing 20.5 kg, with a top speed of 90 kph. The platform can drive itself fully autonomously using only on-board sensing, computing, and power. This hardware complements the theoretical work of the PI and his co-workers in the area of integrated, neuro-inspired perception and control of autonomous, aggressively driven vehicles, developed under MURI ARO award "Neuro-Inspired Adaptive Perception and Control for Agile Mobility of Autonomous Vehicles in Uncertain and Hostile Environments" (ARO award no. FA9550-04-1-0135). This research will enable radically new capabilities of autonomous and semi-autonomous ground vehicles in the battlefield. It will enhance manyfold the maneuverability and mission impact of these vehicles by allowing them to move at high speed, over rough terrain and while operating in hostile environments.

While smaller than many other research-focused autonomous ground vehicles, the platform offers a cost-effective, fast, agile, and safe alternative to operating full-sized autonomous vehicles while retaining realistic vehicle dynamics that the smaller toy platforms lack. The sensor package, hardware, and computing capabilities offer a large performance improvement over traditional scaled autonomous vehicles. Several graduate and undergraduate students use the robots and test site for high-speed control, vision, and autonomy research. Although the research focuses on robotic ground vehicles, nonetheless, the challenges that are associated with this specific application (i.e., limited computational resources, limited on-board power, limited time to reach and execute decisions under an uncertain and rapidly changing environment) will benefit tremendously all unmanned, autonomous robotic systems (aerial, marine, underwater), which are also required to operate under similar constraints.

(a) Autonomous driving at test track (b) Assembled robot

Figure 1: Scaled autonomous platform

In this report we describe the main specifications of the platform (hardware and associated software) and we outline the system identification procedures, along with examples of two control designs implemented so far to allow completely autonomous operation. A movie of the vehicle operating a high-speed over rough terrain can be found at http://www.ae.gatech.edu/labs/dcsl/movies/full_track_2.avi.

2 Major Equipment Purchased

The list of major equipment purchased is given in Table 1.

Table 1: Major equipment purchased

Date Acquired	Description	Cost
5/29/2013	Test track construction materials	$3,143
10/13/2014	9 degree of freedom IMUs	$7,020
11/26/2014	Velodyne VLP-16 3D LIDARs	$15,998
Various	Machine vision cameras, lenses, cabling	$6,370
Various	GPS base station, on-robot GPS Units, antennas, and cabling	$17,686
Various	Robot chassis and on-board computing	$19,787
Various	Materials and supplies	$29,997
	Total	$100,000

3 Fully Autonomous Systems with Real-Time Execution Guarantees

3.1 Current State of the Art and Challenges

Reducing the risk to human lives and ensuring mission success, while operating in hazardous or hostile environments, has led to the development of unmanned, autonomous and semi-autonomous vehicles for many military applications. At the same time, most robotic unmanned ground vehicles (UGVs) currently deployed in the field operate at low-to-moderate speeds and have low-to-moderate maneuverability, thus making them vulnerable in the battlefield. Increasing the speed and agility of these vehicles will have enormous benefits, both in terms of extending the range and type of the missions currently undertaken by these vehicles, as well as in terms of increasing the success rate of these missions.

The completely autonomous operation of robotic wheeled land vehicles remains a challenging problem, despite the impressive levels of perception and autonomous driving demonstrated in the recent DARPA Grand and Urban Challenge competitions [1, 2]. Especially challenging is operating such vehicles at high speed in open terrain, over loose surfaces, and under abnormal driving conditions, such as severe under/oversteer, skidding, etc. This is owing to the short time scales required for planning and (re)action, compounded by the nonlinear and uncertain vehicle and terrain dynamics dominating these regimes. Furthermore, the problem of estimating the vehicle state within the environment (perception) cannot be tackled using the current practice of building detailed 3D maps of the environment, based on either LIDAR or EO (electro-optical) input alone, since this is computationally infeasible with current technology at the time scales required for this problem.

The major challenges one is faced when developing autonomous intelligent vehicles for operation in open terrain at high-speed, possibly in unfriendly territory or hostile environments include, among other things, the following:

Nonlinearity of Vehicle Dynamics and Uncertainty in Surface/Vehicle Interface: Controlling a vehicle close to or at its handling limits, during abnormal driving conditions (e.g., severe understeer or oversteer, while skidding, over rough terrain) is a notoriously difficult problem. Nonlinear effects dominate the dynamics, tire friction is close to (or exceeds) the adhesion limits, and the vehicle is close to instability. Common linear friction models are completely inadequate in these regimes.

Coupling Between Sensing/Perception and Motion/Control: Vehicle motion is coupled with data acquisition and sensing. Therefore, control strategies based on naive separation principle arguments are very

likely to fail. Large and rapid motions of the vehicle may even lead to divergence of traditional (i.e., EKF-based) estimators, not to mention the difficulties stemming from the paucity of inertial state information (e.g., GPS measurements).

Limited On-board Computational Resources: Operation of truly autonomous ground vehicles requires both trajectory design (planning) and trajectory tracking (control) tasks to be completely automated. Given the short response time scales of these vehicles, these are challenging tasks using existing route optimizers. This is especially true for small and agile UGVs, which may not have the on-board computational capabilities (CPU and memory) to implement some of the sophisticated perception and path planning algorithms proposed in the literature thus far.

3.2 A New Paradigm for Perception and Control for Autonomous Systems

Our current research aims at remedying the previous shortfalls by using insights from the perception and action mechanisms of human expert drivers. By encapsulating the cognitive and reflexive planning layers of a human expert driver, we make extensive use of prior information, both at the sensing and execution levels. Human cognitive models provide strong evidence that the manner by which human (especially expert) drivers perceive and (re)act to environmental stimuli and process information, is quite different from the way most current perception techniques deal with this problem. For instance, because of experience, and prior familiarity with typical road conditions and geometry (this is common for instance in rally driving competitions), the driver has already a preconceived notion of "what is coming ahead." (S)he certainly does not "reconstruct" the whole environment at every instant of time, and his/her approach to driving has a significant anticipatory (non-causal) component.

We explore hierarchical and multi-resolution approaches to vision and processing, inspired by the human visual cortex. Departing from the standard paradigm of perception/sensing then control, we promote instead perception/sensing-for-control by leveraging attention-focused, adaptive perception algorithms that operate on actionable data in a timely manner. This will eventually enable real-time processing of the collected information by the on-board sensors and the subsequent fast reaction by the vehicle. We use the element of surprise and context priors as a means to focus attention on the data and events (temporal and spatial) that are most relevant to the task(s) at hand.

3.3 An Open Platform to Test and Validate Perception and Control Algorithms

Part of the difficulty in studying this problem is the large investment in vehicle hardware, track infrastructure, and safety procedures needed to test high-speed algorithms on a full size vehicle. Prior to this work, the only options for testing algorithms were to use a full-scale vehicle or use a small-scale remotely controlled vehicle. While full scale vehicles exhibit dynamics that are of interest and can carry on-board computation, safety and cost make it very difficult to test new algorithms. One or more full time engineers is generally needed to maintain the vehicle, large test facilities are needed, and ensuring vehicle and test personnel safety requires significant thought and planning. While small scale vehicles are much easier and less expensive to work with, they exhibit significantly different dynamics and cannot carry significant computation on-board.

This report summarizes the major equipment purchased to construct two scaled automobile platforms for research and education in autonomous vehicle dynamics, sensing, control, path-planning and perception. Building and testing a scaled model of an automobile bridges the gap between inaccurate software simulations and costly full-scale real vehicle experiments. Development and evaluation of new perception and control algorithms cannot be based solely on simulations, which may fail to capture critical aspects of the real-world. On the other hand, experimentation with full-scale vehicles faces several cost and risk issues. The risk issue is particularly important in our case, as the autonomous vehicles are driven at high-speeds and often at the limits of their handling capacity. Furthermore, when working with full-size vehicles interfering

with factory-installed sensors and actuators poses a severe challenge, while, even for a trial run, a racing track is necessary. Therefore, several researchers have turned their attention to scaled automotive models for their potential to provide valuable results with less cost and effort compared to full-scale cars [3, 4]. To the best of our knowledge, the ability to rapidly deploy advanced control and perception algorithms on a scaled platform maintained and operated by a small team of students is unique.

4 Chassis

The chassis is derived from a 1:5 scale HPI Baja 5SC radio controlled trophy truck model. Fig. 2 shows the assembled chassis with all modifications installed and the plastic protective outer body removed. Some stock components were replaced with off-the-shelf, third-party hardware to accommodate the weight added by the sensor and computing payload. The total weight of the assembled chassis is 13 kg.

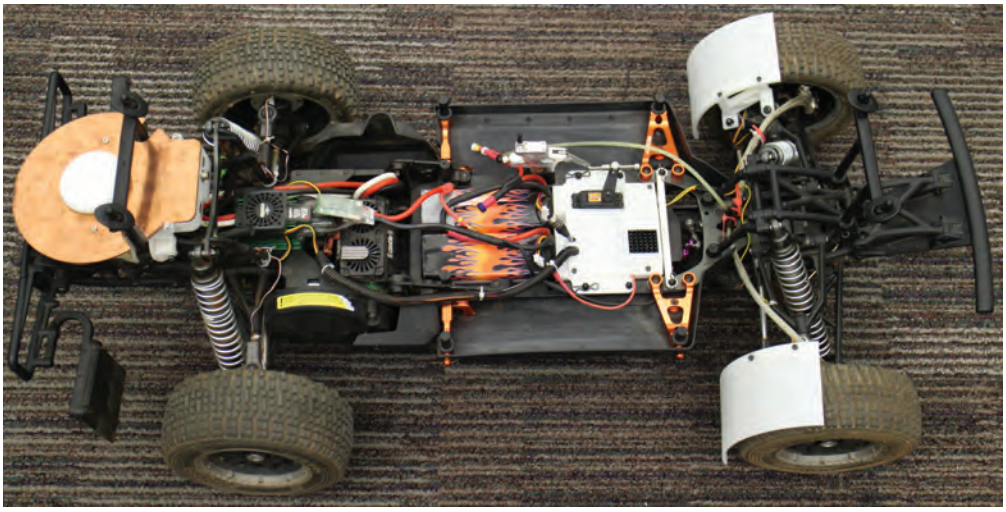


Figure 2: Fully assembled chassis

The most significant change to the chassis is the replacement of the stock 3hp 26cc 2-stroke gasoline engine with the 10hp electric motor from the Castle Creations Baja Conversion Kit. Compared to the stock engine, the electric motor is more than 3 times as powerful, more reliable, can electronically brake the rear wheels, is much more compact, generates little heat, produces no oil residue, and requires no maintenance or cleaning between runs. The electric motor has a much faster response to changes in throttle commands across all operating conditions. The motor and chassis electronics are powered by 2, 4-cell 18.4V 6500mAh Lithium-Polymer (LiPo) batteries connected in series for a total voltage of 36.8V. The batteries provide the same 45 minutes of continuous usage as a full tank of fuel, but take up less space and weigh less.

