RoSCAR: Robot Stock Car Autonomous Racing

Kyle Hart[‡], Corey Montella[†], Georges Petitpas[†], Dylan Schweisinger[†], Armon Shariati[†], Ben Sourbeer[†], Tyler Trephan[†], and John Spletzer[†]

Lehigh University
[†]Department of Computer Science and Engineering
[‡]Department of Mechanical Engineering

ABSTRACT

In this paper, we present the development of a low-cost, high-performance mobile robot platform for educational and research use called RoSCAR. The platform is based on a 1/10-scale short track race car, integrated with an on-board desktop-class computer, odometry, and RGB-D sensing. As part of an experimental robotics course, three student teams used RoSCAR to compete in a race around a 200 m indoor track. Students demonstrated reliable lane tracking and attained average lap speeds of up to 5.7 m/s (equivalent to 202 km/h at full scale). We are currently investigating the viability of a regional robotics competition centered around this concept.

1. INTRODUCTION

Robotics is currently an active area in education, with a wide variety of educational robotics hardware, courses, and competitions designed to engage students and get them interested in various STEM fields. Unfortunately, many educational robotics competitions lack a focus on advanced concepts in robot algorithms, such as closed loop control or 3D perception [1].

For instance, the most popular robotics competition at the high school level is the For Inspiration and Recognition of Science and Technology (FIRST) Robotics Competition, which involves no perception and limited autonomous control [2]. A majority of the competition is performed via tele-operation. At the undergraduate level, the Brown IEEE Robotics Olympiad entails the design and construction of a maze following robot, which requires limited perception but little opportunity to explore advanced control concepts [3]. The Trinity College Fire Fighting Home Robot Contest faces the same shortcoming; competitors build robots to navigate a maze to find and extinguish a flame [4].

A common thread in the aforementioned competitions is that the competitors design and build both robot hardware and software for the competition. We propose that a common, high-performance platform available to all teams shifts focus to advanced algorithm development. To this end, we designed a high-speed mobile robot platform called RoSCAR for research and educational use, pictured in Figure 1. The RoSCAR platform surpassed our expectations with respect to robustness and was remarkably agile, with a maximum speed 16.7 m/s. We believe that this platform could be the basis for a new robot racing competition focused on the design of algorithms for high speed control and advanced perception.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MARS'14, June 16, 2014, Bretton Woods, New Hampshire, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2823-4/14/06 ...$15.00.
http://dx.doi.org/10.1145/2609829.2609837.

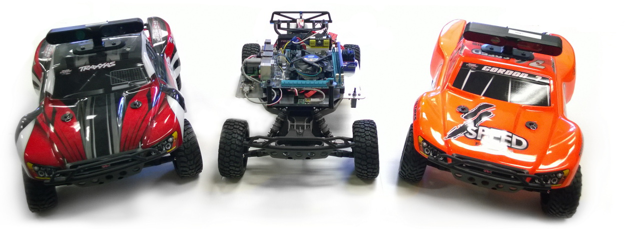
Figure 1: Three RoSCAR platforms pictured with and without shell attached. The only external distinguishing feature is the RGB-D sensor mounted on the roof (left and right cars). Internally, a mini-ITX x86 computer is mounted to the chassis, along with servo control, encoder, and multiplexer hardware (center car).

The most popular robot platform for research and education is the iRobot Create [5]. The Create is a low-cost robot platform featuring differential drive locomotion, native odometry, a bump sensor, and an expansion port for an on-board computer. Users need only provide a laptop and sensor suite. The Create is used by hobbyists, teachers, and researchers, with plenty of accessories and open source software available (see the original Turtle Bot [6]).

Despite its success, the Create platform has its drawbacks, the most significant of which being its slow maximum linear speed (0.5 m/s) and that operation is limited to flat terrain. Addressing the limitations of the Create and incorporating additional functionality, are several other platforms such as the MARS Clodbuster [7], a fairly older platform developed by the University of Pennsylvania; and COMET [8], a multi-vehicle platform developed by the University of New Mexico.

