**Beaver Works Summer Program Technical Paper**

**Alexandra Shen**

**July 10th- August 6th 2016**

**Abstract**

*During this MIT summer program, I was introduced to a new yet highly rigorous area of technology that applied knowledge of software, hardware, and middleware engineering and mathematics. At the conceptual level, we were to understand the components that served the following concerns: Communication, Computation, Configuration, and Coordination. We were to also value real time aspects at the robot’s system level as we had to prioritize internal components necessary for the racecar to operate efficiently as well as effectively.*

*On the contrary to the technical aspect of this course, ideas of communication and teamwork were highly stressed. Values like making oneself slightly vulnerable but still objective or modest yet still important were highlighted in order to ensure that the initial feelings of Imposter Syndrome would simmer down.*

*This paper will elaborate on both the events that took place, their procedure to achieve the end result, and the lessons learned from each circumstance.*

1. **Project Overview [1]**

Beaver Works Summer Institute at MIT is a rigorous four-week summer program that provides forty-five high school students the opportunity to receive a better understanding of what it takes to be an engineer/scientist. The selected students start working from nine in the morning to around five in the evening every day in a college-like classroom. They have lectures about topics pertaining to autonomous vehicles, the basis of this program as well as implementing their acquired knowledge on robotic race cars.

Furthermore, students also have seminars spoken by members from large engineering-based companies, like Nasa’s Jet Propulsion Laboratory or Amazon, that apply similar concepts the students will and have learned. During the program, the students use a Robot Operating System (ROS) that has been installed from a Virtual Machine (VM) in order to develop a software for the autonomous racecar. Therefore, they are able to expand their knowledge and skills in coding using the Python coding language while implementing that to these different software platforms.

[1] MIT Beaver Works Summer Institute. *20160718\_MIT\_Beaver\_Works\_Summer\_Program\_Overview.pptx.* Boston, MA: MIT Lincoln Laboratory, July 2016.

1. **Weekly Technical Learnings**

**Week 1: Drag Racing**

During the first week, students received a crash course lecture by Michael Boulet, on the Robot Operating System (ROS) and the Linux platform (another open-source operating system), their processes and their command lines. ROS employs a modular, peer-to-peer architecture. Students were also given an introduction to the physical hardware of the race car, lectured by Owen Guldner, so that they would have a clearer understanding of where and what every component does in relation to its software counterpart.

Figure 1. Guldner, Owen. “Component Breakdown.” Digital image. Introduction to the RACECAR Platform.pdf.July 11, 2016.

We also learned about Bang Bang Controller and PID controller, whilst comparing the two algorithms. In a nutshell, a bang-bang controller is a controller that outputs only two discrete commands. For example, let’s say you can set a desired room temperature, **T,** on an electric heater. If the actual room temperature is below **T**, then the heater will go full power to heat it up. If it is above **T**, the heater will not do anything but wait for the room to cool down. So this controller can only switch between two states: "full power" and "no power at all".

Thanks to their simple design, bang-bang controllers are common in high inertia systems which do not need to achieve great performance or accuracy (like a thermostat).   
But because of the discontinuity between the two states, they can sometimes lead to the undesired Zeno effect, which has to be taken care of using, for instance, sliding mode control.

Proportional-Integral-Derivative (PID) control is the most common control algorithm used in industrial control. The popularity of PID controllers can be attributed partly to their robust performance in a wide range of operating conditions and partly to their functional simplicity, which allows engineers to operate them in a simple, straightforward manner. As the name suggests, PID algorithm consists of three basic coefficients; proportional, integral and derivative which are varied to get the optimal response.

Imagine you are driving a car, trying to reach and maintain speed of 50 kilometers per hour. You watch the difference (error) between your speed and the desired speed. You also press more or less on the gas pedal (control output).

The P value (proportional) can be viewed as the less you have, the more you push. The farther you are from the desired speed, the more you must press the gas pedal; the closer you are, the less you press it. This works well, but there is a small accuracy problem: when you get at the desired speed, according to this rule, you let go of the gas pedal completely. The end result is that your car slows down and stays a little below the desired speed of 50 (static error). Proportional control is the main ingredient of any control and may be slightly inaccurate.

The I value (integral), can be thought of as waiting a little, and if there is no improvement, you push a little more. If you are stuck below the desired speed for a long period of time without any progress, you push the gas pedal with some more force. If you still do not make it to the desired speed for some time, you push the pedal a little further down. Once you get to the desired speed you leave the pedal where it is. Integral control gives you accuracy (zero static error) but you have to wait.

The D value (derivative) can be seen as the reaction to sudden changes. A strong gust of wind pushes your car. Suddenly, your speed surges upward toward 50. Startled, you release the gas pedal. As the speed surge ends and then stabilizes, you return the pedal to where it was. Derivative control manages sudden surges and may prevent overshooting your target speed.