The chassis requires one servo to operate the steering and one for the front brakes. We chose the newer 7.4V digital hobby servos which give more precise control at a faster rate, higher torque output because of increased power consumption, faster response time, and a reduced dead band compared to traditional 6.0V analog servos. The steering servo we use, which is the most powerful 1/5 scale steering servo we found, is the Savox SV-0235mg with a torque output of 486.1 oz-in and an unloaded rotation speed of 0.15 s/60 deg. The brake servo is the Savox SV-1271SG that has a torque output of 347.2 oz-in and an unloaded rotation speed of 0.08 s/60 deg. The servos are powered by a Castle Creations Battery Eliminator Circuit (BEC) Pro, which has a maximum output current of 20A and can handle the high power draw of the heavy duty servos and high input voltage (38.6V) from the chassis batteries. A BEC is the hobby RC equivalent of a programmable voltage regulator with data logging capabilities. The brake servo actuates the master cylinder

of a Mecatech hydraulic front brake kit.

Parts of the chassis structure were upgraded to handle the increased size and load of the sensor and computing package. The stock injection-molded plastic steering linkage was replaced with billet aluminum parts to withstand the increased steering torques and weight on the linkage. The plastic side rail guards that we use as mount points for the compute box were replaced with billet aluminum parts to carry the weight of the compute box without bending. Axle extenders were installed to increase the track of the vehicle by 1 in. A wider track increases lateral stability and provides room to mount the front brakes. The stock suspension springs originally designed for the 28 lb chassis were replaced with stiffer springs of similar dimensions to reduce body roll and better absorb bumps on the fully assembled 45 lb chassis.

To determine the desired new spring constants, the suspension was analyzed as a second order spring mass damper system where the natural frequency of the system is directly related to the ratio of the spring constant to the system mass. We measured the spring constants of the stock front springs to be $k_f = 8.48\ lb/in$ and the two springs on one of the rear shocks combined to be $k_r = 11.17\ lb/in$. We calculated desired new spring constants given the ratios: $8.48/28 = k_f/45$ and $11.17/28 = k_r/45$, which gives a desired $k_f = 13.63\ lb/in$ and $k_r = 17.95$ lb/in. Custom springs are prohibitively expensive, so we installed off-the-shelf springs with the same overall dimensions as the stock configuration that have $k_f = 15$ lb/in and $k_r = 19.09$ lb/in. These modifications increased the natural frequency of the system, which in turn decreased the peak response time of the suspension, along with the settling time. The oil in the shocks was also chosen to be 40 W, which is the most viscous oil available from the manufacturer. This high weight oil increases the damping ratio of the system and decrease the overshoot of the underdamped system to help prevent scraping the chassis along the ground after jumps and reduce bouncing over bumps.

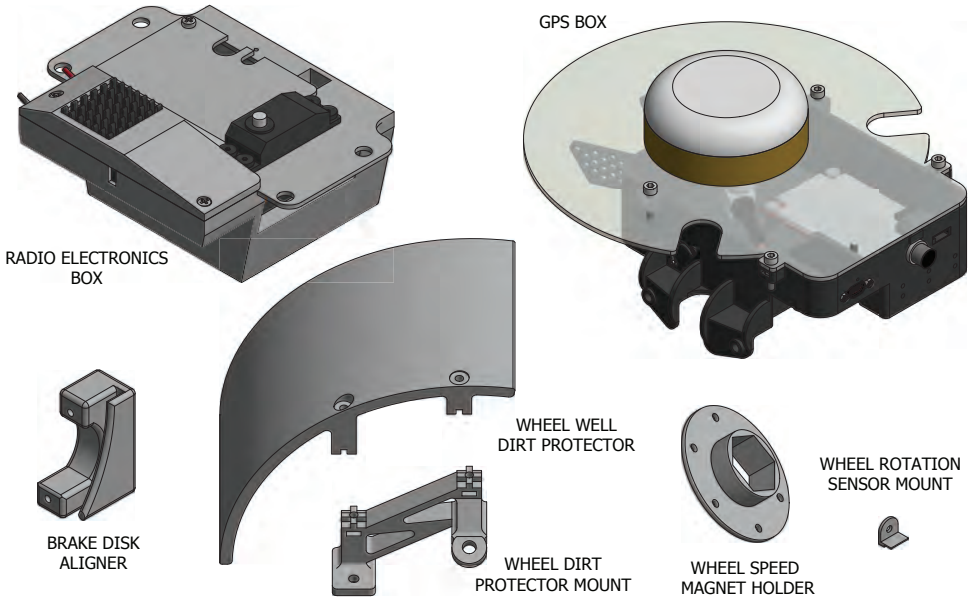
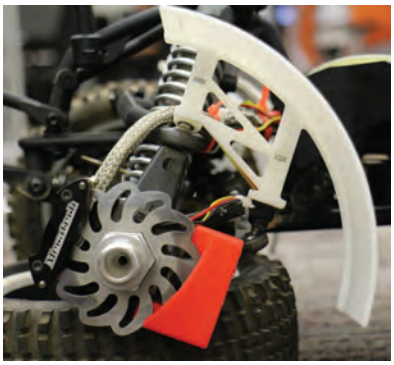


Figure 3: 3D printed chassis components

The standard controller for hobby RC cars is 2-channel to control throttle and steering. We chose a programmable 4-channel transmitter. The first 2 channels control the steering and throttle, respectively, and the remaining channels are used for the vehicle safety system, which is discussed in Section 6.1.

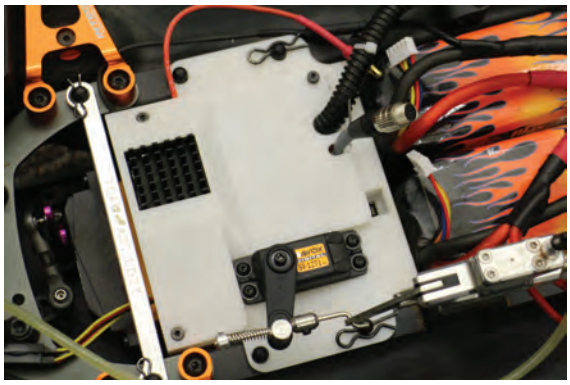
In addition to the third party components purchased from hobby suppliers, we designed and 3D-printed custom components out of ABS plastic. The 3D printed components on the chassis include a replacement electronics box, the GPS/IMU box, mounts for the back wheel rotation sensors and magnets, and alignment

(a) Springs setup, black stock and silver upgraded

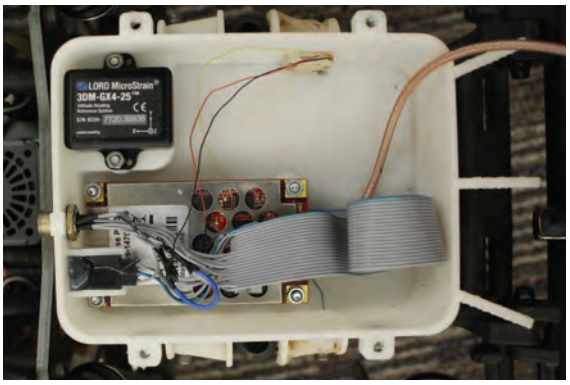


(b) Front wheel hub assembly

(c) Rear wheel hub assembly



(d) Chassis electronics box



(e) GPS and IMU box, lid removed

Figure 4: Upgrades and custom components

guides for the front brake disks. The electronics box replaces the stock one mounted in the front of the chassis hull, just behind the steering servo and linkage. Contained within the box are the stock radio receiver, USB servo controller, servo multiplexer, brake servo, servo capacitor, and run stop relay. Inside the GPS/IMU box is the GPS, IMU, a small fan, and the GPS antenna is mounted to the ground plane on the top of the box. Mounts for wheel rotation sensor magnets and for the Hall-effect wheel rotation sensors are also installed on the chassis. Dust covers for the front tires were printed to reduce the amount of dust kicked up into the compute box. The dust covers were added after the dust filters were repeatedly clogged by fine dust during long runs during dry track conditions. Excessive dust buildup greatly reduces airflow through the compute box and can lead to overheating and component failures, especially because ambient air temperatures during summer tests can reach 100 degrees F. The brake disk aligners were installed because the wheel rotation sensor magnets unbalance the disks, which catch on the brake pads causing the wheels to lock up.

5 On-board Computing

We designed a modular, reconfigurable on-board computing solution that uses standard consumer computer components and a common interface to the chassis. Computing hardware development outpaces advancements in almost all other components, so our solution can be upgraded as necessary and scaled as computing requirements evolve. Here, we present the two different computing solution: a high-performance, desktop-grade computer and a lightweight, low-power NVidia Jetson TK1-based computer. Because the compute

Table 2: Compute box comparison

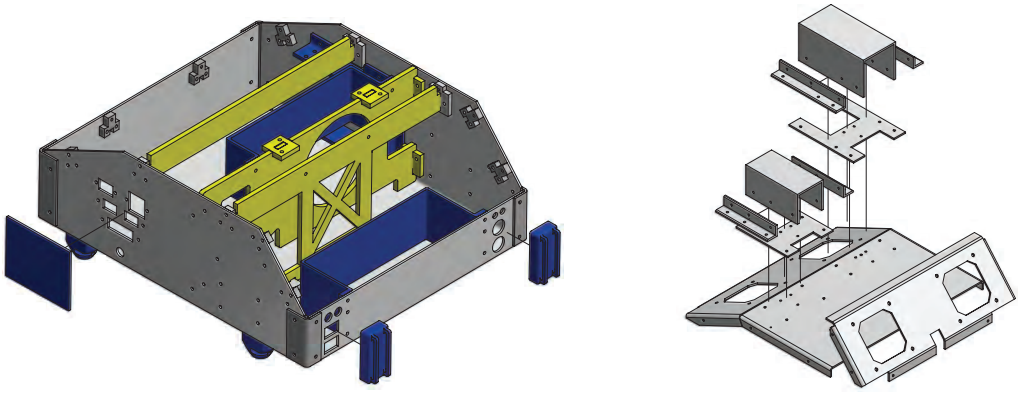
	Mini-ITX	Jetson
CPU	3.1GHz Intel quad-core i7	2.32GHz ARM quad-core Cortex-A15
System Memory	16GB DDR3 1600MHz	2GB DDR3L 933MHz
GPU Cores	640	192
GPU Clock Speed	1020MHz	852MHz
GPU Memory	2GB GDDR5, 5.4 Gbps	Shared with RAM
On-board Storage	-	16GB fast eMMC
Storage SSD	240GB and 1TB SATA3	240GB SATA3
Empty Weight	3.3kg	2.0kg
Full Weight	7.5kg	3.98kg
Max Power Consumption	168W	70W

boxes share the same mounting hardware and data connections, they are interchangeable between the chassis. This way, if one chassis or compute box is damaged or undergoing maintenance, we can simply swap out the non-functional part and continue testing. The software is compute box and chassis agnostic given a configuration script to detect the current chassis, compute box, and sensor package.

5.1 Mini-ITX Compute Box

A major design consideration for the platform is on-board computing performance. Modern control and perception algorithms are very CPU- and GPU-intensive. To maximize performance and reduce hardware development time, the compute box employs standard consumer components instead of specialized embedded hardware typical of scaled autonomous platforms. The compute box design provides a robust enclosure that fits on top of the chassis inside the stock Lexan body. The box is designed to withstand high speed crashes including collisions with fixed objects, rollovers, and crashes from jumps. The compute box exterior is fabricated from 3003 aluminum sheet cut with a water-jet and bent on a Magna-Bend. The interior structure is a combination of laser-cut acrylic and 3D printed ABS plastic parts. The enclosure is designed to withstand a 10g direct impact from any angle without damaging internal components. The 10g impact is larger than any impact experienced in the past, as measured by our IMU. We verified the box's impact tolerance with FEA of the CAD model before fabrication. The all-aluminum exterior provides excellent EMI containment so that the GPS receiver can operate in close proximity. Aluminum dust filters coupled with an inner foam membrane filter out environmental contaminants such as dust and rocks without creating openings for EMI to escape. The cameras and lenses, mounted on the top of the box, are protected by covers made from square 6061 aluminum tube.