In this paper, we describe the design and development of the RoSCAR platform. In Section 2 we describe the hardware architecture, including the chassis, on-board computer, and sensing suite. In Section 3, we discuss the software architecture. In Section 4, we put RoSCAR to the test, and de-

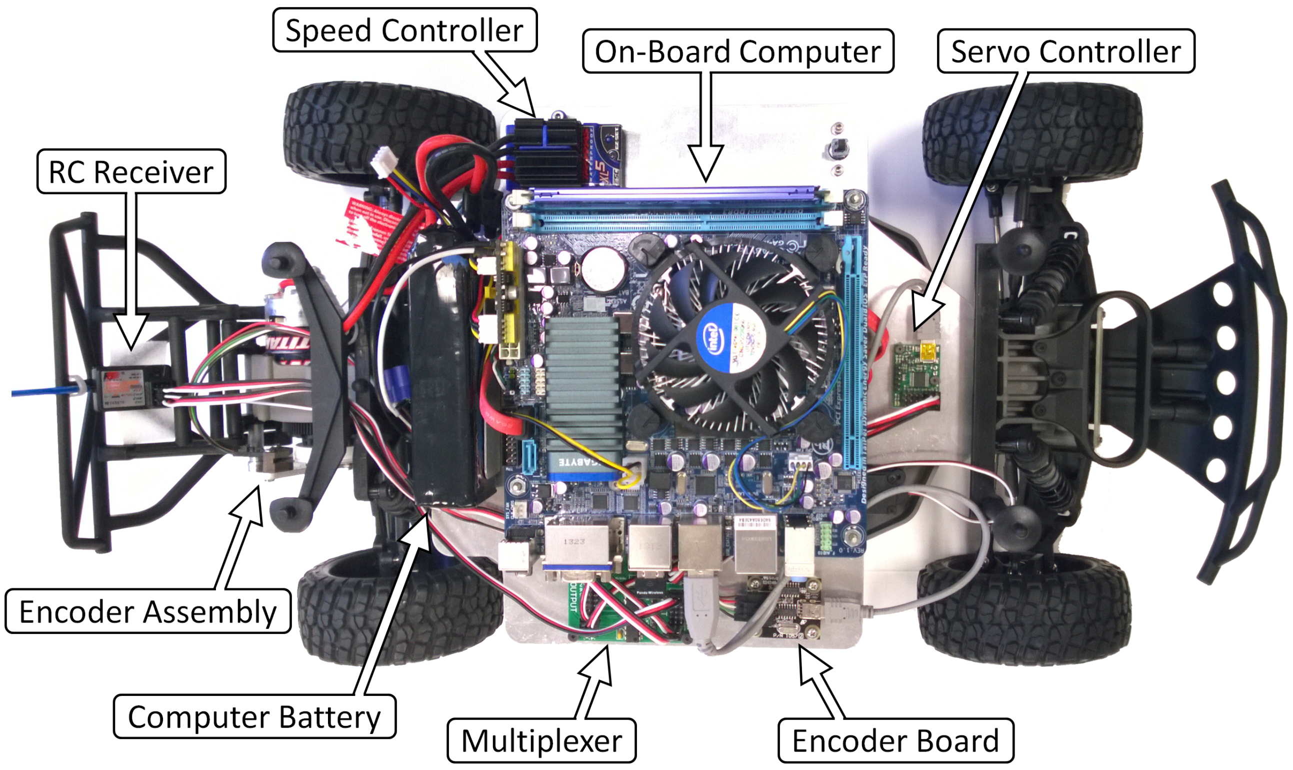

Figure 2: Assembled RoSCAR platform. The mini-ITX motherboard supports a Intel Core i3 processor, 4GB of RAM, and 32GB of solid state storage (mounted underneath the motherboard). The platform is completed with a decorative shell.

scribe the results of an experimental course using RoSCAR, taught in the Fall semester of 2013. We conclude by discussing future directions in Section 5.

2. HARDWARE ARCHITECTURE

RoSCAR is based on a short-track 1/10 RC Traxxas stock car. Mounted on the frame of the car is a mini-ITX PC which ultimately is responsible for control in its autonomous mode. In this section we will detail modifications made to the base platform, as well as the computing and sensing hardware that brought RoSCAR to life.

2.1 Chassis

The RoSCAR platform is a modified version of the Traxxas Slash 2WD 1/10-scale RC car [9]. To support our on-board computing platform, we added an aluminum mounting plate, which required the relocation of the radio receiver, speed controller, and motor battery mount. The base plate is large enough for mounting the on-board computing hardware, computer battery, servo controller, multiplexer, and encoder board. Figure 2 depicts the assembled RoSCAR platform with call-outs indicating the aforementioned components. Further components not visible in Figure 2 are the motor battery, which is located on the underside of the platform; and the solid state hard drive, which is located under the motherboard. The base platform was further modified with the addition of upgraded shocks, rated at 1155 N/m to account for the increased payload.

