UM::Autonomy's "Eve", A Highly Proficient ASV

Michelle Howard, Dominique Kudzia,
Nick Ashcroft, Jakob Hoellerbauer,
Cyrus Anderson, Alex Berman, Steve Ratkowiak

University of Michigan
Ann Arbor, MI

Figure 1: Eve

ABSTRACT

Eve is a fully autonomous surface vehicle built for the purpose of competing in the sixth annual AUVSI RoboBoat Competition, where it will showcase its ability to complete the buoy course and challenge stations. The vehicle uses a pontoon design for maximum stability, which is essential to all of the electronic systems on board. The main focus for the team this year was to redesign the hulls and deck, as well as outfit the boat to complete the challenge stations that are new from last year. This paper will discuss the modifications we have made to our boat since the competition last year.

1. INTRODUCTION

UM::Autonomy presents our autonomous surface vehicle, Eve, for entry in the sixth annual AUVSI RoboBoat Competition. Eve is a fully autonomous vessel, designed to complete the buoy course and challenge stations of the 2013 competition. The design of Eve was heavily influenced by her predecessor, Serenity, which was the winner of the 2012 RoboBoat competition. Our main focus for 2013 was to build off our success from the previous year and to address the issues Serenity faced. Since Eve is a new iteration of Serenity, we will not address every feature of the boat but rather the improvements that have been made for 2013,

While Eve may look similar to Serenity, the boat has been almost entirely rebuilt. Due to our success from the previous years, we are confident Eve will complete a buoy course, so we spent the majority of this past year focusing on the challenge stations. We have new sensors, new systems for attempting the challenges, and a new propulsion configuration for improved maneuverability. .

This paper will discuss all of the developments we have made in the past year, as well as a brief overview of aspects of the system that have remained the same. With all of the changes we have made, we

are excited to present Eve to this year's RoboBoat Competition.

2. HULLS AND DECK

2.1 Hull Design

The final hull design was selected for its success in exceeding all of our imposed design requirements. In 2012, the hulls were unstable when quick transitions from forward to reverse motion occurred; this has been resolved in 2013 as a result of the redesign. Notable design changes in 2013 include a new hull profile, the use of an internal aluminum support structure, new thruster placement, and the interface between the hulls and the deck.

The hulls were designed to be streamlined to help with tracking and reduce water flow separation from the hull. Inspiration for hull design was drawn from pontoon boats, which have low waterplane areas and can easily transition from forward to reverse motion. Like its predecessor, the forward sections of the hulls are semi-planing; unlike its predecessor, the aft section is semi-planing as well. As a result, the boat travels well in both directions of travel. The hulls are very stable in pitch, roll, and yaw and thus provide a stable platform for the vehicle's electronic systems.

Four 80W Seabotix thrusters are used to power the boat and are mounted underneath the hulls at 45˚ normal to the boat. This thruster position was chosen due to an issue in 2012 where the thrusters breached the water surface, causing hardware on the thruster to become brittle and fracture. In addition, the new configuration improves the overall maneuverability of the boat and allows the boat to move laterally in addition to the traditional forward and reverse motion.

The 2013 hulls also feature an aluminum support structure that is used to anchor the hulls to the boat deck. A major design change is that the hulls do not protrude through the deck at the interface; rather, they are held to the deck using bolts. This resulted in more usable deck space and easier installation of the hulls. The support structure also adds mass to the hulls, resulting in greater stability.

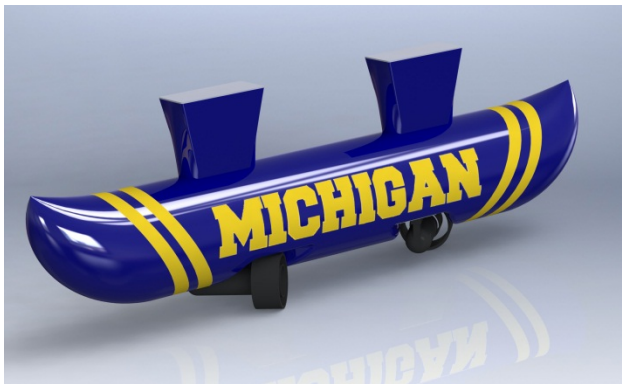

Figure 2: SolidWorks model of the 2013 hulls

2.2 Hull Fabrication

The hulls were fabricated using a male mold fiberglass process. Computer models were created using the modeling programs SolidWorks and Catia, and then four foam half-hull molds were cut using a CNC Router. Sections were grooved out within the hull to accommodate the support structures, which were fabricated from aluminum stock and then TIG welded. After the support structures were embedded in the hulls, the half-hulls were joined and a layer of fiberglass was added over the top.

