James Delande
CIS 602 – HW3
Final Project Paper
2014-05-01

My Final Project Idea is an expansion off some of my previous work. When dealing with underwater target detection, coordination among a team of autonomous underwater vehicles (AUVs) is a very large concern. The sensing areas for each vehicle should complement one another, and overlapping and useless routes should be avoided, in order to maximize the possibility of detecting a target. But when putting together a configuring for a team of AUVs it can be difficult to determine an adequate layout. A number of variables come in to play that makes the problem exponentially hard to brute force, so some better way of visualizing the problem space is necessary to allow a human to determine a sufficient configuration. Some of the variables considered are:

- Area of coverage
- Number of vehicles
    - How to distribute the vehicles
    - Formations if a vehicle fails/leaves
- Different types of sonar
    - Forward facing
    - Side scan/Towed array
- Paths for the vehicles
    - Where to start in the path
    - What direction to start in
- Communication range
- Speed of the vehicles

Previously I made a tool in MATLAB that took in log files from simulation runs and created a heat map of the sonar coverage based on the X, Y coordinates and heading of the vehicles (Figure 1). This tool fulfilled its use, but it is very rigid and if I want to generate any new maps I need to run a simulation first, which can take upwards of 20 minutes. The drawing of the map itself was also memory intensive and slow, and did not take into account the fact that two vehicles looking in the same area at different times is a good thing.

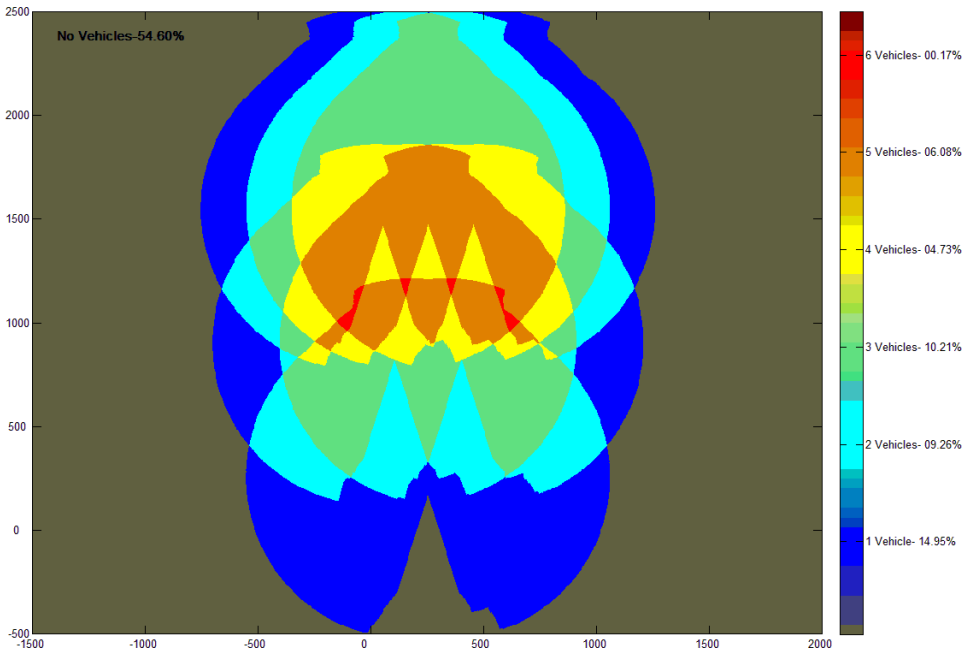

Figure 1 - Heat map made in MATLAB for sonar coverage

The method used is a combination of the JavaScript library D3 and HTML5 canvas. My initial attempt used full D3 instead of canvas, but I was unable to create the heatmap effect this way. Canvas is allows for pixel by pixel manipulation in a fairly efficient manner, which is how I managed the additive drawing for the heatmap. Some issues arose because of using both, like converting the transforms between the two, but overall I believe it was the best and easiest method. I ended up removing most of the transforms due to this issue, since the D3 transitions would not update the actual object data. This made it so that when I got the object data, it would be the initial data and would be useless to the canvas drawing unless I applied the transform to the canvas as well. This got tedious because I was making transforms on entire groups so I would have to check each parent object and chain the transforms. I opted, instead, to do the trigonometry myself and drop the transforms.

I will be continuing with this project, but may have to move it offline at some point, or at least find a way to accomplish more rendering server side. The canvas had to be limited in size since there were some scaling issues, especially on low memory machines or laptops. I believe I was able to speed it up some by doing off-screen rendering and then putting the image on screen after a set interval, but I have not done any performance testing to prove this. In the future I would like to add in the network maps, which I may actually do before my presentation on Tuesday, and have an initial configuration page to input parameters for each vehicle. So far this implementation is much more flexible than the MATLAB one, so I have high hopes, just as long as the scaling up of the display area is not an issue.