MIT Beaver Works Technical Report

Jonathan Borowsky

8/19/16

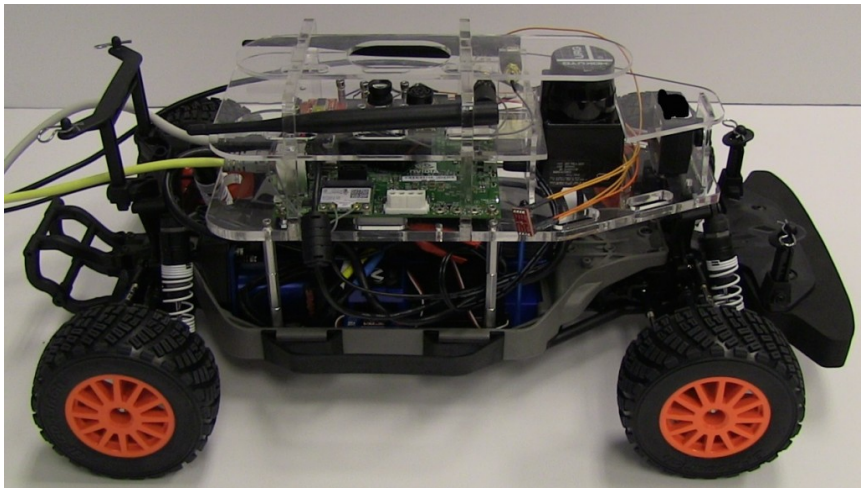

Figure 1. One of the cars which we used for the course (1)

Introduction

Rapid advances in computer technology over the past few decades have made the field of robotics increasingly prominent. Powerful new computers have greatly advanced the capabilities of robots and expanded the range of tasks that can be automated. Sensor technology has also advanced rapidly, producing detailed real time data for robots to use. As a result, there is huge potential for autonomous algorithms which can use the newly available technology efficiently to transform the role of autonomous machines in our lives. One area with particular potential is self driving vehicles, especially cars. Car accidents kill 32,000 people annually in the US, and seriously injure 2.5 million (2). Furthermore, unlike applications such as disaster response robots, the hardware involved, the car, is already very well developed. Therefore, while sensor and computer technology continues to develop quickly, developing autonomous cars is primarily a matter developing software to interpret and react to sensor data. Great progress has been made in recent years, most notably with Google's self driving car (3), which is intended as a fully autonomous vehicle. Other automakers such as Tesla and Continental Tire have worked to develop autonomous subsystems to aid a human driver in specific activities such as lane detection at night or highway cruise control (4). These developments, and the pace of development of the supporting technologies, suggest that self driving cars will soon become a reality.

The beaverworks class is intended to teach students about the growing field of autonomous robotics. The class focuses on autonomous cars due to the developments described above, as well as the ease of working with miniature cars compared to other types of robot. It explores cameras and lidar sensors, two sensors frequently involved in actual self driving cars. The former has great potential due to its high resolution and frequency, but computing power and algorithms needed to process camera data effectively are still under development. Lidar, a laser based radar, provides more reliable data than cameras, and requires less processing, but has much lower resolution and frequency. The combination of these provides an interesting opportunity for students to explore different areas in autonomous robotics. To apply what they learned, the students participated in a series of races, culminating in a grand prix which combined the code and skills that they had developed in the previous races. The grand prix required both obstacle avoidance using lidar and sign detection using a camera, with vehicle interaction providing additional navigational challenges.

Week 1

Goals/overview

The main goal of the first week was to learn how to use Robot Operating System (ROS) to control the racecars. ROS is a tool designed to make the programming of robots straightforward, efficient, and standardized. It comes with various code libraries and supports programming of nodes in multiple languages to achieve this purpose. The nodes communicate via topic spaces through message objects containing various pieces of data (5). Learning about the racecar sensor capabilities, PID (Proportional, Integral, Derivative) control systems, and implementing wall following were additional goals.

