# Fall 2017 CSE 325 Project 6
# GPS Boundary Wall Avoidance

## Deadlines

**When to submit the files of this project on blackboard:**

**Friday, Oct 20, 2017, 6:00pm, AZ time.**

**When/where to demo this project:**

**Friday Oct 20, 2017, 6:00 pm AZ Time**

**In front of Old Main which is in 425 E University Dr, Tempe, AZ 85281,**

**33.421164, -111.934012**

## Project Description

The purpose of this project is to become familiar with path planning. The goal of this project is to navigate inside the defined polygon using GPS and never leave it.

## Required Hardware:

- Arduino Mega 2560 and cable.
- Arduino Nano and cable
- Fully built robotic car from previous project

## Part 1: Understanding the Task

To understand what a GPS boundary is and why we need it, consider an area where the robot should not leave. In this project, the testing location is the lawn in front of Old Main building. At this location, there are some bushes and other obstacles that are inherently hard to detect with LIDAR. Instead of modifying the program to detect these obstacles, it is easier to bound the area with a virtual GPS wall. See the following illustration:

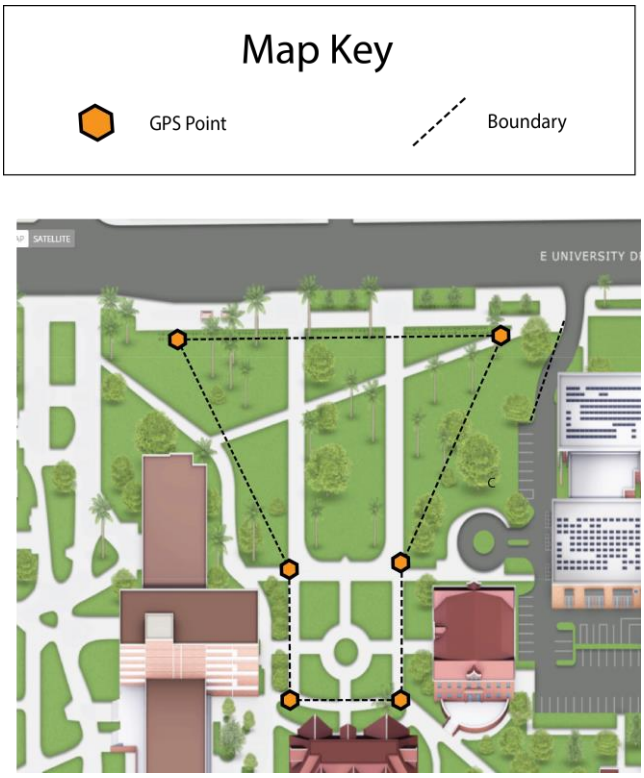



Figure 1 – Points of GPS boundaries

Bounding an area with GPS points is **not** easy. There are 3 tasks to be accomplished for this:

1) Define the vertices of polygon

1) Create the boundary lines using the vertices,

2) Find out how close the robot is to each wall (line),

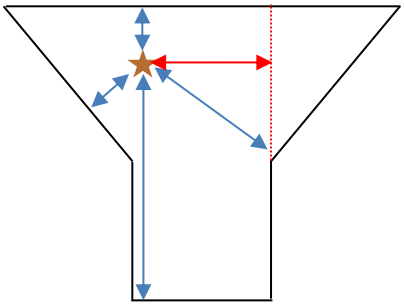
3) Determine an aiming vector to lead the robot away from the wall/walls.

1. **Defining the vertices of polygon**
   As you can see in Figure 1, there are 6 points on the map indicated with orange hexagonal.

2. **Create the boundary lines**
   A boundary wall (line) is defined as a straight line between the pair of vertices next to each other.

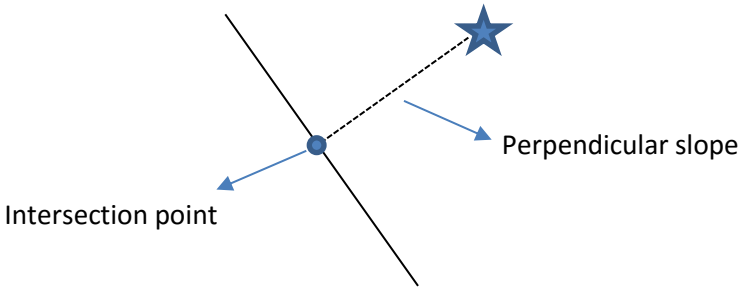
3. **Find out how close the robot is to the walls**
   The next task is determining how close the car is to the edges of the polygon. This task is easy to accomplish by finding the perpendiculars from the current position of the car toward each edge. **(Note that, some perpendiculars does not lay on the edge, so you should ignore them).** This can be accomplished by simple geometry calculation. For

instance, if you are at the point indicated by star, blue lines show the perpendicular from each edge.