An epoxy layer was added to protect the fiberglass layer as well as waterproof the hulls. Silicone was used to waterproof the interfaces between the supports and the foam molds. The paint scheme was updated for 2013 and then an additional coat of epoxy was added to waterproof and protect the paint.

Figure 3: The foam molds for the hulls were fabricated using a CNC Router

2.3 Deck Design

The deck of the vessel is nearly identical to the 2012 design, using the same lightweight aluminum-honeycomb material. The deck's dimensions (28" x 48") were changed to allow for easy transfer through doorways. Access holes were added near the electrical box so that cables could be routed beneath the deck for a cleaner appearance.

3. ELECTRICAL SYSTEM

3.1 Design

This year we are using the same electrical system as our last year's entry, Serenity. One of the most notable changes to our electrical system was the addition of two new motor controllers to support our new propulsion system.

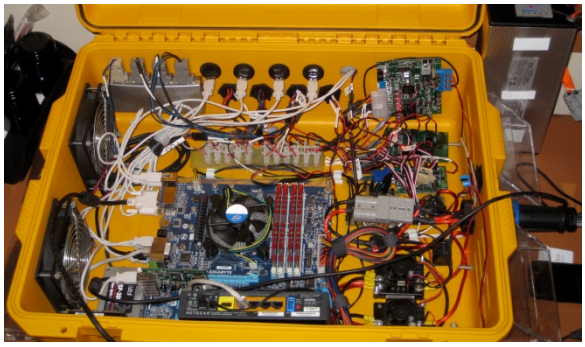

Figure 4: Eve's Electrical Box

3.2 Sensors and Servos

For dead reckoning purposes we rely on a GPS, a Compass, and a Fiber-Optic Gyroscope (FOG):

GPS: Garmin 16-HVS

Used primarily for velocity measurements to project the state forward.

Compass: Ocean Server OS5000

Used primarily for pitch and roll measurements and to compute an initial heading for correlating GPS and Gyro measurements in our Extended Kalman-Filter SLAM.

FOG: KVH DSP-3000

Used exclusively for yaw measurements as its superior drift rate of less than 1 degree per hour provides us with nearly perfect yaw estimation.

For perceptual purposes we use the following sensors (precise usages are explained in detail in Section 4):

Cameras: Point Grey Flea 3 Firewire

LIDAR: Hokuyo UTM-30LX LIDAR actuated using a Dynamixel AX12 Servo.

Underwater Camera: VIVIDIA USB Flexible Inspection Camera Borescope & Endoscope for underwater detection.

We have also added several servos to make our ASV much more capable. These include:

Two Dynamixel AX12 Servos for a pan/tilt dart gun aiming system.

Two Dynamixel AX12 Servos for deploying a ramp and tethering an amphibious RC car.

4. VISION SYSTEM

Like last year's entry, Eve uses two cameras and a LIDAR device to see her surroundings. Where practical, it tries to match objects detected in images with those detected in LIDAR point clouds to gain additional certainty about their location and existence. In other instances, Eve uses only the images from each camera separately to locate objects in 3D space. Her approach is camera-driven, but she also heavily relies on spatial 3D information.

This year we added a third, more specialized camera, whose data is not matched with LIDAR. It sits underwater, and is used to detect an underwater buoy that will be at one of this year's challenge stations.

4.1 Detection from Camera Images

We begin our above-water buoy detection by performing a HSV-based blob detection of each camera image. This process includes several filters and thresholds to reduce false detection rates. However, this alone is not enough to ensure reliable buoy detection. There are numerous situations that can cause false detections, such as lighting conditions, vegetation on the shore, and pedestrians.

For the challenge stations, we have employed a variety of techniques including HSV thresholding, blob dilation, shape fitting, and other constraints. These will be further explained in the corresponding sections.