Details

PID (Proportional Integral Derivative) control is a simple but effective control system. Each PID controller controls a single degree of freedom of a system, and seeks to move it to a single desired value. The proportional term moves to correct the error efficiently by reacting to errors based on their size. Derivative control prevents proportional control from overshooting the target value or oscillating around it. Integral control keeps track of the total error to stop the system from converging on values near but not equal to the desired one, which may happen due to a continuous drift or a low Kp value.

$$R = Kp*e+Kd*(de/dt)+Ki*\int(e*dt) \quad \text{(Eq. 1)}$$

Kp, Kd, and Ki are constants, which must be tuned for the controller to work properly. For mathematical reasons, Kp and Ki have one sign, while Kd has the opposite sign. The variable e (error) is the value of the quantity being controlled minus its desired value, while R is the response value applied to correct for the error.

With our racecar, we used PD control for steering to keep a constant distance from the wall. The PD controller controlled the steering angle of the car based on a distance calculated from lidar input. An integral term was not used because a small constant deviation from the correct wall following distance does not interfere with wall following performance. PD control was sufficient to produce smooth, stable wall following. Several methods were investigated to find distance for the PD controller input. These methods are discussed below.

1. **Single point estimation**

A single lidar point at a fixed angle from the front of the car is used to calculate distance based on the assumption that the car is parallel to the wall.

d = estimated distance to wall
r = distance of lidar point
a = angle of lidar point from the front of the car

$$D = r*\sin(a) \quad \text{(Eq. 2)}$$

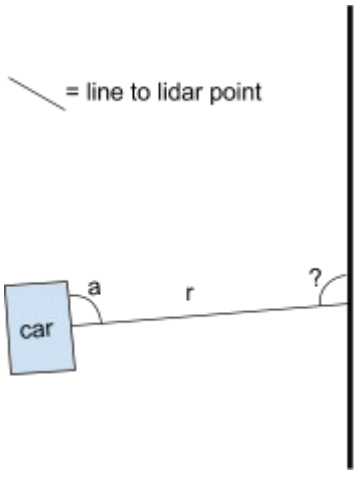

Figure 2. A diagram of single point distance estimation.

Pros:

- Very simple to implement

Cons:

- Doesn't account for the car's angle to the wall, thus giving inaccurate distances except when the car is exactly parallel to the wall. As a result, this method only works within a small range of angles where the real distance to the wall has a greater effect on estimated distance than the angle to the wall.
- Dependent on a single lidar point

2. Closest point in range

Estimate the distance to the wall as the distance to the nearest lidar point on the side of the robot facing the wall. Because the lidar points are closely spaced, this value is roughly equal to the true perpendicular distance to the wall. The index of this point in the lidar data array can be used to compute the car's approximate angle to the wall.

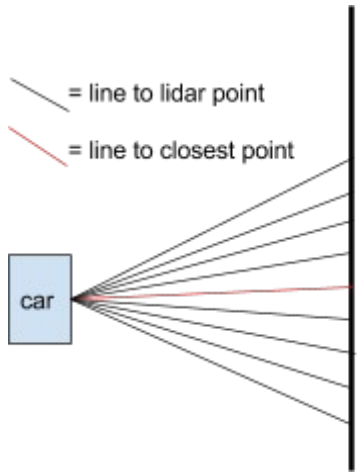

Figure 3. A diagram of nearest point distance estimation.

