Time

I worked alone on this lab. I imagine I spent around 20 hours.

Initial Matrixes

I didn't worry too much about the initial covariance (sigma_0) matrix, knowing that the Kalman filter would be able to correct itself after making several observations. The initial covariance matrix I finally decided on is

200.0	0	0	0	0	0
0	0.1	0	0	0	0
0	0	0.1	0	0	0
0	0	0	200.0	0	0
0	0	0	0	0.1	0
0	0	0	0	0	0.1

In essence, the position noise is 200.0 while the noise for velocity and acceleration are both 0.1. The initial estimate of the mean guesses a velocity of 0 and an acceleration of 0. I always run the hunter tank before I run the enemy tank, so the initial velocity and acceleration are both 0. Thus the initial estimates of velocity and acceleration are always going to be correct, so I used a very small noise of 0.1 for the initial covariance.

The initial estimate of the mean guesses a position of (0, 0). With the world I'm using, the enemy tank starts out around (-200,0). I wanted the actual position to be within one standard deviation of the original estimated position, so I use a position noise of 200. Consequently the footprint of the graph of the original estimate is quite large, but once the hunter tank starts making observations the estimate quickly becomes much more precise.

Figuring out the transition matrix (sigma_x) was much more difficult. At first I used the one provided (0.1 for position and velocity, and 100.0 for acceleration) and my hunter tank was not performing very well. It tended to always shoot behind the constant velocity tank. As I studied my debugging printouts, I saw that my estimated acceleration was usually way off, and that made my future predictions of the enemy's location way off. I knew that I needed to fix the transition matrix.

At first I used trial and error and was able to improve a little that way. The acceleration still messed up my future predictions, however, so I decided to change the way I made future predictions. Instead of assuming that the tank would continue with constant acceleration and linear velocity, I made the prediction assume no acceleration and constant velocity. That made my hunter tank perform much more accurately, but it makes the assumption that the mean estimate of the velocity is correct and that it will be constant over the time it takes for the bullet to

travel. In order to make sure that the estimate of the velocity would be close enough, I performed some experiments to calculate good values for position transition noise and velocity transition noise.

For these experiments, I made the Kalman filter assume no transition noise. I then made 100 observations of the sitting duck tank and recorded how that changed my estimate of the position. I found the covariance of the 100 estimates usually fell between three and five, so I chose four as the value for my position noise.

I ran a similar experiment to calculate velocity noise. I assumed no transition noise and made 100 observations of the constant velocity tank and recorded how that changed my estimate of the velocity. I found the covariance of the 100 estimates usually fell between 10 and 40, so I chose 25 as the value for my velocity noise.

To get a good acceleration noise, I just frobbed a bit until I got good performance. I ended up choosing 50 as the value for my acceleration noise, making my transition matrix as follows:

$$\begin{vmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 25 & 0 & 0 & 0 & 0 \\ 0 & 0 & 50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 0 & 0 & 50 \end{vmatrix}$$

Abilities and Limitations

The point of the Kalman filter for this lab is to help a hunter tank make an educated guess of an enemy tank's position, velocity, and acceleration when it is only given noisy estimates of the enemy's position. If the data is accurate enough, the hunter tank can predict where an enemy tank will be in the time it takes for a bullet to travel between it and the enemy tank. While this works pretty well, there are a few limitations.

One limitation is that the hunter tank only has a noisy estimate of the enemy's position. This makes it difficult to accurately calculate velocity, which in turn makes it even more difficult to accurately calculate acceleration. This makes it difficult to predict into the future, which you have to do because the time it takes for a bullet to travel is not negligible.

Another limitation of my implementation is that future predictions assume the enemy tank maintains constant velocity. In other words, I assume that the enemy tank's speed and direction will be constant between the time I fire the bullet and the time the bullet reaches my estimate of his location. The bullet travel time is

sometimes a few seconds, so if the enemy tank changes speed or direction at all in those few seconds, the bullet could easily end up being way off target.

In a world with obstacles, my implementation would definitely be limited. I don't test to see if there are any obstacles between the hunter tank and the enemy tank before I fire. By wasting bullets to shoot obstacles, I might not be able to recharge in time to make shots that could actually take out enemy tanks. I also don't test to see if other tanks on my team are between me and the enemy tank, so if friendly fire is turned on I could end up wasting shots on my own tanks.