The computer motherboard is an ASUS Z87I-Deluxe Mini-ITX motherboard and the CPU is a 3.1 GHZ Intel 65W Haswell Quad-core i7 processor. On-board RAM is 16GB of DDR3 1600MHz. One 240GB SSD is used as the primary system drive and one 1TB SSD is used for data logging. With all programs running, including both cameras at 60Hz, the 1GB of data is logged every 3 seconds, which fills the logging drive in 16 min. Images compose 98% of the logged data so the camera frame rate is normally set to less than 30Hz to extend logging time. GPU computing is performed on an NVidia GTX-750ti. The GTX-750ti is a half-length design that matches the Mini-ITX form factor. Relatively low 60W maximum power consumption keeps the entire power budget within the capabilities of our power supply. Dual 2.4GHz WiFi antennas are installed on the top of the box and connect to the integrated 802.11ac WiFi card. WiFi is used to remotely monitor high bandwidth, non-time critical data from the platform such as images and diagnostic information. A 900MHz XBee Pro provides a low-latency, low-bandwidth wireless communication channel. This connection carries a global heartbeat message and GPS RTK correction from the base



(a) Compute box assembly

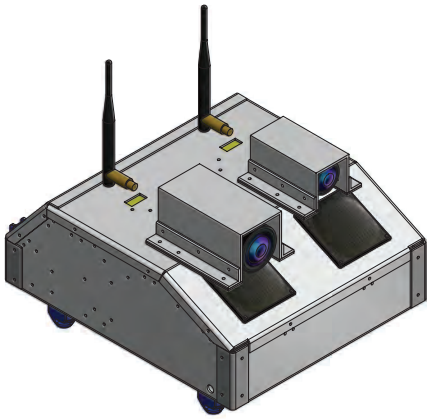
(b) Compute box lid assembly

Figure 5: Mini-ITX compute box CAD models, aluminum is gray, acrylic is yellow, 3D printed ABS is blue

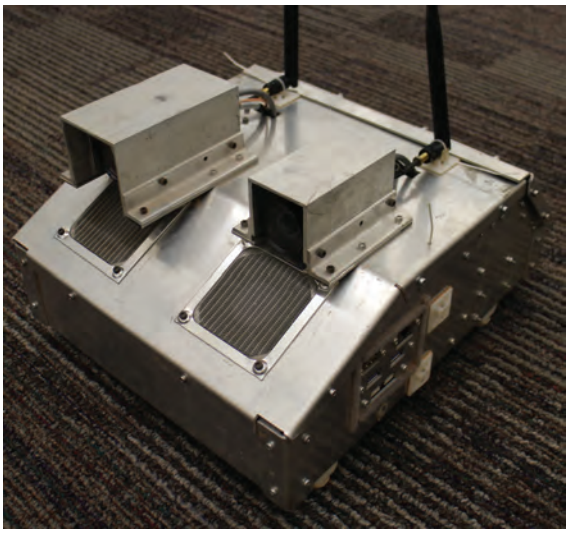
station. The base station XBee functions in broadcast mode, so all robots in communication range can use the same GPS correction information and be stopped simultaneously by the heartbeat. This is akin to the green/red/black flags used to convey information to all drivers simultaneously on a race course. Wheel rotation sensor data, synchronous camera triggering, and capture of the servo signals coming from the hobby RC transmitter/receiver is performed by a 16MHz 8-bit Arduino Mega 2560. The Mega has 128kB of flash memory, 8kB of SRAM, and 4kB of EEPROM memory. The data comes into the Arduino from the chassis via a an 8-pin cable with Hirose HR25 connections. A Mini-Box M4-ATX DC-DC power supply powers everything in the compute box. The M4-ATX is designed for automotive DC power systems and provides 250W of continuous power and 300W peak. A Cui DC-DC 3.3V isolated power supply powers the GPS and GPS antenna through a 3-wire servo cable leading to the GPS/IMU box. The compute battery installed is a 22.2V, 10Ah 6S LiPo battery with a total capacity of 222Wh. The battery is sized to last 1.5 hours at full load, which is twice as long as the chassis batteries because the robot is often stationary with the computer running during testing. Two Mini-Box Y-PWR Hot Swap boards are installed in parallel to allow laptop-like switching between wall and battery power without requiring a system shutdown. External DC power is connected through panel-mounted barrel plugs on the exterior of the compute box. The hot-swaps automatically route the higher voltage source through to the power supply. Internal battery voltage and computer temperature sensors are used to monitor system health. The total weight of the empty compute box is 3.3 kg, and 7.5kg with all components installed.

Inside the compute box there are 4 3D printed components: a battery holder, a SSD holder, a GPU holder, and a RAM holder. The battery holder tightly secures the internal compute battery to the inner back wall of the compute box. It is slightly undersized so that the battery press-fits into the mount and can be removed for charging and maintenance without removing any internal screws. The SSD holder is used to securely mount both SSDs to the side wall of the compute box. The GPU holder provides a vertical contact surface to hold the GPU against the central internal acrylic strut. The RAM holder presses over the 2 RAM DIMMs and screws down to the CPU fan mounts. The need for the RAM holder was discovered because the computer locked up whenever the robot experienced a large lateral acceleration during a roll-over or landing jumps. The behavior was reproduced in lab by pushing laterally on the RAM DIMMs. With the RAM holder installed, the behavior persists only in situations when the robot cannot continue to physically operate such as end-over-end flipping crashes.

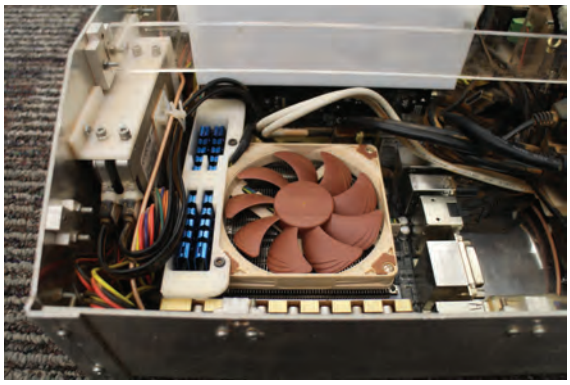
The compute box attached to the chassis with 4 3D printed mounts secured to the bottom of the compute box. The mounts fit over vertical posts on the chassis rail guards and are held in place with a cotter pin



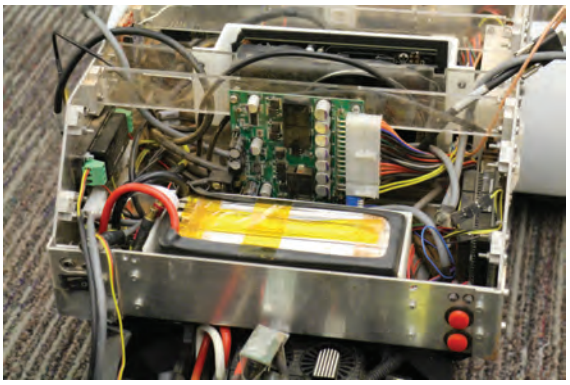
(a) Fully assembled CAD model



(b) Fully assembled



(c) Front view: motherboard, SSDs, and RAM holder



(d) Rear view: battery, power supplies, and Arduino

Figure 6: Mini-ITX compute box

though the mount and post. Special consideration was taken to design the mounts as break-away points for the compute box in the event of a catastrophic crash. The mounts are designed to break away before any of the aluminum compute box parts fail to protect the electronics from damage caused by the compute box warping. By applying lateral forces with FEA and the CAD models, we designed the failure point of the mounts to be at 8g of lateral force on the compute box compared to the 10g design load for the rest of the compute box. In practice, the compute box mounts break away during hard rollover crashes without damaging internal components. Button covers are mounted to the outside of the compute box to protect the buttons during a crash and keep dirt out.

5.2 Jetson Compute Box

The NVidia Jetson TK1-based compute box was designed as a light, low-power compute box compared to the Mini-ITX box. The board is built around a Tegra K1 System On a Chip (SOC) that has a 4-Plus-1 quad-core ARM Cortex A15 CPU, a 192-core Kepler GPU, 2GB RAM shared with the GPU, and 16GB eMMC memory. We installed an Intel 802.11ac WiFi card in the available half mini-PCIE slot, a 240GB SATA3 SSD for data logging, and a 4-port USB3.0 hub and 10-port USB2.0 hub to connect all of the peripherals. The Jetson compute box uses the same dual 2.4 GHz WiFi antennas and 900MHz XBee Pro as the Mini-ITX

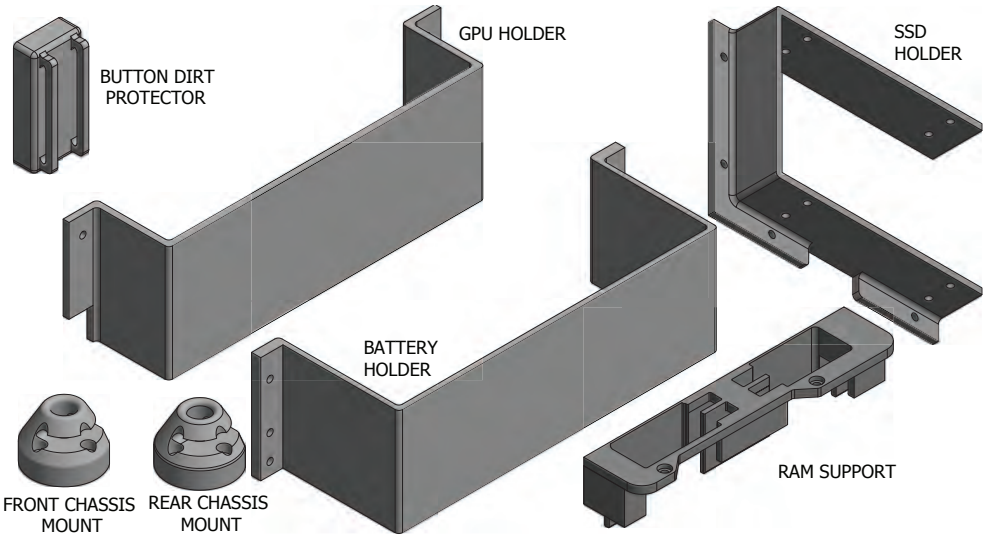
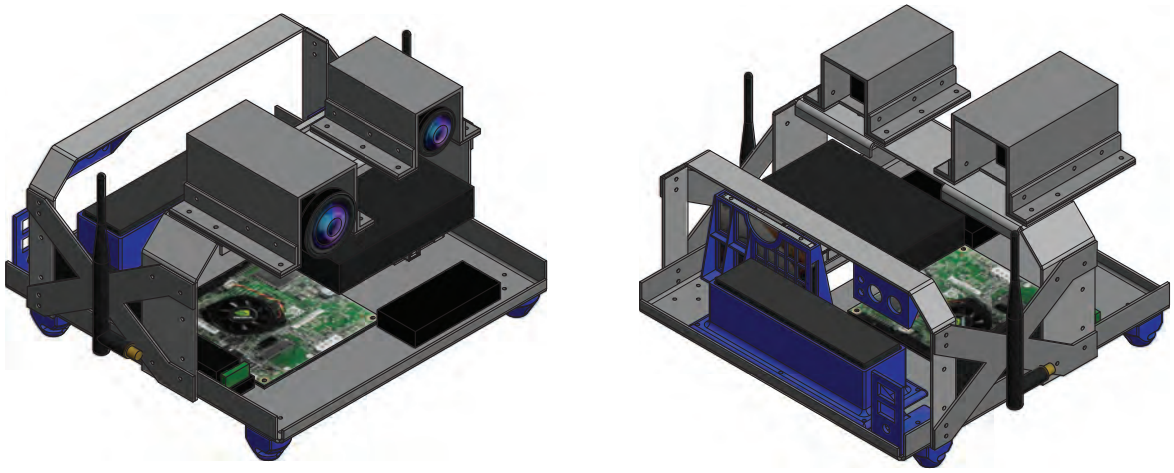


