The Gauntlet

Sparsh Bansal
Quantitative Engineering Analysis
Electrical and Computer Engineering
Olin College of Engineering
Needham, Massachusetts 02492
Email: sbansal@olin.edu

Colin Snow
Quantitative Engineering Analysis
Electrical and Computer Engineering
Olin College of Engineering
Needham, Massachusetts 02492
Email: csnow@olin.edu

I. Introduction

The objective of this challenge, that we chose to accept, was to drive a NEATO robot to a goal marked by a bucket while avoiding two obstacles. The position of all three of these objects was randomized for each trial, and the program had to rely only on its LIDAR (Light Detection and Ranging) to identify, map, and traverse the challenge. Figure 1 and Figure 2 can be used to visualize the challenge.

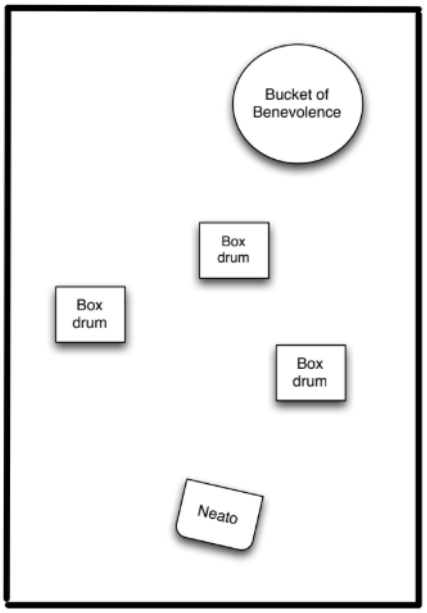

Fig. 1. A diagram of the challenge

The NEATO robot uses differential drive with an open-loop control. The sequence of motor commands for a particular moment of time were determined before the robot executed the published commands. The MATLAB script sends these motor commands to the robot at the prescribed times irrespective of where the robot is along the path. This lack of feedback means that there is a potential for the NEATO to be misaligned at some point down its path, which can lead to issues if it does run into something. Figure 3 shows an example of a NEATO robot from NEATO Robotics.

Fig. 2. A top view of the Gauntlet Challenge

Fig. 3. A NEATO

II. Strategy

Because the Gauntlet is such an open challenge, there are many different strategies that can be used to get to the goal. We tried to find the simplest effective way to deal with finding, identifying, and traversing the challenge.

The NEATO LIDAR returns data in the form of a point cloud with a radius value for every degree of rotation. After converting to Cartesian coordinates, this cloud needs to be decoded into the walls and bucket that it represents. The outer walls of the cage can be disregarded because they have a known position, but the movable objects in the middle need to be identified.

A plot of the map that was generated by the LIDAR is shown in Figure 4.

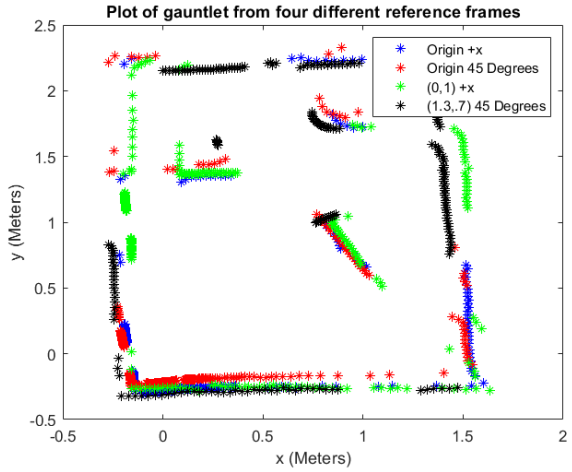

Fig. 4. Four LIDAR generated maps of the pen, each from a different position. The NEATO can only see one of these data-sets at a time.

A. RANSAC

We accomplish this identification with RANSAC, an iterative line-finding algorithm which excels at finding trends even with many outliers. It works by selecting two random points in a data set, drawing a line through them, and then checking how many points lie close to it. By iteratively doing this and selecting the best line, RANSAC is able to maximize the in lying points and effectively ignore outliers.