The main task of the week was to “Move” by producing a code that would make the race car achieve basic motion control and simple object avoidance. [1] The goals for the challenge of the week was to develop a code that would allow the car to go in a straight line under a short amount of time.

However, before jumping straight into coding for the week’s challenge, we needed to learn and get comfortable with our platform. First off, we have to know how to connect our racecars over SSH (Secure Socket Shell), a command that provides administrators with a secure way to access a remote computer/electronic vehicle. We also had to know how to understand and create nodes, messages, and topics. Nodes are processes that perform the computation in the ROS system; they are the vertices in the ROS computational graph. Messages are packets of data sent between ROS nodes. Topics name the unidirectional communication links between ROS nodes; they form edges in the ROS computational graph. One or more nodes may publish messages to a topic and, subsequently, one or more nodes may subscribe to messages on a topic.[2]

A personal technical goal was to have a fuller and richer understanding of ROS and how it pertains to the knowledge I already have in the Python and HTML languages. Self-teaching became a mandatory necessity, a process that can be tedious and extremely frustrating, but oddly satisfying once it is conquered. And even though our team was disqualified for crashing into walls and consequently going in a circular path, a valuable lesson learned this week was to *listen* to the other members of one’s team rather than simply *hear* them.

[2] Boulet, Michael T. *Introduction to ROS.pdf*. Boston, MA: MIT Lincoln Laboratory, July 12, 2016.

**Week 2: Wall Follow, Blob Detection, Using the ZED Camera**

Altogether, the main goal for Week 2 was to develop a code that would enable the racecar to make the correct turn. We were to “Explore” using vision based blob detection, targeting, and object detection. [1] Students are expected to understand previous algorithms and implement that understanding into a code such that it would detect color spacing, segmentation, blob detection, learn new commands in ROS, and understanding and using the ZED Camera on the racecar. We received a lecture on Image Processing from Renaud Detry as well as another lecture on Computer Vision. We also learned how to create custom messages and use the tele operating function and the turtle simulator. Using the ZED Camera became an important factor to Week 2’s challenge.

Image Processing focused on light and color. We focused on the primary colors—red, green, blue—whilst comparing it to its intuitive representation, Hue-Saturation-Value (HSV).

Feature Extraction and Bag-of-feature Classification is a topic under Computer Vision. It is an information concentration step that reduces the data rate from 106—108 bytes s-1 at the output of a camera to something of the order of tens of features (vectors of a few dozen scalars) per frame that can be used as input to a robot’s control system. [3]



Figure 2. Detry, Renaud. “The Spectral Representation of Light, Color, and RGB.” Digital image. Spring 2011. Accessed July 20, 2016.

Figure 3. Detry, Renaud. “Hue-Saturation-Value: an Intuitive Representation of RGB.” Digital image. Spring 2011. Accessed July 20, 2016.

Fitting and Shape Matching is also another aspect of Computer Vision. Fitting usually refers to fitting a model to a set of points, i.e., finding instances of the model in the set of points. We were introduced to principles like the Hough Transform, that makes each data point vote for all the model instances that could pass through it, and select the instance that collects the most votes. We learned about the RANSAC (Random Sample Consensus) Algorithm, a stochastic algorithm for fitting a model to a data set that contains outliers. If we know that 50% of points are outliers, and we fit the model to random pairs of points, 25% of these pairs will yield a satisfactory model instance. Iterative Closest Point(ICP) is part of the Shape Matching portion of the lecture that registers two 3-dimensional point clouds, i.e., finding the 3D transformation that maps the first point cloud to the second one. ICP always converges to a local minimum of *E*, but there is not guarantee that ICP will converge to the global optimum. [4]

My motive for this week was to further my understanding on ROS and its commands, develop my knowledge in Python as well as Linux, and find a way to bind the three platforms together so that the program would be successful to use for the end of the week race. Our team completed the most part of the week’s challenge. We were able to detect the red and green papers but for both times, the racecar went to the right; it was due to some minor coding issues.

[3] Detry, Renaud. *Feature Extraction, Bag-of-feature Classification.pdf*. Liege, Belgium: University of Liege, Belgium, February 25, 2014.

[4] Detry, Renaud. *Fitting and Shape Matching.pdf*. Liege, Belgium: University of Liege, Belgium, March 31, 2015.

**Week 3: Mapping using SLAM, Potential Field Maps, Bug Algorithm, Localization**

This week, our focus was to achieve the unit of “Learning” how to map, localize, and road network navigate. [1] Students were to produce a map in ROS of the racecar’s path by using the concept of Simultaneous Localization and Mapping (SLAM). They were to also learn alternative methods of localization like the Bug Algorithm or Potential Field Mapping so that they can be compared to other processes to understand the pros and cons of each one.