Figure 7: 3D printed Mini-ITX compute box components

Compute Box. Just like in the Mini-ITX box, an Arduino Mega 2560 and 8-pin Hirose HR25 able assembly handle the wheel rotation sensor data, synchronous camera triggering, and capture of the servo signals from the hobby RC transmitter/receiver. We use a Mini-Box DCDC-USB-200 power supply to power everything in the compute box except for the USB hubs, which are powered by a 5V Pololu step-down voltage regulator. The DCDC-USB-200 power supply provides a single user-programmable output voltage and a continuous power output of 150W and 180W peak. A Cui DC-DC 3.3V isolated power supply provides power for the GPS and GPS antenna through a 3-wire servo cable leading to the GPS/IMU box. The battery installed is 22.2V, 4Ah 6S LiPo battery with a total capacity of 88.8Wh.



(a) Front view

(b) Back view

Figure 8: Jetson compute box CAD, aluminum parts are gray, 3D printed ABS parts are blue

The battery is sized to run for 1.5 hours at full system load, just like the other compute box. One Mini-Box Y-PWR Hot Swap board is installed to allow laptop-like switching between wall and battery power without requiring a system shutdown. External DC power can be connected through a panel-mounted barrel

plug on the exterior of the compute box. The total weight of the empty compute box is 2.0kg and 3.98kg with all components installed.

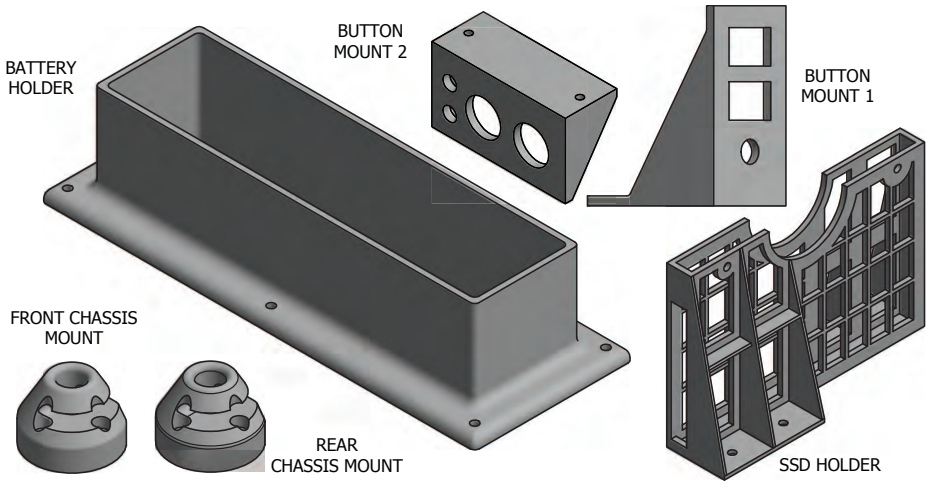


Figure 9: 3D printed Jetson compute box components

The camera mounts and covers on the Jetson box are positioned at the same 3D positions relative to the chassis as the Mini-ITX compute box so that the cameras in both compute boxes have the same perspective and are well-protected in the case of crashes. The cameras are mounted on an aluminum roll bar with flanges added along the length of the bar to increase the rigidity. Flanges were also added to the base plate to prevent deflection and oscillation of the box and attached components. The design of the box was optimized using FEA to be able to protect components from a worst case impact acceleration of 10g, however the roll cage components have a higher chance to permanently deform under high loads than the solid Mini-ITX compute box. This may necessitate occasional replacement of the bars or cross braces in the most severe crashes, however the frequency of these incidents has been low and the weight advantages of the Jetson box are significant enough to justify the occasional component replacement.

The bottom of the Jetson compute box has a smaller footprint than the Mini-ITX box, but uses the same 4 3D printed mounts that secure the compute box to the chassis. The power buttons, switches, and external power jack that are panel-mounted to 3D printed holders. Additionally, the battery holder and SSD holder are 3D printed and secured to the compute box with screws.

6 Sensors and Communication Package

A Hall-effect wheel speed sensor is installed on each wheel hub, along with magnets on the front wheel brake disks and back wheel 3D printed magnet mount to trigger the sensors. The chosen sensor is a Hallogic OH090U unipolar switch with a 90 Gauss trip and 65G release, which is the lowest tripping sensor in the 3-SIP form factor from Digikey, which can be directly wired into the system. The chosen magnets are N52 grade 0.3175 cm diameter, 0.1588 cm thick and produce a magnetic field that can trip this specific sensor from up to 0.58 cm away. Larger magnets could be used to increase the maximum tripping distance but the chosen setup works reliably with little maintenance and fits much better in the wheel hub assembly. Hardware timers in the Arduino are used to accurately measure the time between successive magnets. Timing information is sent from the Arduino to the computer at 70 Hz. The computer translates the inter-magnet time to a rotational velocity. The Arduino also relays PWM control signals from the remote RC transmitter at 50 Hz to the computer. To trigger the cameras, the Arduino generates an output signal that is fed to the

discrete GPIO input of each Point Grey camera.

A 900 MHz XBee is used for an additional wireless communication channel with the ground station. This radio has a RF data rate of 156 Kbps, a range of 6 miles, a wire antenna, and 50 mW of transmit power. The XBee carries low bandwidth, time sensitive signals such as GPS RTK corrections for the on-board GPS and a heartbeat from the remote run-stop. It communicates with the computer and ROS system through a USB port.

A Lord Microstrain 3DM-GX4-25 IMU transmits raw acceleration and angular rate data at 200Hz (max 1kHz) and fused orientation estimates at 200Hz (max 500Hz). The accelerometer can measure $\pm$16g accelerations with 0.04mg bias instability and 100μg/$sqrt\mathrm{Hz}$ noise density. The gyroscopes can measure $\pm$900 deg/s rotation rates with 10 deg/hour bias instability and $0.005\ \mathrm{deg}/\mathrm{s}/\sqrt{\mathrm{Hz}}$. The biases are temperature compensated over their entire operating range of -40C to 85C. The IMU is housed in a rugged enclosure and all data passes through a micro-DB9 connection. Two cameras, mounted at the top of the compute box facing forward, are used to observe the environment. Both are Point Grey Flea3 FL3-U3-13E4C-C color cameras with a global shutter, 60Hz frame rate, and USB 3.0 data connections. All cameras use fixed focal length lenses from Edmonds Optics. One is equipped with a 185 degree FOV fish-eye lens and the other with a 70 degree field of view (FOV) lens. Additionally, an Hemisphere P303 RTK corrected GPS receiver provides absolute position at 20Hz, accurate to approximately 2 cm under ideal conditions. The RTK corrections come over XBee from a stationary Hemisphere R320 GPS base station setup at the test site.

The on-board LIDAR sensor, the Velodyne VLP-16, spins an internal vertical array of 16 sensors at 10 Hz to generate a point cloud composed of 300,000 points every second. The sensor has a +/- 15 deg vertical field of view and a 360 degree horizontal field of view and a range of 100m with +/- 2cm accuracy. In addition, it has internal MEMS accelerometers and gyros for six-axis motion correction. Despite the Velodyne's robust enclosure and unique design among 3D LIDARs that fully encapsulates all moving parts, the Velodyne must be mounted high on the vehicle and exposed to damage from crashes and rollovers for a usable field of view, it is only used during slow-speed testing when the chance of a roll-over or crash is low.

6.1 Safety System

The vehicle has a 3-layer safety system to remotely stop the vehicle: software-based run stop, the ability to remotely assume manual control at any time, and a wireless live man switch to break the PWM signal going to the speed controller using a Pololu PWM-operated relay in the electronics box. The configuration enables seamless, remote switching between manual and autonomous modes. The behavior is accomplished by using a switch on the RC transmitter to control the signal select line on the electronic box's servo multiplexer The servo signal multiplexer controls which of the two sets of PWM signals, the human- or computer-generated signals, are sent to the servos and speed controller.

6.2 Time Synchronization

Another critical issue that must be handled for the platform to be useful is time synchronization. Time synchronization is performed on all of the primary computing and sensing components. System time is synchronized to GPS time using both the NMEA messages coming from the GPS as well as the PPS synchronization signal provided by the GPS. The IMU, LIDAR, and computer automatically synchronize their times with these signals, and the Arduino uses the PPS signal to synchronously trigger the cameras with GPS time.

Many of the sensors have their own internal clocks that need to be synchronized to system time so that data can be accurately time stamped. The system uses the GPS pulse per second (PPS) signal, which is a timed pulse sent out at the beginning of each second according to its internal estimate of GPS time on a

dedicated pin, and NMEA 0183 timing information messages for synchronization. The PPS signal is routed to the computer, IMU, and LIDAR, which all automatically synchronize their clocks to the incoming signal. The cameras images are time stamped with system time when they are received by the computer.

The system clock is synchronized to GPS time with the PPS signal and NMEA 0183 messages from the GPS that are routed to one of the USB ports on the computer via a serial to USB converter. The open source program GPSD connects to this port and acts as a time server for Chrony, which controls the system clock to within a few milliseconds of GPS time.

The Microstrain IMU provides a dedicated pin for a PPS input. In addition to the PPS signal, it requires the current GPS second (GPS time is given in seconds since Jan 6, 1980) of the current PPS pulse. This value can be derived from the system clock, which is also synchronized to GPS time. The IMU uses these 2 pieces of information to synchronize its own clock and time stamp each measurement with an accuracy of significantly less than one millisecond to system time.

The cameras provide an external trigger interface. However, this interface requires that each individual frame is triggered. The Arduino is used to provide the cameras with a triggering pulse at the specified frame rate. Each time a PPS pulse comes from the GPS, a train of evenly spaced pulses at the rate specified in the ROS system is sent to the cameras. This way, the exact time when each frame was taken is known.

7 System ID

Many controllers rely on analytic models that require specific vehicle parameters to be calculated. Some of the commonly used parameters were determined for the chassis and Mini-ITX compute box.

Table 3: Experimentally calculated MOI.