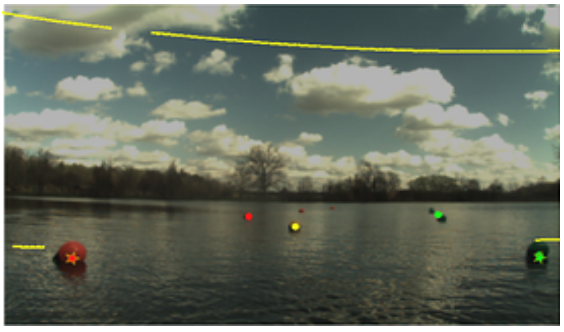

Figure 5: Blob Detection is used to detect buoys during a practice run

4.2 Point Cloud Generation

For spatial data we utilize a 3D Lidar-based system. Our Hokuyo UTM-30LX is actuated by a Dynamixel AX12. Using returns from both the servo and laser, we can construct a set of three dimensional range points, which we refer to as the point cloud. For a detailed explanation of point cloud construction see the corresponding sections in our competition paper for Wolvemarine (2011).

For Eve, we made many decisions similar to last year to maximize utility from our point cloud data. First, we mounted the Lidar beneath the deck. This was done to eliminate returns from debris in the water and reduce the required rotation angle for the buoy channel. Second, we implemented a dynamic servo control system that allows the route planner to reconfigure the rotation range during a run. This allows us to use a very small range for the buoy channel without sacrificing our ability to generate point clouds of challenge station tasks. Eve can produce buoy channel point clouds, and thus full image-point-cloud pairs, at up to 8 Hz.

4.3 Calibration

Before data correlation can be performed, we need to obtain an accurate calibration between the laser and camera. To do this, we gather a large set of image/laser pairs that can be used to compute both camera intrinsics and the rigid-body transform between the sensors. We use the extrinsics model given in [5] and the parameters are computed using the method described in [3].

Figure 6: Calibration of the laser and cameras

5. FEATURE DETECTION

5.1 Buoy Detection

For buoy detection we perform blob and point cloud correlation. By correlating blobs with transformed laser points within a reasonable radius and of reasonable quantity, one can develop a workable buoy detector. However, experimentally, this simple approach fails whenever the shoreline is near. When this occurs, there are a plethora of laser points that will easily be correlated with blobs. From experiment, there tend to be many false detection blobs for the on-shore vegetation.

To reduce this vulnerability, we filter the point cloud data to discard large clusters of nearby points. We accomplish this goal using the Union-Find data-structure. We begin by reducing all points to an X-Y grid map. We can then join nearby points to form several clusters. Next we can apply constraints on each cluster, such as size, to discard unlikely candidates. Clusters that are not discarded in the filter are then correlated with the image-based blobs as before. Because our laser range-finder doesn't return off the water surface, large clusters correspond to the shore, medium clusters correspond to buoys, and small clusters correspond to either noise or to debris in the water. By discarding all large and small clusters, we can be reasonably confident in the resulting correlation.

5.2 Catch the Ball

The detection system we will use to find the ramp is the same as last year, which is illustrated in Figure 7. This consists of using our cameras to detect the pink duct tape along the edges of the ramp as well as our laser to find a planar object. From this the boat will calculate the best method of approach and drive up to the ramp and deploy our autonomous land car.

After Eve has docked with the ramp, she will deploy her drawbridge and use Bluetooth to activate the land car. The car is entirely autonomous and will return to Eve when it has retrieved the puck.

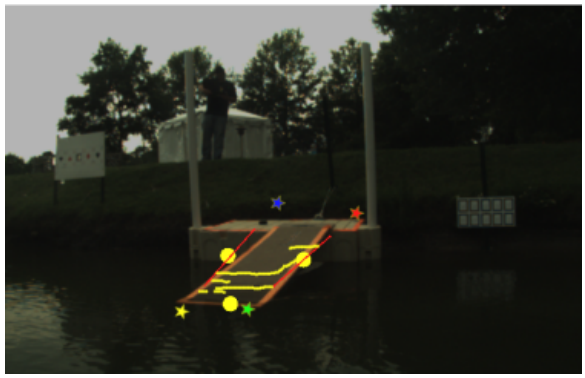

Figure 7: Ramp Detection

5.3 Sneaky Sprinkler

For this challenge we will use the same detection methods as last year. We begin detection by finding all the red blobs in the image and then we filter the blobs by size and circularity constraints. Blobs that are of the same size and approximately the same height above the water are matched and classified as buttons. The vessel investigates the buttons and searches for the submerged buoy using our underwater camera. and a simple white blob detection algorithm. Once the boat identifies the white buoy, it approaches the appropriate button and pushes it using the deployment ramp.