RoSCAR can operate under two modes: training mode and standard mode. In training mode, speed is capped at half the maximum operating speed. Figure 3 depicts the performance of the RoSCAR platform in training and standard modes for a straight drag race. For the purpose of obtaining this comparison, we did not optimize acceleration, but instead were only interested in estimating the top speed in each mode. In training mode (blue line) RoSCAR achieved a top speed of 8 m/s. In standard mode, this is increased

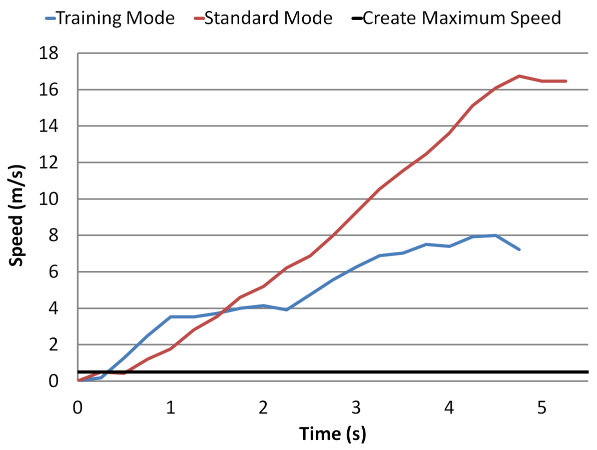

Figure 3: Drag race performance of the RoSCAR platform. Under training mode (blue), RoSCAR attained a peak speed of 8 m/s. On standard mode (red), the peak speed was 16.75 m/s.

Part	Description	Vendor	Cost
Base Platform			
Chassis w/ Battery	Traxxas Slash Pro 2WD Short-Course Truck	*towerhobbies.com*	$207.00
Transmitter	FlySky FS-GT2B 2.4GHz 3-Channel Transmitter	*amazon.com*	$36.64
Encoder Disk	DISK-2-1024-441-NE	US Digital	$15.08
Encoder Head	EM1-2-1024-N	US Digital	$41.80
USB Encoder Board	1057 2 Phidget Encoder High Speed	Phidgets	$60.00
Servo Controller	Pololu Micro Maestro 6-Channel USB Servo Controller	Pololu	$19.95
Multiplexer	Cytron 8-Channel RC RX Multiplexer	*robotshop.com*	$12.90
		Subtotal	**$393.37**
On-Board Computing			
Motherboard	Gigabyte GA-H61N-USB3 Mini ITX Motherboard	*newegg.com*	$76.99
Processor	Intel Core i3-3220T Dual Core Processor 2.8 GHz	*amazon.com*	$127.98
Memory	Kingston HyperX Blu 4GB 1333MHz DDR3 CL9 DIMM	*amazon.com*	$34.24
Storage	Sandisk ReadyCache 32 GB SSD	*amazon.com*	$44.60
USB Wi-Fi Adapter	Panda Ultra WiFi Wireless-N 2.4Ghz Adapter	*amazon.com*	$10.99
Power Converter	PicoPSU-90 Cyncronix 90W 12V DC-DC ATX mini-ITX	*amazon.com*	$29.95
Battery	Venom LiPo 3S 11.1V 5400mAh 20C	*towerhobbies.com*	$61.99
		Subtotal	**$386.74**
RGB-D Sensor	Asus XtionPro Live RGB and Depth Sensor	*newegg.com*	$169.99
		Total	**$950.10**

Table 1: RoSCAR parts list, vendor, and current pricing (as of April 2014). Costs are divided by the base RoSCAR platform, which and the computing platform, which is roughly equivalent to the cost of a netbook computer.

to almost 17 m/s (red line). By way of comparison, the top speed of the iRobot Create is indicated at 0.5 m/s (black line).