Variable	mass	T	b	d	$\text{MOI}_{\text{basic}}$	MOI_{exp}
$I_{cm,\phi}$	20.175	3.434	0.114	2.518	0.473	0.347
$I_{cm,\theta}$	20.175	2.500	0.292	2.477	1.358	1.068
$I_{cm,\psi}$	20.175	2.221	0.343	2.240	1.474	1.155
I_f	0.490	4.316	0.038	1.568	0.005	0.004
I_b	0.655	5.144	0.038	1.578	0.006	0.010

The most accurate way to calculate moments of inertia (MOI) is with a full model in CAD software. Methods exist to compute MOI by precomputation [5, 6], or online estimation [7] in cases where the full model is unknown or changing. No complete CAD model of our robot exists as it is a combination of hobby and custom components. Custom calibration rigs are time consuming, expensive, and difficult to build, so we compute the necessary MOI and center of mass experimentally using relatively simple methods. An extensive survey of popular methods for experimentally determining MOI is presented in [8]. We use a simplified version of the bifilar pendulum method from this paper. Shown in Fig. 10, we attach two fixed, parallel cables equidistant from the center of mass of the body to isolate the desired axis. The body is rotated by a small angle around the desired axis, then released and let freely oscillate. Given the parameters of the test rig and known parameters of the robot, the period of a free oscillation after an excitation determines the MOI about that axis. The equation for the moment of inertia about a single axis using the bifilar pendulum method from [8], with variables renamed for consistency, is:

$$I = m\frac{gR_1R_2}{4\pi^2 h}T^2 \tag{1}$$

where T is the oscillation period, h is the distance of the calibration object from the support taking into account non-vertical support wires, and R_1 and R_2 are the distances from the center of mass to the support wire attachment. The change in height from the mounting location as the vehicle is rotated is assumed zero when the angle of rotation is small. Our setup is simplified by using parallel support wires, $R_1 = R_2 = b$, which results in:

$$h = \sqrt{d^2 - (R_1 - R_2)^2} = d \tag{2}$$

and finally:

$$I = m\frac{gb^2}{4\pi^2 d}T^2 \tag{3}$$

Note that time units are seconds and distances are in meters. Table 3 lists the experimentally determined MOI and the associate testing data. $\text{MOI}_{\text{basic}}$ is calculated using constant density geometric volumes with same dimensions as robot. MOI_{exp} is experimentally calculated using Equation 3 and the measured parameters. The oscillation period T is calculated by averaging 20 oscillation periods using the bifilar pendulum method. Table 4 lists all other experimentally determined parameters.

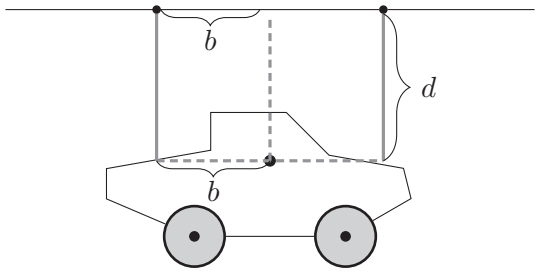


Figure 10: Bifilar pendulum setup for experimental determination of mass moment of inertia around a single axis. Example setup for calibrating chassis yaw axis MOI $I_{cm,\psi}$.

Table 4: Measured robot parameters

Parameter	Value	Units
Chassis mass	13.0	kg
Mini-ITX compute box mass	7.50	kg
Jetson compute box mass	3.95	kg
Front wheel mass	0.49	kg
Back wheel mass	0.66	kg
Overall length	0.90	m
Overall width	0.46	m
Overall height	0.32	m
Wheelbase	0.57	m
Front Track	0.395	m
Rear Track	0.405	m

8 Software Interface

All computers in the system use Ubuntu Desktop 14.04 LTS, which is the current long term support edition that will continue to be maintained until April 2019. The decision was made to go with the most well-supported Ubuntu release, as opposed to using a real-time kernel or a headless version (no GUI) for ease of use, wide availability of pre-compiled packages, and lack of custom configuration requirements out of the box. The custom software on the robot is developed using the Indigo Igloo release of the Robot Operating System (ROS) [9], which is also be supported until April 2019. Over the past few years ROS has become a widely adopted middle-ware in the robotics community that provides many tools useful for rapid project development and testing such as process encapsulation, inter-process publishing via structured messages, time-tagged logging of all data, and real-time data visualization and debugging capabilities. We developed ROS interface programs for each sensor without a publicly available interface. As ROS continues to grow in popularity, the barrier to entry for working with the AutoRally platform will decrease as new robotics students will arrive with at least a passing understanding of ROS.

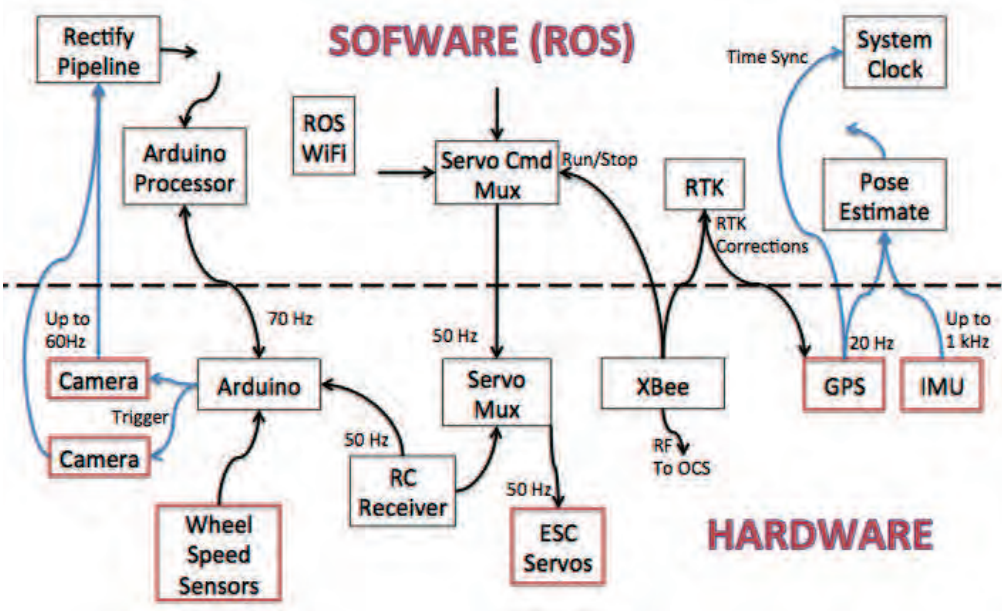
8.1 Operator Control Station

The Operator Control Station (OCS) is a tab-based graphical user interface (GUI) built using QT4 that provides real-time diagnostic information, control debugging capabilities, and a software run stop to a remote operator. Information displayed includes wheel speed data, compressed images from the on-board cameras, and all diagnostic messages on the special ROS topic `/diagnostics` which contain detailed information about the health of running nodes. The OCS colors each diagnostic message in the diagnostic tab according to the status flag in the message: green for OK, yellow for warning, and red for error. Another feature of the OCS is a button for the user to disable motion through the software system that is visible regardless of the tab selected. That signal is transmitted over WiFi to the robot. The robot can be controlled for debugging purposes through the control tab using sliders for the steering, throttle, and front. The OCS runs on a 13.3 inch Toshiba Portegé laptop with a dual-core Intel i5 processor with 4GB of RAM and a 320GB hard drive. There are no specific performance requirements other than being portable and able to run the OCS GUI because all autonomy-related computations happen on the robot. A Lenovo w520 laptop with a quad-core Intel i7, 8GB RAM, and a 240GB SSD serves as the second OCS laptop when both platforms run simultaneously. The Lenovo laptop was originally the on-board computer for the first generation gas-powered platform that has since been upgraded to all-electric and a custom compute box. If needed, a PlayStation style USB game pad can be plugged into the OCS laptop to remotely control the robot through the software system using a ROS standard joystick control node.

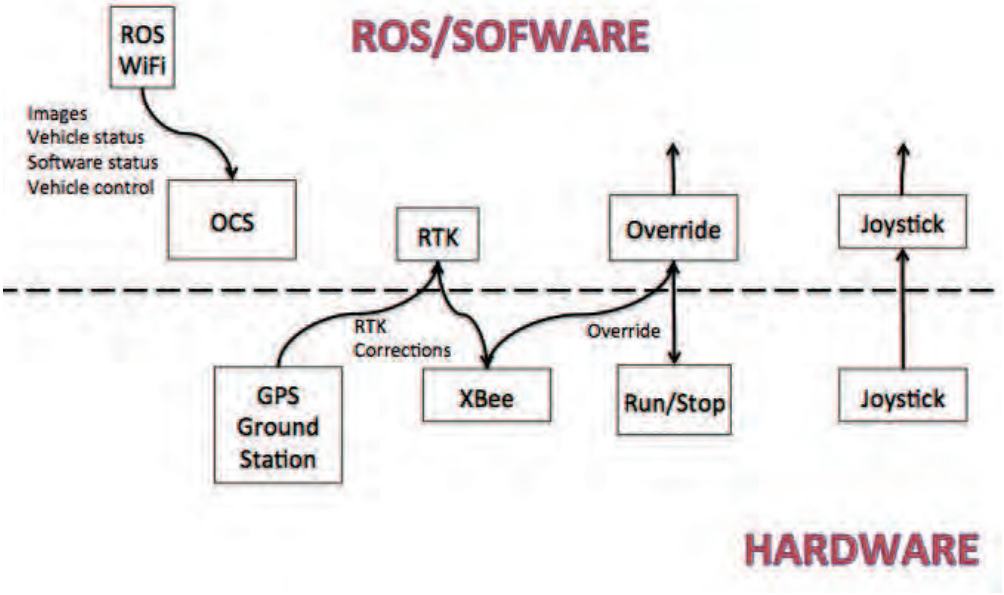
Other hardware setup by the remote operator is the base station GPS, run stop box and base station XBee. These devices provide a global run stop signal and RTK correction data over XBee to all robots within communication range. All of these devices must be plugged into the same computer, but by design it does not have to be an OCS. The run stop is a 4 button box with a locking red mushroom button and red, yellow, and green momentary buttons. To enable motion the mushroom button must be released and the green momentary button pressed once. All buttons are connected to an Arduino Uno mounted inside the run stop case that translates the button states into an enable motion signal sent over the XBee.

8.2 Servo Control Priorities

The most important part of the low-level software interface is how signals are sent to the actuators. All control signals sent from the compute box are go over USB to a Pololu Micro Maestro servo controller. On the computer, a servo interface node provides one interface point to the hardware by listening to all of the servo command messages. Servo command messages, which are a custom ROS message type, are accepted



(a) Platform Information Flow Diagram



(b) Operator Control Station Information Flow Diagram

Figure 11: Autonomous platform

from any node that is in a servo commander priorities configuration file. The file lists the name of each node and a priority relative to other potential actuator controllers. Because this is a research platform, there may be several different controllers running at any one time, each trying to control one or more actuators depending on the situation, but only one signal can be sent to each actuator. This interface provides a simple, automatic way to add new controllers and systematically swap between controllers while testing. The servo interface uses the priorities in the configuration file to select the highest priority valid command for each actuator. Note that commands for all of the actuators do not have to come from one node. This enables separate control of the throttle, front brake, and steering if desired.

9 State Estimation

In order to allow simple testing and development of planning and control strategies, an accurate estimate of the vehicle state is needed. Some vehicle state information, such as wheel speed, is best used as-is. However, used separately, GPS and IMU measurements are of limited usefulness. GPS is inherently low rate and gives no information about orientation. IMU measurements are very high rate, but do not directly give orientation or linear velocity information. By combining the time-synchronized signals from these sensors, a very accurate and high rate estimate of position, velocity, and orientation can be realized.

Previously, many systems have had great success with sensor fusion using methods such the Extended Kalman Filter. However, factor graphs combined with advanced inference algorithms such as iSAM2 [10] allow smoothing over many types of measurements, while retaining the ability to re-linearize previous information. This reduces many of the problems found in the Kalman filter with states or measurements that are not approximately linear in the time frame of measurements.