Pros:

- Fairly simple to implement
- Accounts for the car's angle to the wall

Cons:

- Taking the closest point in a large range makes this method especially vulnerable to noise (although averaging the distances to several of the closest points makes it more robust)
- Gives only approximate angle and distance

3. Two point triangulation

Geometry is used to compute the distance to the wall using two lidar points. Inverse trigonometry can then be used to calculate the car's orientation relative to the wall.

a = angle between points
r1 = distance to first point
r2 = distance to second point

$$D = r1*r2*\sin(a) / \sqrt{(r1^2 + r2^2 - 2*r1*r2*\cos(a))} \quad \text{(Eq. 3)}$$

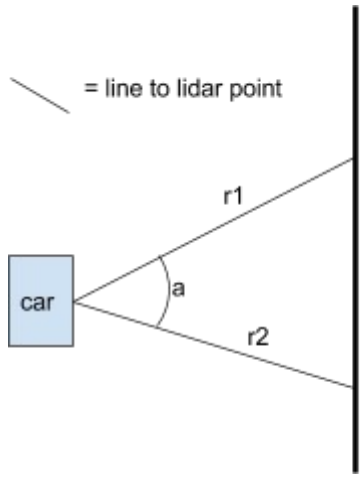

Figure 4. A diagram of two point geometric distance calculation.

Pros:

- Calculates exact angle and distance to wall

Cons:

- Reliant on the assumption of a straight wall
- Based on only two lidar points

4. Linear regression

Compute the regression line of the points on the side of the car where the wall is. Next, find the equation of the line perpendicular to the wall and passing through the car. The distance to the wall is the length of this line between the car and its intersection with the regression line. Inverse trigonometry can be used to find the car's orientation relative to the wall.

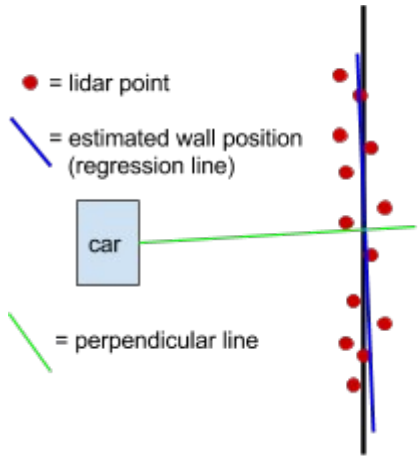

Figure 5. A diagram of linear regression based distance calculation.

Pros:

- Uses all the lidar points which fall on the wall, making it very stable against noise
- Calculates exact angle and distance to wall

Cons:

- Based on the assumption of a straight wall
- Becomes very computationally expensive for large data sets

Results

The single point method worked surprisingly well if the car started nearly parallel to the wall at about the desired distance. If the car started at an angle, turned significantly to reach the desired distance, or was subject to large disturbances, its angle would make the wall appear much further away than it actually was, so the car would turn and crash. The minimum distance estimation was much more robust, reliably giving the correct distance. The latter two methods were implemented in code but not tested due to time constraints. However, they might well be necessary for reliable wall following under more challenging circumstances than the straight, flat, opaque walls we used. We did not test the wall following performance on glass or screens, which would have posed greater lidar processing challenges.

Week 2

Goals/overview

The main goal of the week was to learn about Opencv and image processing. Image processing has great potential because of the availability of light, durable, high resolution, high fps cameras. Cameras can also function over long distances where diffraction and the optical effects of air render lidar unusable. However, image processing is much more computationally expensive than lidar point processing, as structure must be inferred from 2d camera images. Opencv is a python library for efficient image processing, with various filters and masks for finding objects in images. The combination of the car's onboard GPU and efficient Opencv image processing allowed us to perform image recognition using the camera. Our programming objective was to use these capabilities to detect and react to colored signs, something known as visual servoing. The car had to navigate to the sign, which was located along a wall, turn either right or left based on its color, and then follow the wall away from the sign.