2.2 On-board Computer

Our choice of a mini-ITX x86 computing platform was motivated by several factors. First, mini-ITX is the smallest form factor available with mainstream component support. This makes it low cost, and components such as the processor, memory modules, and primary storage can be upgraded with readily-available commodity hardware. Second, as we will discuss in Section 3, the choice of MATLAB as a development platform necessitated an x86 processor architecture. Third, Intel offers an energy-friendly Core i3-3220T processor, which at 35 W TDP, balances our requirements of long battery life and fast performance for point-cloud processing. This processor allowed our platform to operate for nearly one hour at full load powered by a 48 Wh battery. Our computing platform used a single module of 4 GB DDR3 RAM, and a 32 GB solid state disk for mass storage. A compact USB Wi-Fi adapter was used to establish an ad-hoc wireless network between the on-board computer and development computers.

Several daughter boards were also necessary for interfacing with our sensors and servos. A Phidgets 1057 2 high-speed encoder board was used to read data from the encoder. Both the front and rear servos were controlled by the Pololu Micro Maestro 6-channel USB servo controller. The electronic speed controller (ESC) that was removed from the chassis previously was later mounted on top of the plate. Finally, a Cytron 8-channel RC RX multiplexer was added for both convenience and safety. The multiplexer toggled servo input between the Maestro controller and a wireless receiver which was paired to wireless transmitter for manual control. The servo input was switched from one input to another when the user pressed a button on the wireless transmitter. This way, if a RoSCAR was ever about to crash, go off course, or just needed to be driven to a specific location, the user could toggle the button on the transmitter and manually control the car.

2.3 Proprioceptive Sensing

Our main proprioceptive sensor was an encoder disk secured on the rear axle of the car. The encoder disk is 2 inches in outer diameter, with an inner diameter of 0.441 inches and 1024 counts per revolution. In order to mount the encoder disk in this location, it first needed to be modified with three holes for mounting. Then we used a set of screws to attach the disk hub, the encoder disk, and the disk cover to a gear. The encoder head was attached to a plate which was then screwed onto the differential box.

We used only one encoder and it was placed in the middle of the axle, providing us with the average angular velocity of both wheels. This information was used to estimate linear velocity of the RoSCAR. We currently do not have the capability to estimate angular velocity. However, we are exploring the possibility of mounting a low cost gyroscope to the platform.

2.4 Exteroceptive Sensing

Exteroceptive sensing on the RoSCAR platform was performed by the Asus Xtion Pro Live RGB-D sensor [11]. With a resolution of 640 $\times$ 480 pixels, a 58° $\times$ 45° field of view, and a frame rate of 30 fps, this sensor is commonly used for tasks such as skeleton tracking, facial recognition, and object detection. Figure 4 (left) depicts an Xtion sensor mounted to the roof of a RoSCAR shell. The sensor is hinged, so its gaze can be adjusted easily. In our case, it focused on the ground in front of the RoSCAR to facilitate lane segmentation. Figure 4 (right) displays a point cloud taken from the

Figure 4: (Left) The Asus XtionPro RGB-D sensor is mounted on the roof of the RoSCAR shell. The sensor provides 640 x 480 points at 30 fps (Right) Visualization of 3D point cloud with RGB color registered to each point, taken from the view of a racing RoSCAR. Here, a second RoSCAR can be seen.

point of view of a RoSCAR, with RGB values registered to each point.

In the context of RoSCAR, point clouds from the Xtion sensor were analyzed to segment features using common image processing techniques, such as color segmentation and edge detection. The corresponding features were then extracted from the data provided by the depth sensor. Student solutions included transforming point clouds in the sensor frame, where lane markings intersect at a horizon line; to a world frame, where lane markings are parallel lines. Then lanes were segmented using image processing techniques.

2.5 Construction

Table 1 lists the various components of RoSCAR, vendors from which they were purchased, and their prices as of April 2014. Not included in this list is the cost of custom machined parts, such as the motherboard mounting plate. In this table, we draw a distinction between the base platform, and the computing platform. This is to facilitate a direct comparison between other educational platforms that do not include a computing platform in their base price, such as the iRobot Create. Total cost of the RoSCAR base platform is \$393.37, neglecting tax and shipping. By way of comparison, the iRobot Create currently (as of April 2014) sells for \$220 with battery [5].

The cost of the computing platform with battery totals \$386.74, which is comparable to a netbook computer of similar performance that might be used in conjunction with other robot platforms. With the Asus Xtion at \$169.99, the total parts cost of the RoSCAR is \$950.10. Not included in this cost is the necessary labor to assemble the RoSCAR.