The challenging part of implementing this algorithm is finding the perpendicular distance between any point and the line to determine whether it is an inlier or outlier. In order to do this we must first find the tangent and normal vectors to the line:

$$\hat{T} = ||\vec{V}||$$

$$\hat{N} = \hat{T} \times \hat{K}$$

Where $\vec{V}$ is the vector representation of the line and $\hat{K}$ is a unit vector in the $+k$ direction (Into the floor)

We can now define a vector $\vec{A}$ which is the distance between the first point of the line and any point in the data-set. With this definition, the component of this vector which is parallel with the normal vector is the perpendicular distance and the component parallel with the tangent vector is the parallel distance. We can find these vectors simply by finding the dot product:

$$ParallelDistance = \vec{A} \cdot \hat{T}$$

$$PerpendicularDistance = \vec{A} \cdot \hat{N}$$

With these distances found, all the points inside this can be replaced by the line and all outside remain for the next iteration of RANSAC, thus the program can find all lines in a point cloud regardless of how noisy it is.

In our specific context, we run RANSAC two times, once for each obstacle. The output of this run can be observed in Figure 5 as red and blue lines.

B. Finding the bucket

In line with our quest for simplicity, we find the bucket simply by specifying that it should be the last thing left after all the obstacles are removed. Once we remove the walls and run the RANSAC, all that is left of the point cloud is a few outliers and the points that define the bucket. By finding the median of these remaining points, we are almost always able to identify one of the points on the outside of the bucket, which is exactly where we want to stop. This method is limited in that when the RANSAC fails it can sometimes misidentify the bucket, but other than that it has been remarkably effective for a range of obstacle positions. This approach gives us the approximate location of the bucket, as shown as the yellow star in Figure 5.

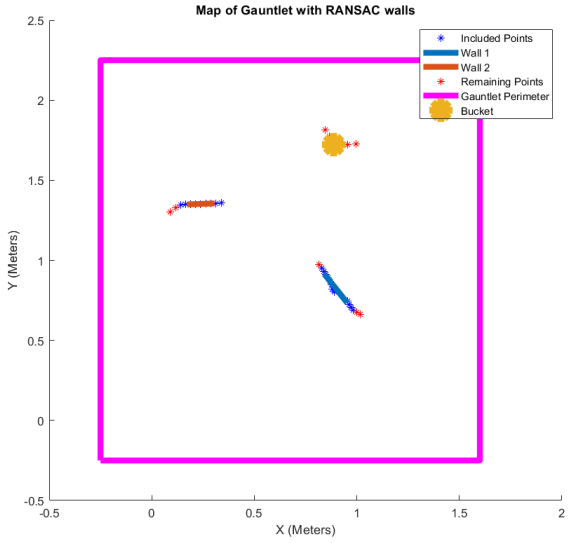

Fig. 5. A plot of the RANSAC processed map is shown in Figure 5. The walls are displayed in magenta, the obstacles in blue and red, and the bucket in yellow.

C. Creating the Gradient

Once all goals and obstacles are identified, we can create a function whose gradient directs the NEATO to the goal while avoiding the obstacles. In order for the descent to work properly this function must have its only minimum at the bucket, and have local maximums along the length of each obstacle. In order to guarantee that our function had no extra local maximums or minimums, we defined each object with the function:

$$F(x,y) = \pm \int_a^b ln\sqrt{(x-x_0)^2 + (y-(m \cdot x_0 + b))^2}\, dx_0$$

where $m \cdot x^2 + b$ is the change in y for every change in x_0 and $[a, b]$ is the domain of the segment. This function allows us to integrate along the length of any obstacle line and create a continuous high point to drive the NEATO away from it when the function is negative, or a low point at the goal when the function is positive.

The end result of this construction is a 3D function where the NEATO starts at (0,0) and proceeds downhill into the target. A 3D contour plot that can be used to visualize the gradient that we programmed is shown in Figure 6. The obstacles can be spotted as the 'hills' (yellow-orange surface), while the BoB can be spotted as the deep sink (blue-green surface).