Details

We broke the challenge into three steps; sign detection, navigation to the sign, and wall following. For wall following, we reused the code we had written the previous week, which used the second algorithm described due to its comparative simplicity. Navigating to the sign used proportional control, with the error being the distance of the colored sign from the center of the camera image. Reliably detecting the sign was then the main challenge. To represent the camera image, we used the HSV (hue [color], saturation [purity of hue], value [brightness]) color scheme (6). HSV cleanly separates the different properties of lighting, making HSV values much easier to adjust to different objects and lighting conditions than RGB ones. To find the sign, we

first filtered the image by hsv values, creating a binary mask to differentiate areas with hsv values matching those of the sign from the rest of the image. We then used opencv's contour finding to efficiently compute the edges of these regions. The largest such region was taken to be the sign, allowing us to ignore misleading red or green objects such as clothes. As the sign was rectangular, we fit a rectangle to the contour, giving us its dimensions in pixels and its position in the camera image. Knowing the square's real size, the distance to it could be calculated from its height in pixels, the angle of the square could then be calculated from its width in pixels, and the angle of the car could be computed from the position of the center of the square on the camera image. However, because the car started perpendicular to the square and pointing directly at it, only the position of the square in the camera image was required. The extra detail on the car's location that could have been extracted wasn't useful enough to justify the additional complexity. However, it would have been necessary in more complex situations where the car's angle to the sign was variable and important. Proportional control was adequate to control steering, with the integral and derivative terms being unnecessary. The car simply steered to point itself directly at the square while driving forward at a fixed speed. While reaching the sign was relatively straightforward, the curved wall made wall following more challenging. Making sharp turns required more aggressive steering, which in turn required better derivative control to prevent oscillation.

Results

We failed to complete the challenge despite being able to navigate to the square and follow walls. This was a result of our failure to anticipate the difficulty of making the 90 degree turn to align with the wall once we reached the square. Because the curves in the wall started very close to the sign, the wall following didn't have time to stabilize on a section of straight wall before turning. This greatly reduced the margin of error for the turn, and made the wall following code's job even harder. Due to the image detection algorithm's imperfect handling of variability in lighting, the calculated size of the square, and thus the stopping position, varied by run. The combination of this variability, our dead-reckoning based turn, and the small margin of error led to the car failing to follow the wall. A better approach would have used lidar to get a more reliable stopping distance, as well as to make sure our turn ended parallel to the wall. This would have provided much more reliable transitions to wall following. Our failure to implement it was a result of our breakdown of the problem, which underestimated and therefore neglected the difficulty of transitioning between different control modes. This underscores the importance

of considering robotics problems wholistically before trying to break them down into distinct parts.

Week 3

Goals/overview

The week's objective was about navigation problems in two dimensions, and to implement 2d navigation using the racecars. The weekly challenge was to combine this navigation with image recognition to explore and catalogue images. The car operated on an open field, rather than a linear track, with images to detect scattered around.

Details

SLAM (simultaneous localization and mapping) is the basis of robot navigation in complex environments which are not perfectly consistent over time. When navigating towards a specific goal, the robot begins with a global map. It collects sensor data as it moves, and fits this data both to the previous data and to the global map with help from short range odometry data. SLAM allows the robot to determine its location far more accurately than would be possible with odometry alone. The sensor data forms a more detailed and up-to-date map than the global map, and is used to plan the robot's motion over short distances. The global map gives the robot its long range goals and is used to plan its overall path (7). SLAM can also be used without a global map to create a map by stitching together sensor readings. In this setting, loop closing becomes a major challenge, because, like odometry, series of matched sensor readings can gradually drift. We suffered from this while attempting to map a small racecourse, and were thus unable to proceed with using SLAM for navigation.

Due to the complexity of SLAM and the poor odometry capability of the racecars, we did not attempt to use SLAM to navigate. A different method of navigation which is useful for exploration is potential field control. Each object (each lidar point in our case) is treated as a positive charge. Our car is treated as another such charge, and a large positive charge is placed behind it to propel it forward. The net force vector on the car produced by these charges causes it to steer and drive forwards or backwards (3). Because potential field control models objects as electrical charges, the inverse square law causes field strength to asymptotically approach infinity near objects, theoretically rendering crashes impossible. However, the car's physical momentum can cause it to crash in practice, as its lidar scans have finite frequency, and its motor reactions do not occur instantly. Furthermore, unlike a charged particle, the car's motion was nonholonomic, and the motors would stop at speed commands below ~0.3 m/s. As a result,

the car could get stuck in local minima in the potential field when it either couldn't steer sharply enough to escape or moved too slowly in an area with a weak net field.