5.4 Rock, Paper, Scissors, Lizard, Spock

This year, our strategy is to detect the signs using blob detection and color matching, similar to the Sneaky Sprinkler challenge. As each sign is a different color, we can determine what the sign is based on the color. One difference from last year's boat, Serenity, is that Eve does not contain an Infrared camera. We attempted the Hot Suite challenge in 2012 and the camera was not capable of detecting a difference in the temperatures of the signs. This was due to the fact that the signs were made out of a reflective material that reflected the heat of the sun. Therefore, this year we will make a random guess as to which is the hottest sign, and will then determine the "hand" that beats it.

5.5 Capture the Flag

For this challenge we will use blob detection and have it look for purple in the same way we use blob detection to navigate through the buoy course. We will treat the boat the same way as we would treat a buoy, but in this case we will attempt to hit the target instead of avoid it.

5.6 Shoot Through the Hoops

The hoops are detected using each of our two primary cameras separately. For each camera and each frame, we first detect blobs of certain HSV values and sizes. Then we fit a circle to each blob, and use multiple rules to ignore certain blobs. We also eliminate a blob if the standard deviation of the distances of the blob's points from its circle is too large. After these eliminations, we have a set of up to three blobs and fitted circles that we believe correspond to hoops. We use the centers and radii of these circles in the image to estimate where the hoops are in 3D space, relative to Eve, and send our estimates to SLAM.

6. SLAM

Eve employs an Extended Kalman-Filter SLAM feature based mapping system [4]. Since there are few things to localize on in the competition pond, we use a feature-based mapper and fall back on dead-reckoning when feature detections are not available for localization purposes. Due to the inclusion of the highly accurate Fiber Optic Gyro (FOG), we now use the FOG returns exclusively for heading information. On initialization, we collect a set of compass and fog observations and compute a "globalization" constant for the FOG measurements. This allows us to use both GPS and FOG data in a global frame for the purpose of dead reckoning.

For building data correspondences between buoy detections, we use the recursive Joint-Compatibility Branch and Bound algorithm [2]. Since challenge station features are unique, there is a known correspondence and data association is trivial.

We have also added some capabilities for map correction. These include: detection and elimination of duplicate features and detection of map corruption. By maintaining a good map of the competition environment, we are afforded many advantages that are not possible with a simple short-term mapping technique, namely, adaptive buoy channel navigation. This will be discussed at length in the following section.

7. ROUTE PLANNING

7.1 Buoy Channel Planning

Due to our success with the buoy channel at last year's competition, we decided to use the same buoy channel navigation method. The system maps each buoy independently and then classifies buoys into pairs based on their distance and orientation. With each new buoy that is detected the system recalculates the best

route based on all past buoys. This is done so that even if the boat makes an incorrect decision it is able to recover.

7.2 Challenge Task Planning

7.2.1 Catch the Ball

When Eve reaches the Amphibious Station ramp, the docking sequence begins. This phase results in lowering the deployment ramp mounted on the front of our ASV.. Using the load feedback from the ramp actuator, we can detect when it has been lowered. . For ease of deployment, the amphibious car is mounted on the edge of the ramp; when the ramp is fully extended the car moves off the boat and onto the challenge station.

Our amphibious landing vehicle is autonomously controlled by an Arduino microcontroller housed in a small water-resistant electric box on top of the car. The car, named WallE, uses two Pololu 1447 motors for rear-wheel drive. It slides across most surfaces using two skis in front that pivot to allow the car to move across inclined planes. The car drives forward a set distance measured by the encoders on the motors. It then turns left ninety degrees and follows a grid pattern to find the puck. We attached large amounts of hook-side Velcro between the skis, which causes the puck to stick to WallE when the car runs into it. WallE has two TCS230 color sensors on the front facing the ground, which are used to detect the pink tape marking the edges of the platform. When it encounters the tape, the car turns around and then continues to travel on a parallel path until it sees pink again or its time limit is exceeded. All this time it knows its relative Cartesian position to the dock from the motor encoder data. After a set amount of time, WallE will travel perpendicular to the ramp by the amount recorded from the motor encoders, and then travel back onto the ramp using the encoder data in that direction. An actuated tether line has been attached to WallE in case it fails to properly return to the ramp; regardless of success or failure, Eve can easily pull the vehicle back to the 'mothership' and continue her voyage.