Simultaneous Localization and Mapping (SLAM) is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent’s location within it.

Figure 4. “Overall Process of Data Association.” Digital image. Hindawi Publishing Company. August 25, 2011. http://www.hindawi.com/journals/jr/2011/257852.fig.006.jpg

The goal for this week was to explore space and find colored blobs such that the racecar would be able to avoid obstacle around the race course. The teams also have a code that would keep track of the colored blobs.

With the higher understanding of ROS and Python, I tried to help my teammates create as well as edit trigonometric algorithms that were eventually used detect the week’s criteria. The week’s challenge race was a success, however there was one problem. The member of my team that was in charge of activating the racecar during the challenge accidentally saved the detected images into a non-existent directory. I have become more aware of how mistakes, no matter how small, will greatly affect the outcome of a situation, especially something like in programming.

**Week 4: Bring all the pieces together for a final competition**

The task for the the final week was to wrap up everything we learned and created into the final code for race day on Friday. We had miniature tech challenges that emulated the weekly competitions from the previous three weeks, but now on the official race track. There was, no doubt, so many things that needed to be brushed up upon and created. The small and insignificant glitches became the biggest causes of our failure.

The main goal for my team was to detect the source of the malfunctions, understand the issues, and then fix them. Natural lighting became a problem for every team: due to the different angle of the sun in relation to us, the amount light entering the building changed as the day went on. We decided to create two files - one that holds the code that correlated with the morning light and one that worked better with the afternoon/evening light. It was a success.



Figure 5. Week 4’s racecar: Team 66 – A. Shen, B. Spiegel, W. Hopkins, J. Borowsky, M. Krieger

My goal for the week was to understand and incorporate the making and activation of launch files in ROS, a very crucial tool that would help ease the process of launching all of the necessary files at the same time.

1. **Your approach:  how did you decide to set about your work, and why?**

Slow and steady wins the race. Despite having an initial disadvantage for the lack of understanding and usage of ROS, I took the time to ask for help and teach myself the system’s platform. However, I understood that helping out my teammates was my main priority, so I contributed to the team by helping out those who were not as familiar in Python coding as well as creating nodes that would benefit our progression because that is something that I do have expertise in.

I discovered that the best way to learn for me was through trial and error. Yes, it was tedious, but it benefitted my technical understanding of the system so that I would be able to help out my team. On the mental-processing aspect, I felt like controlled multi-tasking was key to efficiency in learning, understanding, and implementing my acquired knowledge.

I decided that the most fitting role I should play was the mediator of my team. I have observed the significant lack of positive communication and the overwhelming amount of miscommunication and ignorance my team had during Week 3 and Week 4. If there were to be tumultuous bickering all the time, our team will go nowhere. I also became one of my teammate’s “sanity checker,” a person who listens, questions, and understands the math used within the code and the code itself that is trying to be created. As I helped out, I realized that it also internally helped me such that I had a more detailed understanding of the implementation of ROS rather than just the conceptual aspect and the big picture.

1. **Your process:  step by step, what did you do?**

Every day, I would write notes from the daily technical lectures and seminars. I would go over the notes once the classes end to get a reiteration of what I had just learned. I took the time to experiment with ROS by searching and using tutorials I found online so that I would be more comfortable to this new environment. I also decided that the best way for me to fully understand and learn about ROS is to ask my teammates as well as the assisting instructors for help.

My means of a successful team is balancing the technical work with constructive and positive human collaboration and authentic interaction. Despite the fact that this course contains extremely technical concepts and applications, the only way this competition would succeed was shift our minds around the constant “me” to more of “us.”



Figure 6. Team 66 with our poster: Sanic, The Hedgehog Meme

1. **Your results:  What did you test?  What did you accomplish?**

I was able to finally enjoy and understand a more in-depth picture of ROS as well as the field of autonomous vehicles.

With the effort I put into learning and testing out different perspectives of coding, I was able to put input to my team’s works by creating a code with a fellow teammate that would optimize the racecar’s options when it comes to deciding how to maneuver around obstacles that are put onto the track using trigonometry. After my teammates had completed the necessary codes for our racecar, I was able to create launch files that would activate specific nodes and other launch files for given situations. Furthermore, I contributed to my team by wrapping and editing the final codes before the “code freeze” session, although we still needed some more time to fix the sudden issues that decided to pop up.

1. **Technical Conclusions**

My role within the majority of my teams was researching and simplifying different ways to complete the labs. Eventually, I found the most optimal way to do so. Especially after the daily lectures, I would search for research papers on the new algorithm(s) and concepts that were taught, pick and choose the crucial factors necessary for our team’s weekly task, and discuss my findings and the ideal procedure to follow through with the rest of the team.