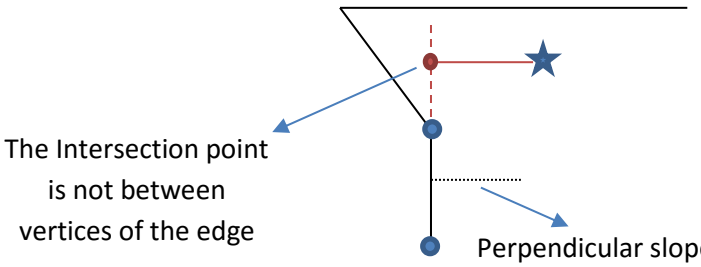


**Note that, perpendiculars may not exist for some edges (The red one in the above figure)!**

To find distance from an edge, you should first find the perpendicular slope of the edge and then create a line with one point (robot's position) and the computed slope. Then, by crossing the computed line with the edge, you can find the intersection point. Finally, you can compute the distance from the intersection point.

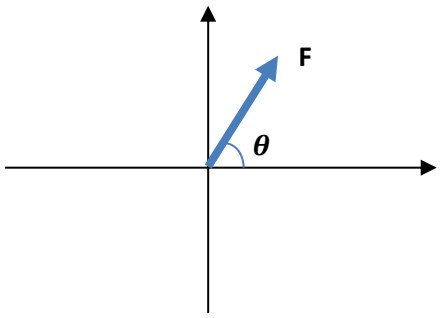



Then, you should check if the intersection point is between vertices of the edge.

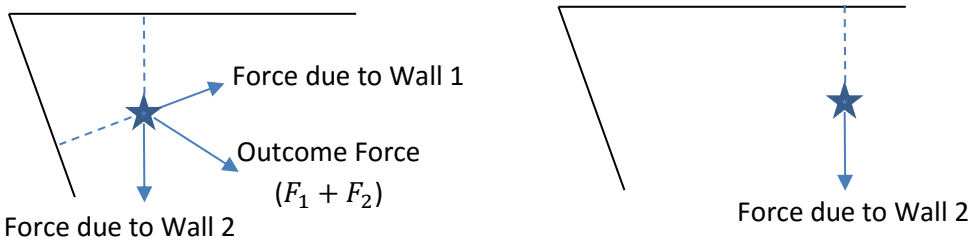



If the intersection point is not between the vertices (Figure above), ignore it.

4. **Determine a force vector to push the robot away from the wall**
   Aiming vector can be defined by force F and angle θ. Force can be calculated according to perpendiculars of each wall.




Whenever distance is less than some threshold, force vector is calculated.




# Part 2: Car Navigation

You should program the car to initially drives towards a straight line. If the car reaches near to any of the predetermined GPS walls, it should turn and continue driving in opposite direction according to computed force mentioned above. In case of obstacles, you can pick your car and pass it manually.

Since the test would be taken in front of Old Main building, the car may be driving on the grass and between the trees. The robot may be set to start driving from anywhere in the polygon and pointed towards an arbitrary direction. TAs may pick up your car and put it near to a wall to see if it's working. This will allow the TA to quickly test multiple aspects of the project.

# Submission files:

You have to submit the following files

1. Mega.ino (Arduino Mega program)
2. Integrity.txt
3. Group's_members.txt

You should submit any libraries you may have written for this project. All necessary codes should be zipped into a single file called CSE325_Fall2017_Project6.zip.

**Submit on time. Late submissions will get 0.**

# How the project is graded:

On the demo day, you should press the select button on the keypad to start navigating. The TA may set the orientation of the car at the start point. You may perform 3 demos with different TAs. Grading scale is shown in the table below.

| Points | Description |
|---|---|
| 3 | Car moves in a straight line |
| -5 | Car goes outside of any GPS boundary |
| 2 | Car avoid boundaries |
| 5 | Car should not get closer to any boundary more than 1 meter at all time and also should not try to avoid until it is within 5 meters of a boundary (i.e. doesn't react too early or too late) |
| -5 | If the implementation is polling based rather than interrupt based |
| | If you do not upload your code up to the deadline, you will be graded 0. |