The iSAM2 algorithm is a part of the GTSAM [11, 12] software package, which uses a factor graph representation of the smoothing and mapping problem. The factor graph representation of sensor fusion is a method of visualizing states and measurements as a bipartite graph. The factor graph has two types of nodes, *factor nodes* $f_i \in F$ and *variable nodes* $\theta_j \in \Theta$. *Edges* e_{ij} always connect factor and variable nodes. The factor graph represents the factorization of the function $f(\Theta)$ as:

$$f(\Theta) = \prod_i f_i(\Theta_i), \tag{4}$$

where Θ_i is the set of variables θ_j adjacent to the factor f_i. Independence relationships in the measurements in $f(\Theta)$ are encoded in the edges e_{ij}, where each factor f_i is a function of variables Θ_j. The goal in sensor fusion is to find the variable assignment Θ^* that minimized the function $f(\Theta)$ in (4):

$$\Theta^* = \arg\max_{\Theta} f(\Theta) \tag{5}$$

To perform this optimization and estimation, a factor graph (Fig. 12) is constructed with successive measurements and iteratively optimized using GTSAM. At each smoothing time step, an additional set of states X, V, and B are added to the graph. Additionally, measurement factors for GPS and IMU are added along with a bias smoothness factor. In order to keep computational loads low while maintaining high accuracy, this graph contains state nodes for measurements taken at 10Hz. The factors in this graph correspond to GPS measurements and pre-integrated IMU measurements[13]. To obtain a state estimate at 200 Hz, IMU measurements are integrated to interpolate the 10Hz smoothed position at 200 Hz. Example trajectories are shown in Fig. 13.

In practice, the measurements from GPS can drift some from day to day, primarily due to the stationary RTK correction antenna being re-setup each day. To counteract these changes, the robot is started in the same position on the track each time the state estimator is started. This track position is used as the origin of a Euclidean coordinate system, oriented tangent to the GPS reference ellipsoid. This prevents GPS drift from effecting the vehicle state estimate relative to the fixed track boundaries.

10 Testing Facility

We constructed an outdoor dirt track on a recently acquired unused tract of land owned by Georgia Tech on the edge of campus that contains a large, relatively flat grassy area, a parking lot, and a warehouse used for furniture storage. A chain-link fence runs along the perimeter of the property and three padlocked access points limit access to the site.

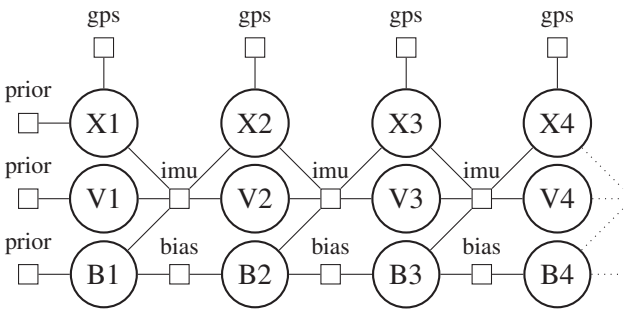
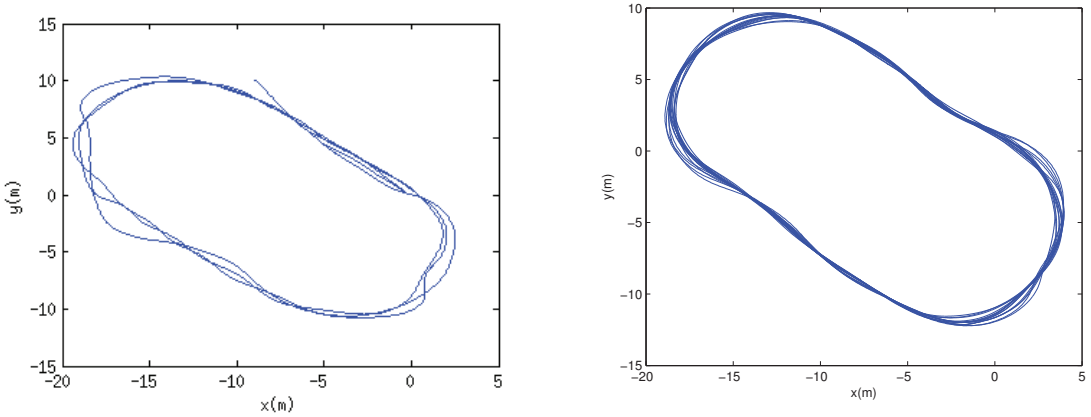


Figure 12: Factor Graph structure used for GPS/IMU fusion, circles represent states and squares represent factors



(a) Example pose of vehicle on track surface while sliding and performing aggressive maneuvers

(b) Example track of vehicle under control of waypoint based controller

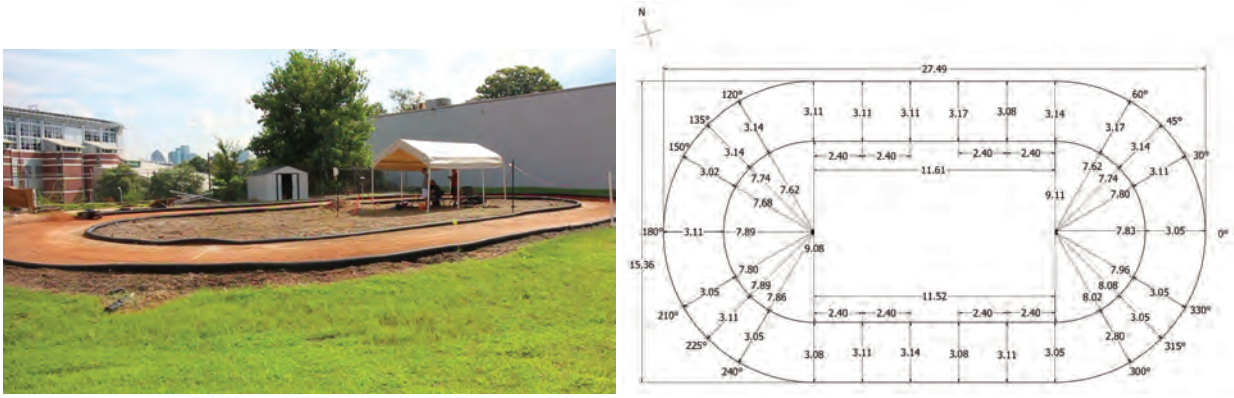
Figure 13: Example tracks generated by GTSAM based state estimator under different driving conditions

The track is a 3.3 m wide flat clay surface in the shape of an oval with outer dimensions of 27.5 m and 15.5 m. The boundaries of the track are 0.15m diameter corrugated drainage pipe which are secured in place with stakes to provide a semi-rigid crash barrier that will not damage the robots upon contact. The track is designed to allow the robot to operate autonomously at the limits of its mechanical, electrical, and software systems. On-site testing materials such as tables, chairs, a wireless access point and other tools are stored in a shed next to the track. A car port is setup in the middle of the oval to provide shade and protection from the elements while testing. Water and 120V AC power are available via exterior connections on the building next to the track.

11 Simulation Environment

Despite the robustness of the hardware platform, there are still maneuvers we want to test in simulation before performing moving to the hardware. The simulation also allows the careful control of environmental parameters for validation purposed. Other times, a simulation is valuable for performing repetitive or time-consuming experiments that would take weeks of testing on the physical platform.

For a simulation environment we chose to use Gazebo, which is a ROS-compatible, physics-based simulator. We created a simulation environment and robot model to match their real-world counterparts and



(a) Test track during testing

(b) As-Build track dimensions, units are in m and deg

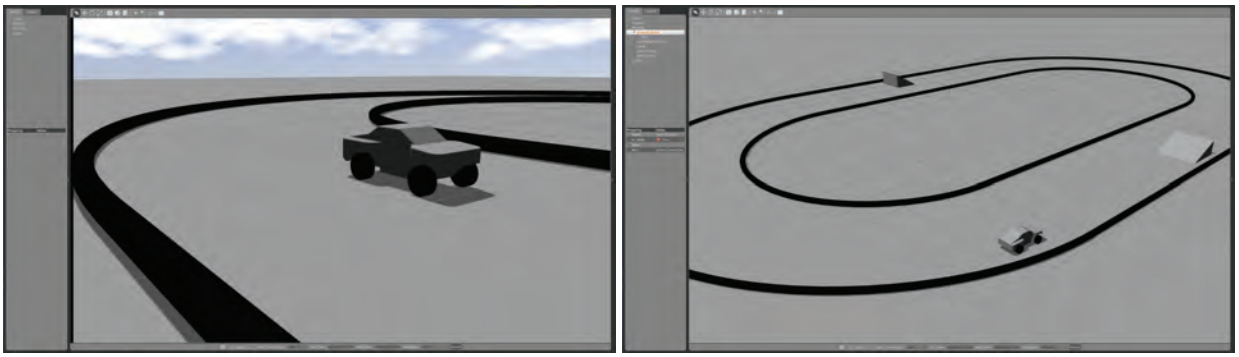
Figure 14: Georgia Tech Autonomous Vehicle Racing Facility

support the same software interface as the real hardware. The track and track boundaries were constructed using CAD software and the as-built track dimensions from the testing facility. The steering servo and Ackermann linkage of the physical robot are approximated by a ROS joint effort controller that applies torque to turn the wheel about the vertical axis of the inside of each front wheel. The unloaded rotation speed, maximum torque, and joint limits used in simulation are measured from the steering servo specs provided by the manufacturer and by measuring the steering linkage angles relative to the body center line. The electric motor, powertrain, and drivetrain are approximated by another ROS effort controller that applies torque on the rear axle of the Gazebo model. The maximum applied torque and angular velocity are calculated from the motor manufacturers' specs and calibration on the physical robot. The differential in the physical platform is neglected in simulation. The suspension for each wheel is modeled with a PID controller on a linear actuator with a target set-point which determines the ride height of the vehicle. The I and D terms are calculated from the dimensions and coefficients of the robot's spring configuration.

Switching between simulation and the physical platform is designed to be seamless from the software perspective. Simulated GPS, IMU, and cameras come from the `hector_gazebo_plugins` ROS package from TU Darmstadt. We developed our own wheel speed sensor node, as no similar functionality was publicly available. The AutoRally Gazebo simulation world provides the exact same ROS message interface as the physical platform and an environment with the same configuration as the testing facility. IMU, GPS, and cameras are all configured according to the specs of their physical analogs. While the images are of limited use because of the simplicity of the simulation environment compared to the real world, the IMU, GPS, and wheel speed data produce data very similar to data collected at the test track.

12 Waypoint Following Controller

In order to test all sensors and assure that state estimation is working as desired, a basic controller was created to drive the vehicle at a constant velocity following a pre-recorded set of waypoints. The controller consists of three pieces: (1) constant speed controller, (2) waypoint recorder, and (3) waypoint follower. The constant speed controller uses a proportional-integral feedback law and speed feedback from wheel speed sensors to maintain a specified velocity. The waypoint recorder/follower record a set of manually driven waypoints and follow these waypoints using a proportional control law on the heading error.