Capture the Flag Tournament

In the tournament, I would definitely want to make a few changes. One, I would want to test for obstacles or friendly tanks between the hunter tank and enemy tanks. I wouldn't want to shoot allies and I wouldn't want to waste shots on obstacles.

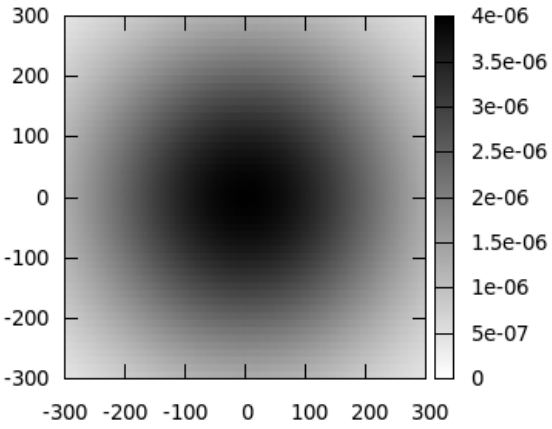
I would want to have two defensive tanks that stayed near the base as hunter tanks. In this lab, the hunter tanks are stationary and just rotate and shoot. I would want the tanks to patrol back and forth, waiting for an enemy tank's estimated position to be within firing range. Once an enemy came into range, I would stop the hunter tank and use the Kalman filter to determine how I should rotate and where I should shoot.

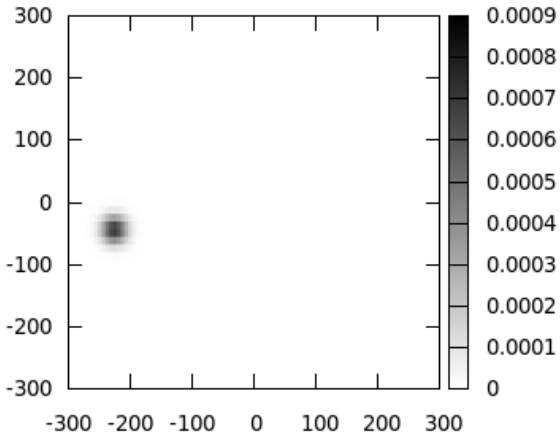
I would also make some changes to my tanks going after enemy flags. I would want them to avoid getting shot by enemy tanks. One way to do that is to randomly vary my acceleration so that it's difficult for the enemy to accurately predict where my tanks will be in the time it takes the bullet to travel. I could use a changing random potential field to help with that. If friendly fire is disabled, I would also have my offensive tanks shoot randomly in the hopes that they might shoot enemy hunter tanks before getting shot themselves.

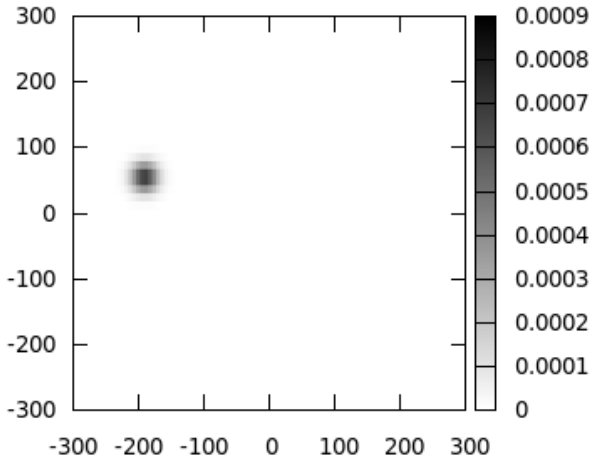
How the Estimate Changes Over Time

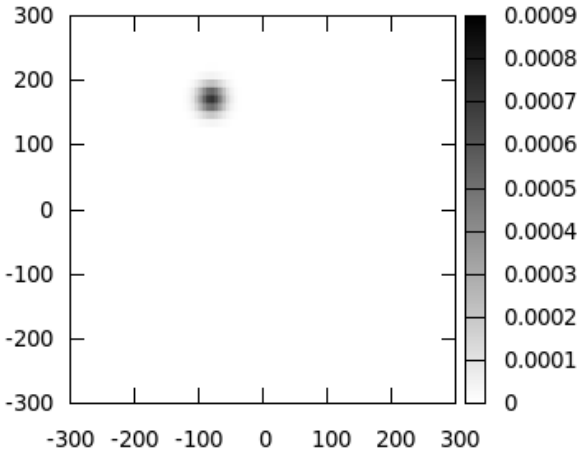
For each category (position, velocity, and acceleration), the first graph is before any observation is made. The second is after 50 observations, the third after 100, and the fourth after 150.