Instructions, diagrams, and downloadable CAD files necessary to assemble a RoSCAR can be downloaded at our website: http://vader.cse.lehigh.edu/roscar

3. SOFTWARE ARCHITECTURE

Because the processor architecture is x86, we have wide latitude in what is installed on the on-board computer. In a competition setting, competitors would be provided a universal set of drivers to access all sensors, and could choose any middleware they are familiar with to complete the race, such as ROS [12]. Our software architecture was developed using lessons learned from the DARPA Urban Challenge (DUC) as part of the Ben Franklin Racing Team (BFRT) [8]. Software development for Little Ben, the BFRT entry, was done in MATLAB. While the code bases of other competitors using C++ or Java was measured in the 100,000s of lines, Little Ben needed less than 5,000 lines of MATLAB code to complete the 55 mile race. In this section, we detail the software architecture, and the necessary components to enable the car to run using MATLAB. All software detailed herein ran on Ubuntu 12.04 64-Bit.

3.1 Inter-Process Communication

In the DUC, Little Ben was architected as a set of MATLAB processes, each of which acted as producers and consumers of data. These processes (often called "nodes") corresponded to high-level tasks such as perception, control, and planning, which were interfaced using an inter-process communication library called the Spread Toolkit [13]. We replicate this paradigm in RoSCAR, with some new modifications.

The Spread Toolkit is a C-library that allows a process to broadcast messages over a network, or between nodes running on the same computer. It works by running a central Spread Daemon, which allows processes to subscribe to a "topic", which is just a channel of communication over which messages are exchanged. Messages can be published to a topic, and then received by every other process subscribed to that topic. Since Spread exists as a C-library, we used the MATLAB Executable (MEX) interface to wrap Spread for access within MATLAB.

Aside from simply wrapping Spread, we extended its functionality to facilitate rapid development of the aforementioned robot architecture, so students can focus on algorithm development. We call this extension RoLAB, short for Robot MATLAB. RoLAB takes the spread interface and adds the following facilities:

- Implements an efficient serialization and deserialization protocol written in C, to handle the serialization of MATLAB datatypes including multidimensional arrays, doubles, ints, chars, strings, bools, and structs.
- Allows a programmer to specify whether a node publishes or subscribed to a topic.
- Handles message sorting and management within a node.
- Allows a programmer to specify a node frequency at which the node will check for new data or publish data over the Spread network.

With MATLAB and RoLAB, defining a new node can be accomplished in as few as 5 lines of code. This includes

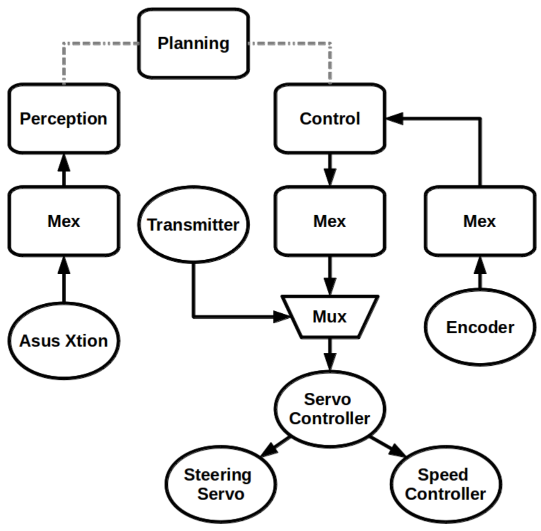

Figure 5: Software architecture running on RoSCAR. Perception and Control are separate MATLAB processes and communicate via message passing (dotted lines) to a Planning MATLAB process. Each process communicates to hardware via MEX wrappers.

instantiating the core RoLAB object (1 line of code), creating either a publisher or a subscriber (1 line of code), and initializing the main node loop (3 lines of code).

3.2 Student Interface

Students had access to MATLAB, RoLAB, and MEX interfaces to the hardware components including the Asus Xtion, encoder, velocity control, and steering control. The rest of the software architecture was left for them to design. Students typically interfaced with the on-board computer via an ad-hoc wireless network, which was indispensable for testing this mobile robot platform. For more control over the hardware, students also could connect a monitor and keyboard to the on-board computer and access a desktop environment, although this negated the mobility of the platform.