7.2.2 Sneaky Sprinkler

Since GPS coordinates are provided for all stations, Eve will proceed from the center of the challenge station cove to the given GPS coordinate of the Sneaky Sprinkler challenge until she is able to detect the buttons. At that point, she will investigate each button to find the underwater buoy and then press the button using the front of the deployment ramp. As Eve makes detections of the buttons, she will add them to her SLAM map so that she can localize off of them later, similar to the buoy course.

7.2.3 Rock, Paper, Scissors, Lizard, Spock

For the Rock, Paper, Scissors, Lizard, Spock challenge we plan to approach the station using the provided GPS coordinate. When Eve is approximately 2 meters away from the challenge station we will activate our detection algorithm, discussed in section 5.4. Eve will then navigate towards the signs until she can see them clearly and decide which gesture to play. At that point she will send a ping to the RoboBoat network containing the GPS coordinate and the gesture she wishes to play.

7.2.4 Capture the Flag

Using the provided GPS coordinate, Eve will navigate in circles around the GPS point until she detects a purple blob using the blob detection algorithm. Once the small boat is detected Eve will approach the boat head on until the flag comes into contact with the Velcro on the deployment ramp. A contact switch will be used to determine when the ramp has hit the flag, at which

point Eve will return to the center of the challenge station cove.

7.2.5 Shoot Through the Hoop

When Eve is looking for challenges, she runs our hoop-detecting algorithm. If SLAM becomes certain enough about the location of some set of hoops, Eve will attempt to shoot through them. The gun has the ability to aim up and down, and side to side, independently of the boat. When shooting, Eve continuously sends commands to the gun with best estimates of how the gun's servo motors should be oriented, and the servos continuously adjust to try to reach those estimates. For each shot, the boat will try to minimize her distance from the hoop, and shoot from a position from which the hoop looks the most circular. The gun fires only when Eve believes she and the gun are in a good enough position and orientation.

7.3 Obstacle Avoidance

By delaying obstacle avoidance to a later planning stage we are able to significantly simplify planning. We allow the individual planning systems to command any desired waypoint with little concern about how to reach the location.
Our obstacle avoidance system then constructs an occupancy grid map. This map is dilated to allow the boat to be treated as a single point and then the Wave front algorithm is used to compute the shortest path. We then select a fixed point on the path as a look-ahead waypoint and we order the low-level navigation systems to this location.

8. UTILITIES

We use a variety of utilities to improve software development and make our entire software stack work together in harmony. These include a network message passing system, process manager, visualization library, and special calibration application. We discuss some of these tools below.

8.1 LCM: Lightweight Communications and Marshaling

LCM is a message passing system developed by MIT for the DARPA Urban Challenge in 2007 [1]. This system is specially designed for low-latency communications by using a simple UDP based communication scheme. By defining specialized message types, bindings for C/C++, Java, and Python can be automatically generated and all messages can be automatically marshaled. This allows us to write high-performance code in C and take advantage of Java for high-level code and for visualizations. LCM also provides logging and playback capabilities which have proven to be invaluable for debugging.

8.2 Bot-Procman: Process Management

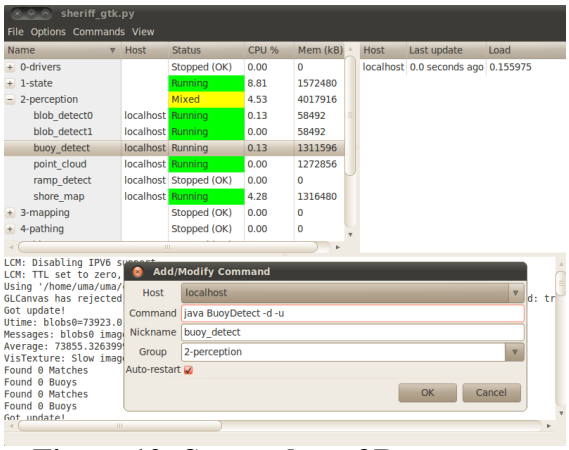

Figure 10: Screenshot of Bot-procman

One of the large failures in the execution of Wolvemarine in 2011 was difficulty in launching due to the high number of processes that were required for our entire system to work. We utilized a process manager that was developed in-house, but was still a bit light on features and buggy. In addition, the process manager was a late addition to the design and had poor overall integration. This year we've

moved to the well-tested and proven Bot-procman that was developed alongside LCM for the DARPA Urban Challenge. This process manager has many advantages such as: the usage of LCM as a backbone to manage processes across multiple machines, the ability to group processes for greater organization, and well-polished interface.