Position

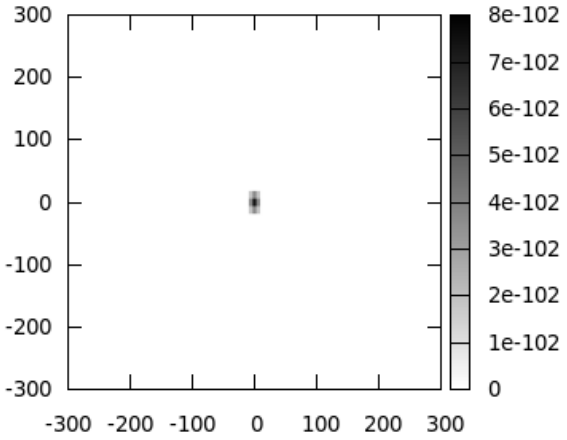

300
200
100
0
-100
-200
-300
-300 -200 -100 0 100 200 300
4e-06
3.5e-06
3e-06
2.5e-06
2e-06
1.5e-06
1e-06
5e-07
0

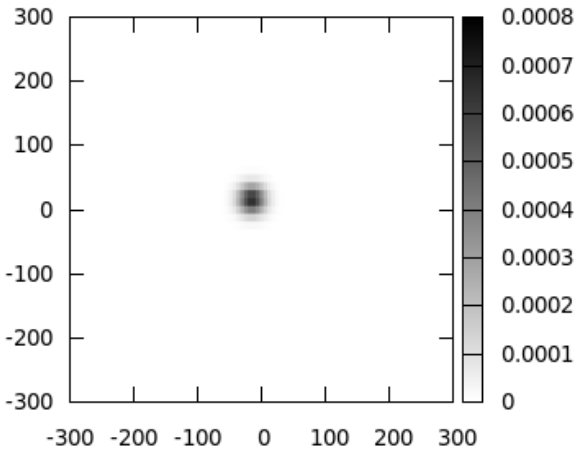

300
200
100
0
-100
-200
-300
-300 -200 -100 0 100 200 300
0.0009
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0

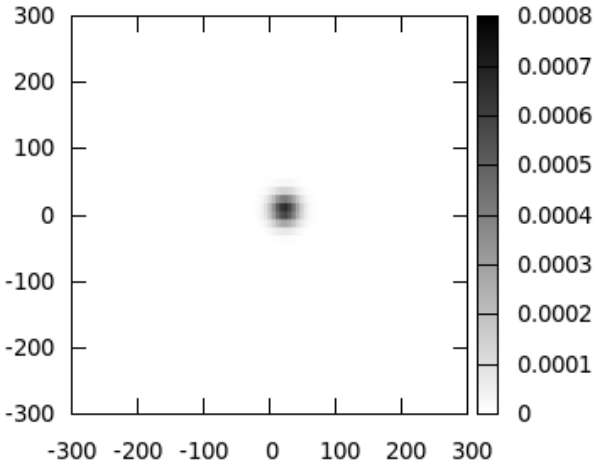

300
200
100
0
-100
-200
-300
-300 -200 -100 0 100 200 300
0.0009
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0

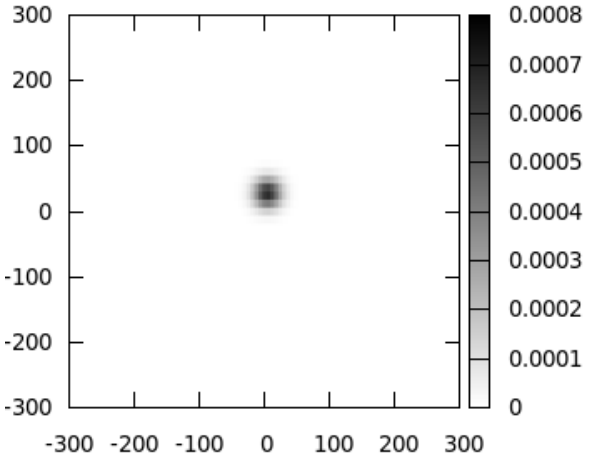

300
200
100
0
-100
-200
-300
-300 -200 -100 0 100 200 300
0.0009
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0