Students were free to architect their RoSCAR as they chose. An example software architecture is depicted in Figure 5. Here, the student-designed portions consist of a perception node, a control node, and a planning node. The perception node interfaced with the RGB-D camera via a MEX wrapper, and filtered depth images to segment lane boundaries. These boundaries were transmitted to the planning node via RoLAB (the dotted line). Using the lane boundaries, the planning node generated a global plan for the robot to follow, which was transmitted to the control node. The control node interfaced with the encoder, steering, and velocity controls. It generated control signals to send to the steering servos and DC motor in order to execute the plan sent by the planning node.

The foregoing node architecture is simply one example of how RoSCAR was designed by a student team. Thus, the details of these node are omitted from this paper, because they necessarily vary from student to student. The implementation of these nodes involves high-level perception algorithms which draw on the field of computer vision and image processing; and control algorithms which use elements of control theory and motion planning. How the students develop their algorithms determine how well their RoSCAR will perform in the race.

4. PILOT COURSE

To test the efficacy of RoSCAR as a robot platform suitable in a competition and educational environment, we used an experimental course held at Lehigh University during the fall semester of 2013. The course was comprised of advanced undergraduates (junior/senior level) and graduate students (masters/first year Ph.D.) who had previously participated in an introduction to mobile robotics course. This prerequisite course focused on algorithms employed in mobile robotics for navigation, sensing, and estimation. Topics included common sensor systems, motion planning, robust estimation, Bayesian estimation techniques, Kalman and particle filters, and localization and mapping techniques. Three undergraduate and four graduate students participated in the RoSCAR course. The students were divided into three teams of 2-3 students. Each team was given a RoSCAR platform on which to develop over the semester. This section details the structure and outcome of the course.

4.1 Curriculum

The course was project-based, where most class time was allocated for development on the hardware platform. Incremental development goals of increasing difficulty were assigned to the students throughout the semester, which cumulatively prepared the teams to compete in a capstone race at the Lehigh University Rauch Field House. These goals included calibrating the steering servo, calibrating the RGB-D camera, transforming point-clouds between frames of reference, segmenting lanes, following a straight track segment, and following a curved track segment.

Rauch Field House, pictured in Figure 6, is a 5,760 m^2 indoor track with ample space for field testing. Each lane of the track is 1 m wide with a length of 200 m. The objective of the final race was to circle the track while staying in their respective lanes without violating any lane boundaries. The large indoor environment served as a testing site for the platform, as it was isolated from pedestrians and provided many features to test the performance capabilities of RoSCAR.

Also visible in Figure 6 is substantial clutter on the track surface, which made lane detection more challenging. This necessitated not only robust segmentation algorithms, but also filtering and tracking algorithms to make sure the appropriate set of lane markings were followed. This clutter ensured the proposed task was more challenging than building a simple line following robot.

4.2 Results

Here we present the results of the course and some observations on student performance. The results of the final competition are presented in Table 2. All RoSCARs were configured in training mode for the final exam. Impressively, the first place team managed to max-out the speed of their RoSCAR in training mode, with a best lap time of 35 seconds. One interesting feature of the results is the spread between the first place team and last place team, which had a best lap time of 160 seconds. We believe this is due to the layered design of the course; since each new benchmark depended on successful completion of the previous benchmark,