8.3 Vis: Visualization Library

One of the most crucial aspects in autonomous robotics is having a clear understanding of how your system is reacting to the environment and why. With only a simple "printf-style" debugging scheme, this goal is difficult if at all possible. Thus, developing versatile and intuitive visualizations is perhaps the most useful tool a roboticist can employ. For this purpose we use Vis, a 3D visualization library developed by the University of Michigan April Labortory (april.eecs.umich.edu). Vis is a Java library built around OpenGL that allows the user to construct very useful visual-8.

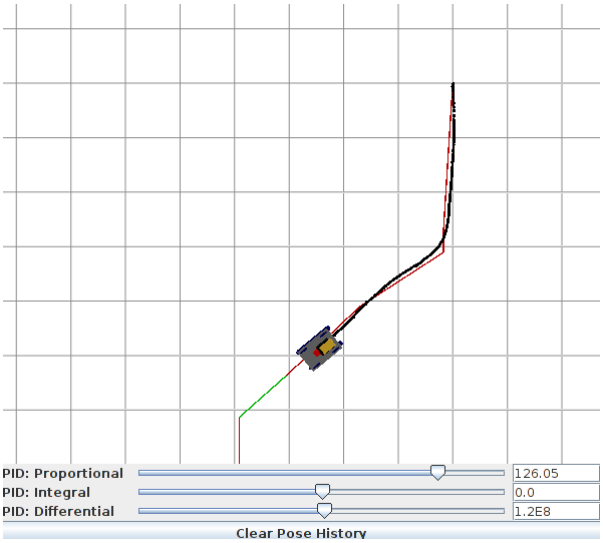

Figure 11: A screenshot of our PID tuning application. The desired path is red while the actual path is black. Note the sliders at the bottom which allow PID terms to be adjusted in real-time.

8.4 PID Tuning

One of the processes that prove frustrating to most developers is the process of PID tuning. Some part of these frustrations stem from the uncertainty about how the PID system is performing. It can be incredibly challenging to decide which PID term to adjust without having appropriate ground truth to compare against. To help ease PID tuning, we have developed a special application to assist us. Our application allows the user to draw a poly-line of the desired path for the ASV to travel, and then it plots the actual path in real time. The application also allows the user to adjust PID terms within the application itself. These features enable quick PID tuning and allow the user to compare different settings quickly and effectively.

9. CONCLUSION

This year, we began with a goal to build on our success in the previous year. Although Serenity won the competition in 2012, we knew that there were many improvements that could be made to the vessel. Our vision was to improve the rough areas of Serenity, and redesign and reinvent where necessary.

We worked to improve the system developed in 2012 to make it even easier to deploy and debug in 2013. New materials and manufacturing processes were integrated into the mechanical systems this year to create a lighter and faster vessel. We emphasized rigorous testing of the new vessel and its components to ensure continued success this year. We believe that Eve is the best RoboBoat entry to date and eagerly await the opportunity to showcase her at RoboBoat 2013.

10. REFERENCES

[1] A. Huang, E. Olson, and D. Moore. *LCM: Lightweight communications and marshalling*. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), October 2010.

[2] J. Neira and J. Tardos. *Data association in stochastic mapping using the joint compatibility test*. Robotics and Automation, IEEE Transactions on, 17(6):890 –897, December 2001.

[3] G. Pandey, J. McBride, S. Savarese, and R. Eustice. *Extrinsic calibration of a 3d laser scanner and an omnidirectional camera*. In 7th IFAC Symposium on Intelligent Autonomous Vehicles, volume 7, Leece, Italy, 2010.

[4] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[5] R. Unnikrishnan and M. Hebert. *Fast extrinsic calibration of a laser rangefinder to a camera.* Technical Report.