Velocity

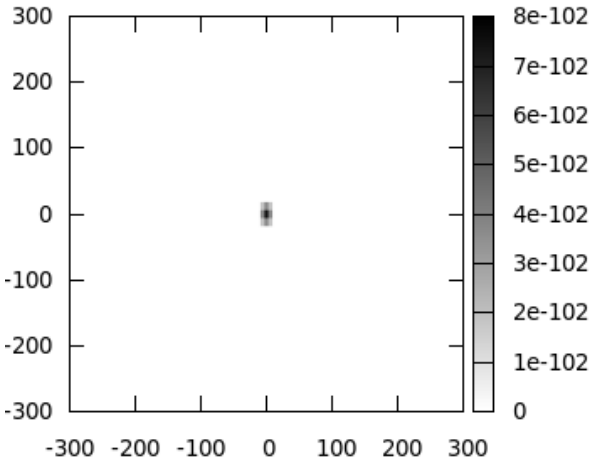

300
200
100
0
-100
-200
-300
-300 -200 -100 0 100 200 300
8e-102
7e-102
6e-102
5e-102
4e-102
3e-102
2e-102
1e-102
0

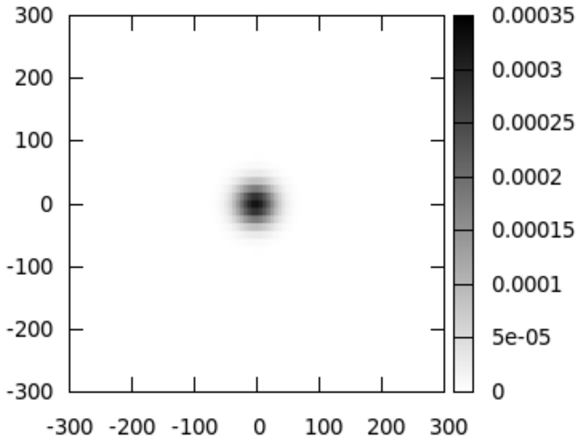

300
200
100
0
-100
-200
-300
-300 -200 -100 0 100 200 300
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0

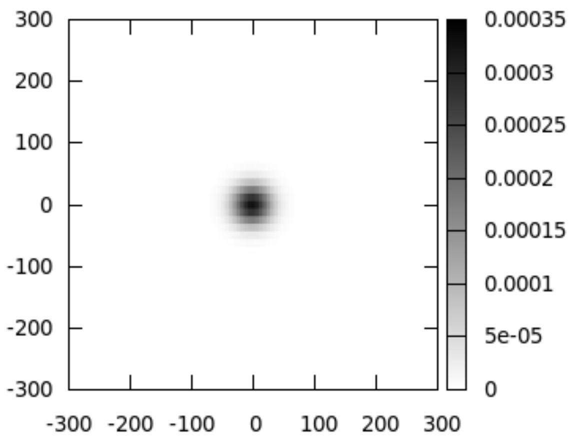


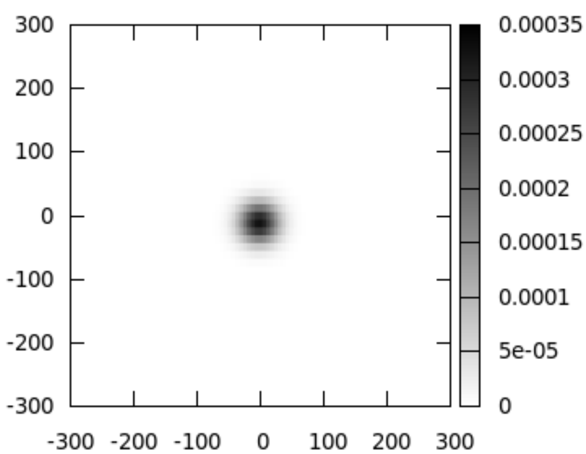


Acceleration




300
200
100
0
-100
-200
-300
-300 -200 -100 0 100 200 300
0.00035
0.0003
0.00025
0.0002
0.00015
0.0001
5e-05
0


300
200
100
0
-100
-200
-300
-300 -200 -100 0 100 200 300
0.00035
0.0003
0.00025
0.0002
0.00015
0.0001
5e-05
0


300
200
100
0
-100
-200
-300
-300 -200 -100 0 100 200 300
0.00035
0.0003
0.00025
0.0002
0.00015
0.0001
5e-05
0