To improve our potential field navigation, we first tried raising the strength of the repulsive charges to prevent crashes. This worked, but caused the car to stop in narrow but passable areas due to repulsion by the walls. We therefore returned wall repulsion to its original level, but capped the car's speed. This kept it from building up momentum and thus reduced its stopping distance to safe levels, while allowing it to keep moving close to walls and obstacles. However, the car still sometimes got stuck in concave areas of the wall. To remedy this, when the car stopped, we reduced the charge behind it gradually, turning it into a negative charge for a short time to draw the car out of the local minimum. When reversing direction or entering large open spaces, however, the car would accelerate rapidly, causing a velocity spike and sometimes minor crashes. This was a result of the car's internal PID motor controller being too aggressive and overshooting the desired velocity. To prevent acceleration spikes, we limited the maximum acceleration which our code could command the car produce to get smoother and safer driving.

Results

Our experiences with potential field navigation highlighted the need to explicitly account for the differences between the car and a charged particle. Because such modeling is complex and hardware specific, and would likely be time consuming to calibrate, we chose to adjust the potential field control to deal with those differences. However, this indirect method didn't yield optimal results. The car main cost was the low speed required to keep stopping distance short. The car also stopped and backed up from walls it could have avoided altogether with better steering. Our success with capping acceleration demonstrates the potential of modeling low level vehicle software behavior. These issues highlight the need for accurate modeling of the robot hardware's response to the software. The car's turning radius and stopping distance must be taken into account when the software determines steering angle and velocity. For example, as shown below, a considerable area around the car is unreachable at any given time due to the nonholonomic nature of ackermann steering. However, potential field control treats the points in these areas the same as any others, leading the car to stop for objects it could not possibly crash into. Accounting for this alone could considerably improve the ability of potential field control to negotiate narrow passages.

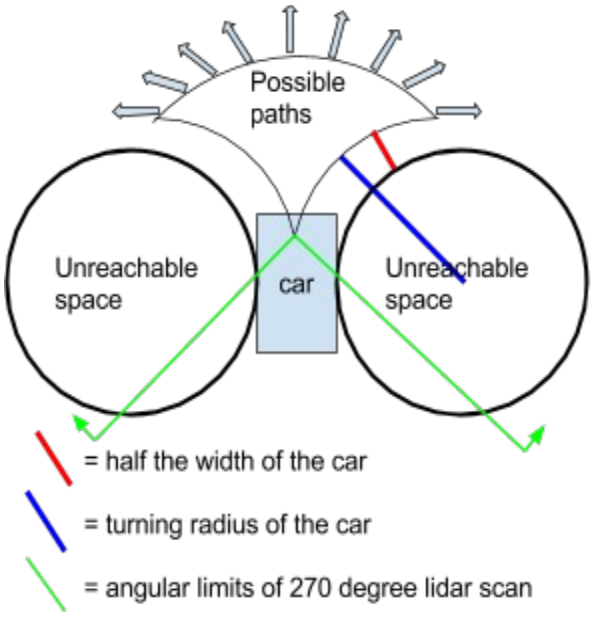

Figure 6. A diagram of the potential paths of an Ackerman steered car in response a single movement command, showing the overlap between unreachable areas and the lidar scan.

Week 4:

Goals/overview

The goals in the final week were centered around the successful completion of the race. The car was supposed to drive rapidly the racecourse without crashing, taking the shortcut when it was available and turning when it was not. The car should also be able to complete the racecourse in the presence of other vehicles. Cars circulate counterclockwise through the racecourse. Only the end of the shortcut is closed during the time trials, and the colored sign is centered above the beginning of the shortcut. In the multi-car rounds, both ends of the shortcut were closed.