(a) Physics-based vehicle simulation in Gazebo

(b) Track with robot and jumps

Figure 15: Simulation environment of track and robot

12.1 Constant Speed Controller

The constant speed controller outputs a throttle value U_x in the range $[-1, 1]$ according to the following:

$$U_x = P_x(V_x - V_{x,des}) + I_x \int (V_x - V_{x,des}) \tag{6}$$

This value is fed to the motor controller after being transformed into a PWM signal value. A value of 1 corresponds to full throttle, while a value of -1 corresponds to full brake. The vehicle speed V_x is approximated by the rear axle rotational speed as measured by the rear wheel speed sensors. Error terms are generated relative to a desired velocity $V_{x,des}$. Integral and proportional gains I_x and P_x are multiplied by the error terms according to (6).

12.2 Waypoint Follower

Initially, waypoints are recorded by driving the vehicle along the desired trajectory manually. GPS coordinates are recorded at equally spaced intervals along the driven path. The job of the waypoint follower is to steer the front wheels of the vehicle so that the vehicle can navigate this set of waypoints. It has two tunable parameters, the look-ahead distance and a proportional heading error gain. Considering a heading error β (calculated as the difference between the vehicle heading and the bearing to the next waypoint) and control gain P_h, the steering command U_s in the range $[-1, 1]$ is calculated as:

$$U_s = P_h \beta \tag{7}$$

An illustration of heading error, look-ahead distance, and a recorded set of waypoints is shown in figure 16

13 Monocular Visual SLAM

For an autonomous vehicle to move from one point to another, it must have some notion about its position and orientation in the world and some type of world map. Currently, our most accurate robot state estimate comes from fusing GPS and IMU sensor readings. However, GPS is not always reliable, especially in cities with large buildings or in areas with dense tree cover. Vision sensors are comparatively inexpensive and do not require any additional infrastructure (such as GPS satellites or remote correction) sense the world. Simultaneous Localization and Mapping, or SLAM, is a class of algorithm that attempts to simultaneously localize a camera and build a representation of the world that is visible to that camera. To step toward a vehicle pose estimate without GPS, a monocular slam algorithm was tested and tuned for our vehicle.

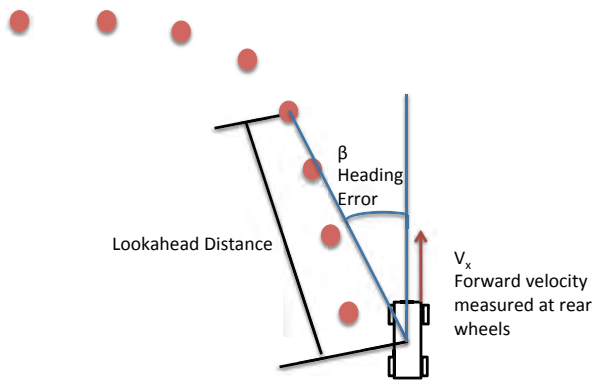


Figure 16: Waypoint following controller, with heading error β and forward velocity V_x. Waypoints are red circles.

Large Scale Direct Monocular SLAM (LSD-SLAM) [14] was chosen as our test algorithm. LSD-SLAM is a recent algorithm which represents the world as a pose graph of keyframes, and it generates a semi-dense depth map (which can generate a point cloud) along with pose estimates. We now briefly review the key steps in the LSD-SLAM algorithm.

13.1 Tracking

The camera's pose is a rigid body transformation, consisting of a rotation and translation. There are a variety of ways to compactly represent rotations in 3-space (e.g Euler angles, quaternions). A mathematically elegant method employed here is to recognize the space of rigid body transformations as a Lie group, and thus the elements of the group can be completely characterized by the groups Lie algebra. Geometrically this corresponds to associating all of the elements of the manifold of degree 6 containing all rigid body transformations to the manifold's tangent hyperplane located at the groups identity element.

Elements can be mapped to the space of rigid body transforms $\mathrm{SE}(3)$ from the associated Lie Algebra $\mathfrak{se}(3)$ by the standard matrix exponential $\exp(\xi)$ where ξ is an element of the Lie algebra. Using this fact its possible to define the 3D projective warp function ω which maps an image point $\mathbf{p}$ and its inverse depth to a location in another image frame differing from the current frame by the rigid body transform associated with ξ

$$\omega(\xi, \mathbf{p}, d) = \begin{pmatrix} x'/z' \\ y'/z' \\ 1/z' \end{pmatrix} \quad \text{where} \quad \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \exp(\xi) \begin{pmatrix} \mathbf{p}_x/d \\ \mathbf{p}_y/d \\ 1/d \\ 1 \end{pmatrix} \tag{8}$$

Using the 3D projective warp function we can define the photometric error term between a known image I_r and depth map, and a new image I as a function of the relative pose. This is defined as:

$$E(\xi) = \sum_i \left(I_r - I(\omega(\xi, \mathbf{p}, d))\right)^2 \tag{9}$$

The ξ which minimizes E then corresponds to the maximum likelihood estimate of the relative pose between the two images. Minimization is done with an iteratively re-weighted Gauss-Newton method. Additionally, since the depth map is uncertain, its important to normalize the error function by the variance of the inverse depth, and only pixels with sufficiently strong gradients are used in the optimization. We refer the reader to [14] for details.

13.2 Keyframe Creation and Depth Map Estimation

Each keyframe consists of a pose, image, and semi-dense depth map. The generated 3D point cloud is the union of all of the individual keyframe depth estimates. Keyframes are created when the pose reaches a certain generalized distance (measuring distance as a function of location and orientation) threshold from the current keyframe. When this happens the new keyframes depth map is generated by projecting points from the old keyframe into the new one. Then, when new frames are tracked, stereo correspondences are found using the method described in [15] between the current keyframe and the new image and depth from stereo is done by using the baseline from the estimated relative pose. This method repeatedly improves the keyframes depth map until a new frame is created and the keyframe is permanently added into the map.

13.3 Constraints Between Keyframes

The edges of the pose graph are similarity transforms (i.e., rigid body transform with scaling). Similarity transforms are used in order to account for scale drift, since the scale of the world is not observable in monocular SLAM it is possible for the scale to drift over long trajectories and accumulate large errors. Using similarity transforms overcomes this by allowing each keyframe to have its own depth (the mean depth of each keyframe is normalized to 1), and then figuring out a consistent scale later. Mathematically this corresponds to an extra term in a minimization procedure, the relative pose between two keyframes is estimated as:

$$E(\xi) = \sum_i \left(I_r - I(\omega(\xi, \mathbf{p}, d))\right)^2 + \left(\omega(\xi, \mathbf{p}, d) - D_j(\omega(\xi, \mathbf{p}, d))\right)^2, \tag{10}$$

where, once again, we have left out the normalization by the variance of the inverse depth, and the fact that the summation is only performed for points with sufficient gradient. The latter term penalizes deviations in the depth between the two keyframes. Note the $\xi \in \mathfrak{sim}(3)$ not $\mathfrak{se}(3)$ as in the previous sections.

13.4 Loop Closures and Global Optimization

When a new keyframe is added into the map, the closest 10 keyframes are considered as loop closure candidates. If a suitable candidate is detected a $\mathfrak{sim}(3)$ constraint is added between the two keyframes. Global map optimization is being continuously performed in the background by a pose graph optimization algorithm described in [16].

14 Initialization of LSD-SLAM

One of the main difficulties with LSD-SLAM, or any monocular SLAM algorithm, is initialization. In order for tracking to work, we need to have an existing keyframe with a depth map. But in order for the depth map to created and refined we need to be able to perform tracking. Upon initialization neither of these exist. The approach taken by LSD-SLAM is to randomly initialize the depth map, and hope that the algorithm quickly converges to a correct depth map. Although this may work in some cases, we found that this yielded very poor performance on our platform. Therefore some form of bootstrapping was necessary.

The approach we took was to take a single stereo image upon initialization and then give the depth map computed from stereo to LSD-SLAM for initialization. The depth map computed from stereo is sparse, however it sufficient to initialize the algorithm.

14.1 Testing

The more important test for our application was performance outdoors on a race track. For these experiments we drove the car around the track at low speeds (about 1 meter/sec) for three laps. Both of the random

Figure 17: Disparity map of first frame.

initialization and the stereo initialization produce a coherent point cloud and overall tracking is good.

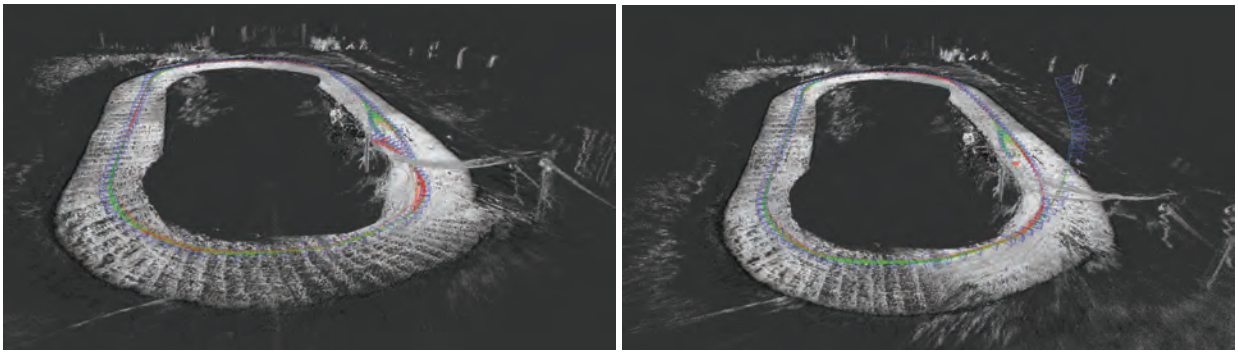


Figure 18: Mapping the race track. Left: with stereo initialization, Right: Default (random) initialization

However, there are some important differences between the stereo initialization and the default initialization. The most dramatic difference is that the default initialization appears to "fly in" to the track before converging to the correct configuration. Even though it eventually converges the keyframes created while it still had a poor depth map remain, and the vehicle occasionally returns to those keyframes which do not properly reflect reality. This could be problematic for a controller.

Additionally, the stereo initialized algorithm creates a much cleaner point cloud around the origin. Figure (8) demonstrates the differences between the two methods: in the random initialization the point cloud includes two versions of a string of cords overhanging the track and also includes two sheds facing the track. This orientation of the shed is incorrect: the shed is aligned with the straight-aways on the track not facing it. There is also only one shed.

15 Model Predictive Path Integral Controller

For aggressive autonomous control we deployed a model predictive path integral control (MPPI) algorithm on the vehicle [17]. This controller is able to maneuver the vehicle around the test track when the desired speed is set beyond the friction limits of the vehicle (Fig. 20b), and it is even able to perform controlled power slides around the corners (Fig. 20a). The MPPI algorithm uses the on-board GPU to sample thousands of possible trajectories from a dynamics model of the system, it then uses these samples to compute a control input according to the path integral control framework.