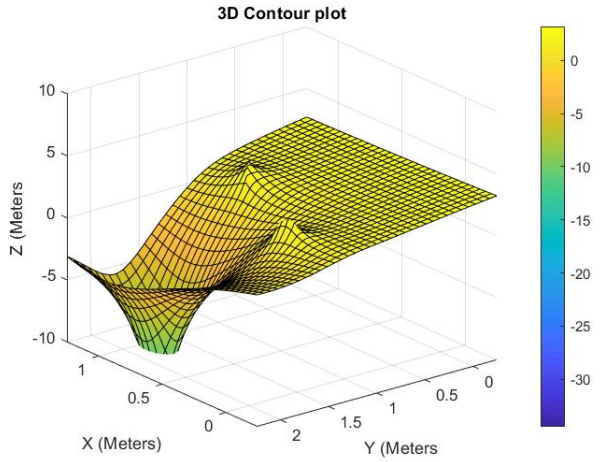

Fig. 6. A 3D Contour Plot of The Gauntlet with raised boundaries at obstacles and the target at the lowest point.

D. Descending the Gradient

The final step in reaching the goal is to perform gradient descent on the function we defined previously. Gradient descent is the process of iteratively moving in exactly the opposite direction of the gradient of a function until a local minimum is reached. This process involves starting at some point (x,y) and then moving a distance λ in the direction of steepest descent, where:

$$\lambda_{i+1} = \delta\lambda_i$$

and where λ_0 and δ are arbitrary numbers chosen for the application.

However, for this particular real world case where our robots have a maximum linear speed of $.3m/s$, it makes more sense to descend the gradient with a constant rather than a variable step size. So, instead of using the traditional formula above, we define:

$$\lambda_i = \frac{\delta}{\nabla}$$

Where now δ does not define the change in step size but instead is the constant step size for each iteration.

While this method is very advantageous for the robots where speed is an issue, it is often less effective because it tends to overshoot the solution and has extreme trouble converging on a solution. We fix this problem by putting a simple tolerance on the gradient magnitude at which we want it to stop descending. This works for this case because the gradient approaches $-\infty$ at the minimum, but if this was not the case a more complicated stopping mechanism would be necessary.

III. The Program

The equation that we use to drive the gradient can be accessed through this link: Generated Equation

The NEATO in action in high-quality can be accessed through this link: Challenge Accepted

The full program, named "neatoGradientAscent" can be accessed on the authors' GitHub repository through this link: GitHub Repository

It takes the NEATO about 29 seconds to gently touch the BoB, starting at the global origin of the Gauntlet. The total distance covered by the NEATO was 1.95 m.

A potential field of the pen is as shown in Figure 7.

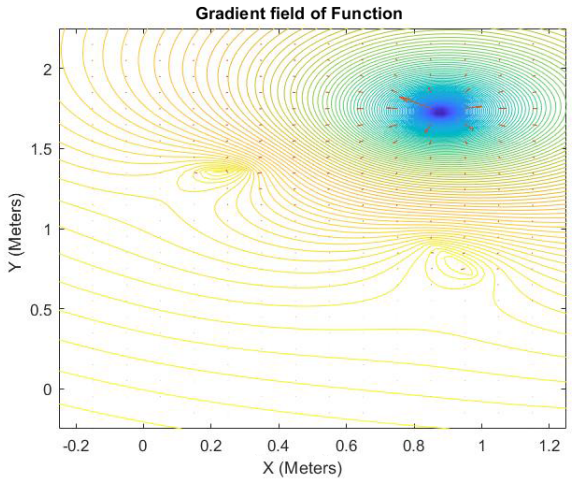

Fig. 7. Potential Field of the pen

Runnign the program sends the NEATO left and right wheel velocities at discrete time steps, approximating the path of the ascent. The theoretical and experimental paths of the NEATO are shown below

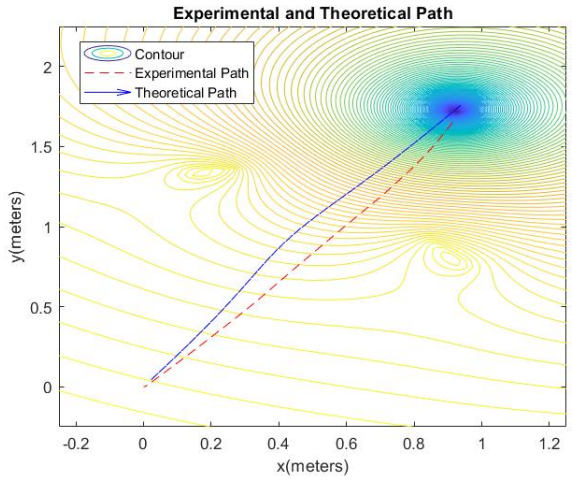

Fig. 8. Theoretical Path vs. Encoder Data