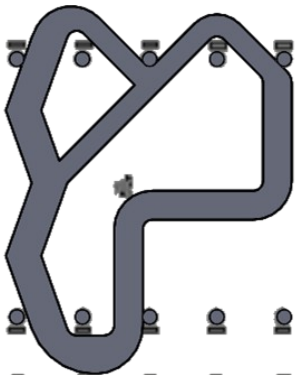

Figure 7. A map of the racecourse. The shortcut is the central straight segment.

Details

We decided to use potential field control to traverse the racecourse because it would be able to navigate effectively in the multi-car rounds where wall following would fail. We did try wall following for the time trial rounds where we were the only car. However, it had a hard time switching sides and then making a sharp turn when the shortcut was closed. Potential field control suffered some similar turning issues because we ran out of time before we could test and calibrate the timing and duration of the virtual charge that was placed in response to detection of the colored sign. However, in the multi-car rounds, the addition of a barrier at the start of the shortcut allowed potential field controller to make the turn successfully.

When testing potential field control at high speeds on the final racecourse, our car would start turning too late and crash into the walls or oscillate violently between them. We therefore used a 1/distance rule instead of a 1/distance^2 rule to generate our potential fields. The 1/distance rule caused the car to turn sooner due to the steeper field gradient farther away from the walls/lidar points. Turning more sharply sooner allowed the overall turns to be less sharp, making the car's path straighter and more stable, and preventing crashes. We then switched to a 1/distance^.5 rule to make this effect even stronger, further improving our driving. This result suggests that the exponent of the distance in inverse potential field equations should be treated as a tunable constant. One could go even further and test a broad range of distance based equations to generate the potential fields in different applications.

Pure potential field control has dynamic speed to model the behavior of actual particles and allow the car to bounce off walls and turn around when exploring. However, this isn't as well suited for racing, as cars will repel off of cars ahead of them rather than accounting for their velocity and moving with them. Furthermore, the car's route through the racecourse is determined only by its steering angle as a function of position. Velocity merely determines how

fast the car executes the route, so the optimal behavior in a race has the car traveling at maximum speed. We therefore set our car to always travel at the maximum 3.0 m/s regardless of the field vectors, relying on our navigation to avoid crashes instead of the slower process of stopping, backing up, and turning.

Regardless of its charge and Kp tuning, the car tended to oscillate between the walls. To dampen these oscillations, we implemented derivative control within potential field control. Because the time derivative of the field vector perpendicular to the car is a function of the car's orientation relative to the track walls, derivative control works directly to control steering oscillation without losing the ability to steer sharply and abruptly on tight turns. This made our potential field control more robust than the basic form, keeping the car stable and well away from walls even when it passed near other cars.

Our potential field control and sign detection involved a number of constants and color values. Some of these are summarized below:

Potential field constants

Quantity	value
Lidar point charge	0.07 (arbitrary unit of charge)
speed	3.0 m/s
Steering p value	1.5 (dimensionless)
Steering d value	0.2 (dimensionless)

HSV values (for green)

Quantity	Range
hue	54-72
saturation	110-180
value	10-200

Results

We did poorly in the time trials, with only one of the three runs completed, due to our difficulties with turns. This was a result of our lack of time to tune the color response turn for when the shortcut was closed, as both potential field navigation and color detection worked reliably in isolation. This resembles the problem we had switching between control modes during the second week. However, our well developed, constant speed potential field navigation worked very well during the group races when turning was less of an issue. These results demonstrated the capabilities of potential field navigation but reinforced the need for better function integration.