Path integral control is a branch of optimal control theory which allows for the mathematically rigorous development of optimal control algorithms based on random samplings of trajectories. The path integral

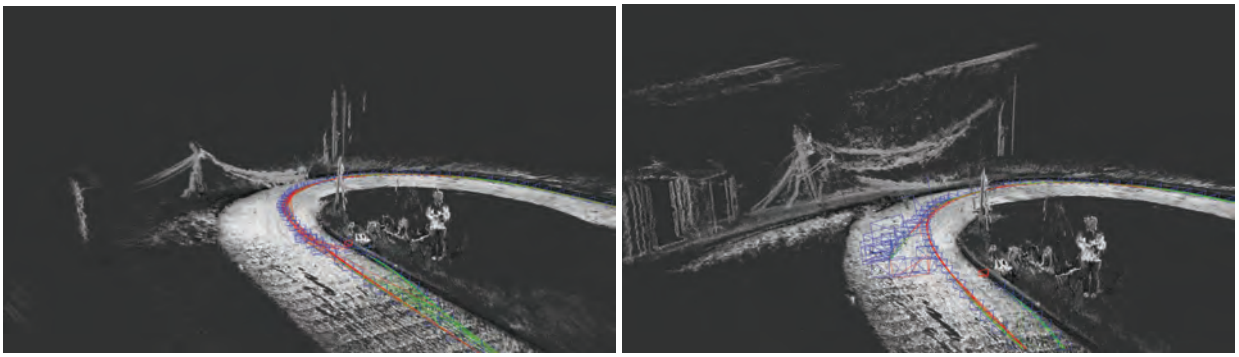
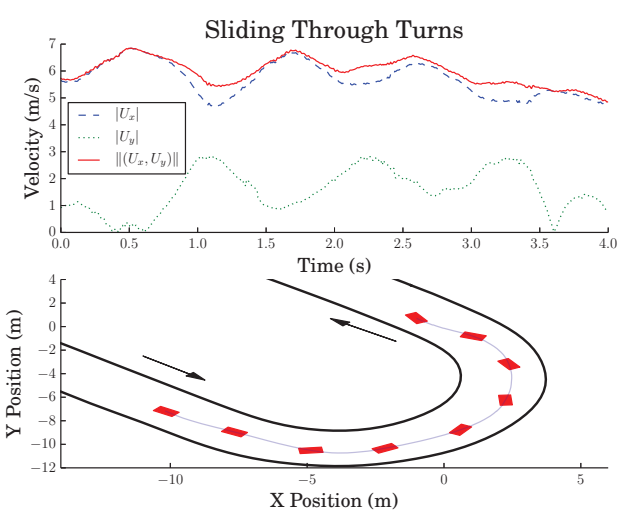
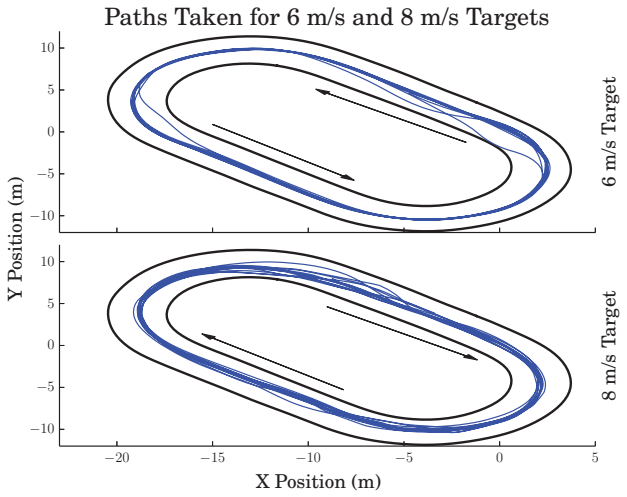


Figure 19: Point cloud of the outdoor track near the origin. Left: with stereo initialization, Right: Default (random) initialization



(a) The robot performing a power slide around a corner using the MPPI controller.



(b) Consistent navigation of the test track using the MPPI controller. Top: 6 m/s target speed, Bottom: 8 m/s target speed

Figure 20: Track result from PI-MPC controller.

control framework considers stochastic dynamical systems of the form:

$$\mathrm{d}\mathbf{x} = (\mathbf{f}(\mathbf{x}_t, t) + \mathbf{G}(\mathbf{x}_t, t)\mathbf{u}_t)\,\mathrm{d}t + \mathbf{B}(\mathbf{x}_t, t)\mathrm{d}\mathbf{w} \tag{11}$$

Here the state and controls at time t are denoted $\mathbf{x}_t \in \mathbb{R}^n$ and $\mathbf{u}_t \in \mathbb{R}^m$ respectively. These dynamics are disturbed by the Brownian motion $\mathrm{d}\mathbf{w} \in \mathbb{R}^p$. The classical objective in stochastic optimal control is to the find the controls $\mathbf{u}^*$ such that:

$$\mathbf{u}^*(\cdot) = \underset{\mathbf{u}(\cdot)}{\operatorname{argmin}}\, \mathbb{E}_{\mathbb{Q}}\left[\phi(\mathbf{x}_T, T) + \int_{t_0}^{T} \mathcal{L}(\mathbf{x}_t, \mathbf{u}_t, t)\right] \tag{12}$$

where the expectation is taken with respect to the dynamics (11). The running cost $\boldsymbol{\mathcal{L}}$ is composed of an arbitrary state-dependent term and a quadratic control cost:

$$\mathcal{L}(\mathbf{x}_t, \mathbf{u}_t, t) = q(\mathbf{x}_t, t) + \frac{1}{2}\mathbf{u}_t^{\mathrm{T}}\mathbf{R}(\mathbf{x}_t, t)\mathbf{u}_t \tag{13}$$

Algorithm 1: Model Predictive Path Integral Control

Given: K: Number of samples;
N: Number of time steps;
$(\mathbf{u}_0, \mathbf{u}_1, ...\mathbf{u}_{N-1})$: Initial control sequence;
$\Delta t, \mathbf{x}_{t_0}, \mathbf{f}, \mathbf{G}, \mathbf{B}, \mathbf{B}_E$: System/sampling dynamics;
$\phi, q, \mathbf{R}, \lambda$: Cost parameters;
$\mathbf{u}_{\text{init}}$: Value to initialize new controls to;

while *task not completed* **do**
  for $k \leftarrow 0$ **to** $K-1$ **do**
    $\mathbf{x} = \mathbf{x}_{t_0}$;
    for $i \leftarrow 1$ **to** $N-1$ **do**
      $\mathbf{x}_{i+1} = \mathbf{x}_i + (\mathbf{f} + \mathbf{G}\mathbf{u}_i)\Delta t + \mathbf{B}_E \epsilon_{i,k}\sqrt{\Delta t}$;
      $\tilde{S}(\tau_k) = \tilde{S}(\tau_k) + \tilde{q}(\mathbf{x}_i, \mathbf{u}_i, \epsilon_{i,k}, t_i)$;
  for $i \leftarrow 0$ **to** $N-1$ **do**
    $\mathbf{u}_i = \mathbf{u}_i + \boldsymbol{\mathcal{H}}^{-1}\boldsymbol{\mathcal{G}}\left[\sum_{k=1}^{K}\left(\frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_k)\right)\frac{\epsilon_{j,k}}{\sqrt{\Delta t}}}{\sum_{k=1}^{K}\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_k)\right)}\right)\right]$;
  send to actuators$(\mathbf{u}_0)$;
  for $i \leftarrow 0$ **to** $N-2$ **do**
    $\mathbf{u}_i = \mathbf{u}_{i+1}$;
  $\mathbf{u}_{N-1} = \mathbf{u}_{\text{init}}$
  Update the current state after receiving feedback;
  check for task completion;

In [17] it is shown that this optimal control problem is related to the following KL-divergence minimization problem:

$$\mathbf{u}^*(\cdot) = \underset{\mathbf{u}(\cdot)}{\operatorname{argmin}}\, \mathbb{D}_{KL}\left(\mathbb{Q}^* \| \mathbb{Q}(\mathbf{u})\right), \tag{14}$$

where $\mathbb{Q}(\mathbf{u})$ is the distribution of trajectories induced by the control input $\mathbf{u}$, and $\mathbb{Q}^*$ is an "optimal distribution" over trajectories which is described by the formula

$$\frac{\exp\left(-\frac{1}{\lambda}S(\tau)\right)}{\mathbb{E}_{\mathbb{P}}\left[\exp\left(-\frac{1}{\lambda}S(\tau)\right)\right]} \tag{15}$$

In this equation $\mathbb{P}$ represents the uncontrolled dynamics of the system, and $S(\tau)$ is the cost of the trajectory τ. Solving this KL-divergence problem yields the optimal controls in the form of a path integral. This then leads to the iterative approximation formula based on reward-weighted averaging:

$$\mathbf{u}^*_{t_j} \approx \mathbf{u}_j + \sum_{k=1}^{K}\left(\frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_k)\right)\delta\mathbf{u}_{t_j,k}}{\sum_{k=1}^{K}\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_k)\right)}\right) \tag{16}$$

Here $\delta\mathbf{u}$ is a random variation in the control input, and each τ_k is a trajectory generated using the system dynamics. Since the GPU is able to efficiently sample thousands of trajectories in real-time (15 milliseconds) this iterative update law can be applied in a model predictive control fashion. In this setting an iteration of optimization takes place, then a single control is executed, and then another iteration of optimization occurs

using the unused portion of the previously computed trajectory to warm-start the process. This algorithm is summarized in Algorithm 1.

The cost function consists of a cost for maintaining a desired speed, a control cost, and a cost for being on the track. The vehicle is able to keep track of its position in relation to the track using the current GPS signal and pre-computed GPS coordinates of the track.

References

[1] M. Buehler, K. Iagnemma, and S. Singh, *The 2005 DARPA grand challenge: the great robot race*. Springer Science & Business Media, 2007, vol. 36.

[2] ——, *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Springer, 2009, vol. 56.

[3] D. I. Katzourakis, I. Papaefstathiou, and M. G. Lagoudakis, "An open-source scaled automobile platform for fault-tolerant electronic stability control," *Instrumentation and Measurement, IEEE Transactions on*, vol. 59, no. 9, pp. 2303–2314, 2010.

[4] W. E. Travis, R. J. Whitehead, D. M. Bevly, and G. T. Flowers, "Using scaled vehicles to investigate the influence of various properties on rollover propensity," in *American Control Conference, 2004.*, vol. 4. IEEE, 2004, pp. 3381–3386.

[5] C. Doniselli, M. Gobbi, and G. Mastinu, "Measuring the inertia tensor of vehicles," *Vehicle Syst. Dynamics*, vol. 37, pp. 301–313, 2003.

[6] M. Gobbi, G. Mastinu, and G. Previati, "A method for measuring the inertia properties of rigid bodies," *Mech. Syst. and Signal Process.*, vol. 25, no. 1, pp. 305–318, 2011.

[7] M. Rozyn and N. Zhang, "A method for estimation of vehicle inertial parameters," *Vehicle System Dynamics*, vol. 48, no. 5, pp. 547–565, 2010.

[8] G. Genta and C. Delprete, "Some considerations on the experimental determination of moments of inertia," *Meccanica*, vol. 29, no. 2, pp. 125–141, 1994.

[9] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, 2009.

[10] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *International Journal of Robotics Research*, vol. 31, no. 2, SI, pp. 216–235, FEB 2012, 9th International Workshop on Algorithmic Foundations of Robotics (WAFR), Natl Univ Singapore, Singapore, SINGAPORE, DEC 13-15, 2010.

[11] F. Dellaert, "Factor graphs and GTSAM: a hands-on introduction," 2012.

[12] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, DEC 2006, Conference on Robotics - Science and Systems, Cambridge, MA, JUN, 2005.

[13] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[14] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision (ECCV)*, September 2014.

[15] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *IEEE International Conference on Computer Vision (ICCV)*, Sydney, Australia, December 2013.

[16] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 3607–3613.

[17] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," *IEEE International Conference on Robotics and Automation*, 2016 (submitted).