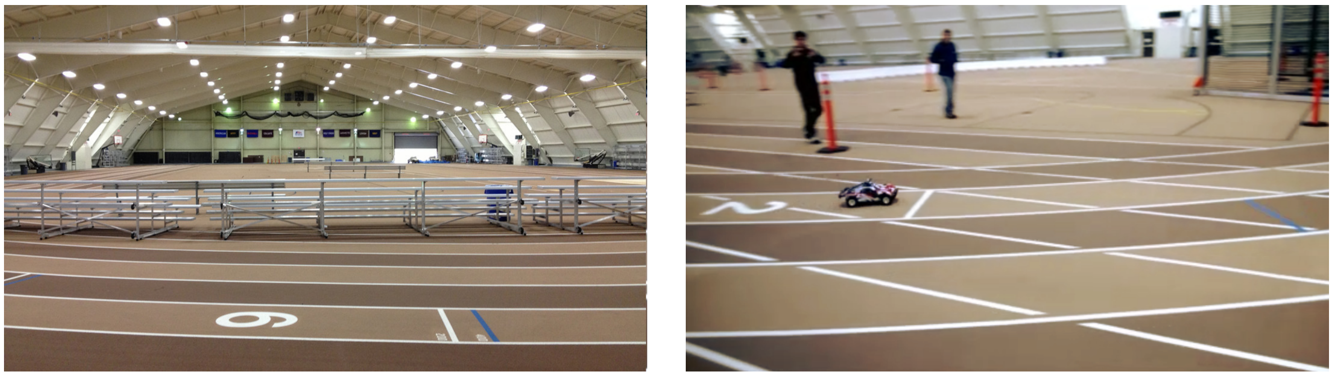

Figure 6: (Left) A view from Rauch Field House. Total track length around the course was 200 m, which is appropriate for the scale of the RoSCAR. (Right) Extraneous markings on the Rauch track necessitated robust solutions to lane segmentation and tracking.

a team struggling with one component tended to fall behind. After initial development of algorithms, teams worked on incrementally increasing speed, which sometimes necessitated the development of new algorithms as previous methods failed at higher speeds. Thus, a struggling team did not have the opportunity for this fine-tuning phase.

Nonetheless, all three teams were able to complete at least one lap around the track. A video of the final exam can be viewed at: http://youtu.be/jwRjv3D7lGo.

Table 2: Final Race Results

Place	Best Lap Time	Average Speed	Scaled Speed
1	35 seconds	5.7 m/s	202 km/h
2	46 seconds	4.5 m/s	162 km/h
3	160 seconds	1.25 m/s	45 km/h

5. FUTURE WORK

We have presented RoSCAR, a low-cost, high-performance platform designed for educational and research use. From the results of our experimental course, we believe RoSCAR is ideal for bridging the gap between theory and practice for advanced undergraduates and graduate students. However, we are only just beginning to realize the potential of RoSCAR. In our experimental course, RoSCAR was tested in training mode on a level surface. However, as Figure 3 shows, RoSCAR can attain speeds of nearly 17 m/s. Further, while we have yet to test RoSCAR in an outdoor setting, the Traxxas Slash Truck was originally designed as a competitive off-road race car. With some modifications to the sensor suite and the hardware housing, we expect the RoSCAR platform could potentially perform effectively outdoors as well.

6. ACKNOWLEDGMENTS

The authors would like to thank the following students for their participation in the RoSCAR course: Tashwin Kaur Khurana, Vikram Mehta, Jim O'Connor, Adam Schaub, and Elton Wong. We also extend a special thanks to the Lehigh University Department of Computer Science and Engineering for their generous financial support of this project. Finally, we would like to thank David Luksenberg for early discussions about a Robot NASCAR course.

7. REFERENCES

[1] J. R. Croxell, R. Mead, and J. B. Weinberg, "Designing robot competitions that promote ai solutions: Lessons learned competing and designing," pp. 29 – 34, 2007.

[2] FIRST, "First robotics competition." [Online]. Available: http://www.usfirst.org/roboticsprograms/frc

[3] Brown ieee. Brown IEEE Robotics Competition. [Online]. Available: http://www.brown.edu/Departments/Engineering/Organizations/Ieee/competition/

[4] T. College, "Trinity college fire fighting home robot contest." [Online]. Available: http://www.trincoll.edu/events/robot/

[5] iRobot, "irobot create programmable robot." [Online]. Available: http://www.irobot.com/us/learn/Educators/Create.aspx

[6] O. S. R. Foundation, "Turtlebot." [Online]. Available: http://www.turtlebot.com/

[7] G. Laboratory, "Mars, multiple autonomous robots." [Online]. Available: http://www.cis.upenn.edu/mars/site/platforms.htm

[8] D. Cruz, J. McClintock, B. Perteet, O. A. Orqueda, Y. Cao, and R. Fierro, "Decentralized cooperative control," *IEEE Control Systems Magazine*, pp. 58–78, June 2007.

[9] Traxxas, "Slash, 1/10 scale pro 2wd short course race truck." [Online]. Available: http://www.cis.upenn.edu/mars/site/platforms.htm

[10] MathWorks, "Matlab: The language of technical computing." [Online]. Available: http://www.mathworks.com/products/matlab/

[11] A. C. Inc., "Xtion pro live." [Online]. Available: http://www.asus.com/Multimedia/Xtion_PRO_LIVE

[12] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[13] S. C. LLC, "The spread toolkit." [Online]. Available: http://www.spread.org/