During Week Three, for example, we learned about the Potential Field Algorithm within the larger scope of motion planning. Motion planning is a robotics term that explains the process of deconstructing a robot’s desired task into distinct motions necessary and ideal to that overall end movement. In our case, Potential Fields were the way to go. We were taught to view this concept in a physics perspective using electron fields. Imagine the racecar as a loose electron floating between walls of positive ions, a clear path, and negative ions, the obstacles. The racecar (electron) should be attracted to the goal (+ ions) and repulsive to the obstacles. The advantages of using this approach shows that the trajectory is produced with minimal computational power. However, they can become trapped in local minima of the potential field, and eventually fail to find a path. My team and I decided that the best way to go about this was to focus on simplicity in terms of optimization using trigonometry. In other words, we each tried to create what we felt was the simplest code that emulates the potential field algorithm. Eventually, we would compare and choose the most effective and efficient code to use for the test trials.

During Week 2 and so forth, my teams were able to successfully complete the weekly tasks. I believe that it was due to a clearer understanding of how important team collaboration and communication. We all realized that if we all want to get the most out of this summer program, we would have to focus more on learning and having fun during the process rather than “winning” through the works of one skilled person. Personally, I was also able to have a much stronger grasp of ROS’s usage and its commands as well as its link with Python and Linux.

Week 1 was definitely a difficult week for me. But that’s what I love about struggles and failures. One can only learn and improve from the mistakes and in Week 1’s case, it was lack of overall teamwork. We were all hearing one another, but not really *listening*. There were sporadic periods of voices being raised, miscommunication, and ignorance. With all this accumulating from the beginning of the week, our team got disqualified twice during the challenge. The racecar was working fine during our own test runs, but one teammate decided to change the code last minute without any consent from the rest of the team. Of course, we were all upset, but we all learned a valuable lesson from one of Professor Jane Connor’s lectures: “If you want to go fast, go alone. If you want to go far, go together.”—African Proverb.

1. **What did you learn?**

I learned that the best way to learn is to expose myself to series of learning bursts; I give myself half hour periods to improve my skills on ROS. I also try finishing up the incomplete labs and/or go over old labs while researching new algorithms and useful tools introduced during the lectures. It is also important to me to balance out different situations at hand pertaining to the team’s goals and problems. This way, I would not lose focus or interest, but also keep my brain running effectively.

How one could contribute to a team comes in many unique but helpful forms. Something thought to be so insignificant may result to a successful “win.” Listening and giving/receiving constructive criticism, for instance, are essential to any strong group. These skills give the team a chance to take and step back to reflect and understand the bigger picture before initiating any actions.

Another important key that enlightened me was how crucial prioritization and completing tasks became, especially during the final week. My team still had so much to do, yet the most of the team focused too much on editing the already written code to make it look more “visually appealing” than on actually finishing the actual criteria needed for the racecar to successful operate during the runs. When it came down to the last minute before “Code Freeze” occurred, many of us needed “more time” to complete what should have been already done beforehand. In the future, I will spend time trying to keep things moving efficiently by setting time constraints for every task to be finalized. This way, by keeping track of the time and our pace will decrease the unnecessary pressure and stress we put upon ourselves as well as actually finishing up what needs to be done.

1. **What might you do differently going forward?**

Instead of feeling slightly vulnerable when it came to asking for help, I now understand that people are more than willing to support and teach others without any negative judgement. I will, to my best ability, prioritize my time to focus more on the task at hand rather than worry about what could happen. I’ve realized that there are two types of questions: the one that I *think/perceive* is being asked and the one that is actually being asked.

1. **Personal Reflections**
2. **Describe the key things you learned as a person, for example how to think about yourself and your future, how to relate with others, how to work on a team, and anything else you learned.**

I already know that I have an over-evaluating/ over- thinking mentality that I feel is more of a burden than a strength. During this program, I’ve learned to accept that part of me and rather than let it drag me down, I took advantage of its perks. I’ve learned that this enables me to focus and edit on the minor mistakes that will, in fact, build up into a bigger conundrum.

Initially, I referred and joked about being the epitome of Imposter Syndrome. Deep down, I actually did feel like I didn’t belong in this program, seeing that the people around me were so much more advanced with coding and robotics in general. But eventually with a little bit of discipline, I was able to catch up. Self- doubt is never a good thing until you learn how to overcome it.

I am told to be very relatable in the sense where I have high levels of compassion and empathy. From my experience here, I’ve learned the important skill of group teamwork, such that it is not like a swim or track competition where everything is more or less individualistically based.

I’ve learned how to collaborate with all types of people, all of whom learn and code differently. I had a chance to further develop my compromising skills as well as process what I am about to say before actually saying it.

1. **You can include here any thoughts you have about how to improve the program for the future.**

I suggest expanding the time here for the program, therefore the learning process would not be so overwhelming.