Technical conclusion

Our experience in the third and fourth weeks enabled us to develop potential field navigation to a high degree, but also taught us some of its limitations. Potential field control works quite well on a linear course, but doesn't produce an optimal path. This because it fails to use information about the large scale geometry of the course to take the most efficient possible routes. These routes typically involve a combination of hugging the walls in some areas and moving diagonally across the lane to switch walls in others. Potential field control, on the other hand, tries to stay centered in its lane based on the local racecourse geometry, thus taking longer routes than necessary around corners. A more serious issue than suboptimal routing is the inability of potential field control to handle forks well. A car entering a three way fork has two viable exit routes. However, this often produces a symmetrical field, causing the car to continue straight and crash into the wall halfway between the two forks. Furthermore, when exploring, potential field control wanders aimlessly, often moving only between or within a few large open areas. It will rarely or never enter smaller spaces even if it can fit. Because it of this, it is not ideal for thorough exploration of complex spaces. These limitations stem from the local nature of our potential field control, and they are among the issues that SLAM seeks to address by giving the car global goals. Within SLAM navigation, path planning algorithms such as rapidly exploring random trees (RRT) look for the shortest viable complete path to a goal rather than responding only to local features (3). This avoids both potential field control's bias against narrow but usable routes and its tendency to get stuck in dead end paths. More generally, my experience at Beaver Works, particularly in weeks two and four, helped to warn me against excessive division of problems. While dividing robotics problems allowed us to solve parts of

them efficiently, modular solutions were useless without plans to integrate them effectively into a single program.

Personal Reflection

Participating in the Beaver Works summer program taught me a lot about teamwork and team organization. Working on hard robotics programs showed me the necessity of working in a team and trusting one another to write reliable code without knowing every detail of it. Given the limited time that we had, no one person could have written as good a body of code as each team did. Trusting others to develop reliable code for their functions so that you can focus on writing good code in a single area is necessary to effectively address complex problems. However, our problems with code integration also taught us that it is important to have someone focus on the interactions between different pieces of code. In some cases this means planning input and output of ROS nodes, while in others it means creating new pieces of code to handle the transition between larger ones. We in some cases put too many people on a few pieces of code, rather than splitting the team up further so people had the change keep an eye on the overall project. In the final week, the decision to pursue both wall following and potential field navigation spread the team too thin, resulting in nonfunctional wall following and inadequately tested potential field code. A more cohesive team attitude would have helped us focus our energy on only one solution, resulting in better race performance. Finally, keeping a calm and positive attitude even when you are short on time is important. It both helps the team run smoothly and lets you continue to work efficiently on the project.

Works Cited

1. Karaman, Sertac. *Beaver Works racecar*. Png file, 2015. A photograph of one of the cars used in the course.
2. “Injury Prevention and Control: Motor Vehicle Safety.” *CDC*, Centers for Disease Control and Prevention, 29 Dec. 2015, www.cdc.gov/motorvehiclesafety/costs/index.html. Accessed 18 Aug. 2016. Information on car accident injuries and deaths.
3. Karaman, Sertac. “Basic Autonomy: Planning and Control.” *Index of racecar lectures 2016*, Massachusetts Institute of Technology, peris.mit.edu/racecar/lectures/2016/Lecture3.pdf. A source covering potential field control, RRT path planning, and google self driving cars.
4. “Continental Replaces Exterior and Interior Mirrors with an Innovative Camera Monitor System.” *Continental*, 1 July 2015, www.continental-corporation.com/www/pressportal_us_en/themes/press-releases-us/3_automotive_group/chassis_safety/pr_2015_07_01_mirror_replacement_en.html. Accessed 18 Aug. 2016. Information on autonomous subsystems to assist drivers.
5. Boulet, Michael T. “Introduction to ROS.” PDF file, 12 July 2016. An introduction to ROS nodes, topics, and messages.
6. Detry, Renaud. “Image Processing.” PDF file, 20 July 2016. A source on image processing and HSV color schemes.
7. Boulet, Michael T. “Localization.” PDF file, 29 July 2016. A presentation on SLAM.