An attempt has been made

1

The supplied robot path begins with six successive steps that are nominally straight forward.

A

If there is random disturbance along the robot's path caused by wheel effects, as characterised by σ_forward in the robot's process noise covariance matrix, then find the expected mean and variance in the robot's end position at the end of the six steps.

As the first 6 steps are steps of 10 and bearing changes of 0, the forward mean should be 60, the y mean should be 0.

From here, the variance of x after 6 steps should be n * variance at 1 step, as that is the variance of forward.

$\sigma^2_x = 1^2 \times 6 = 6$

The variance of the y position should be

$\sigma^2_y = 0$

As there is no bearing variance in this theoretical run, that comes later.

B

Similar to above, the bearing mean should be 0 as the input plan makes no change to it. Additionally, the variance is multiplied by 6

$\sigma^2_{bearing} = 2^2 \times 6 = 24$

The covariance matrix should have the diagonal multiplied by 6 and then the non-diagonal portions of the matrix should be non-zero as the bearing now has an effect on the heading (x,y pos) and vise versa.

2

After running the simulation for 100 runs of 6 steps, the following was calculated from the end points:

$\mu_x = 59.931$

$\mu_y =- 0.331$

$\mu_{bearing} =- 0.278$

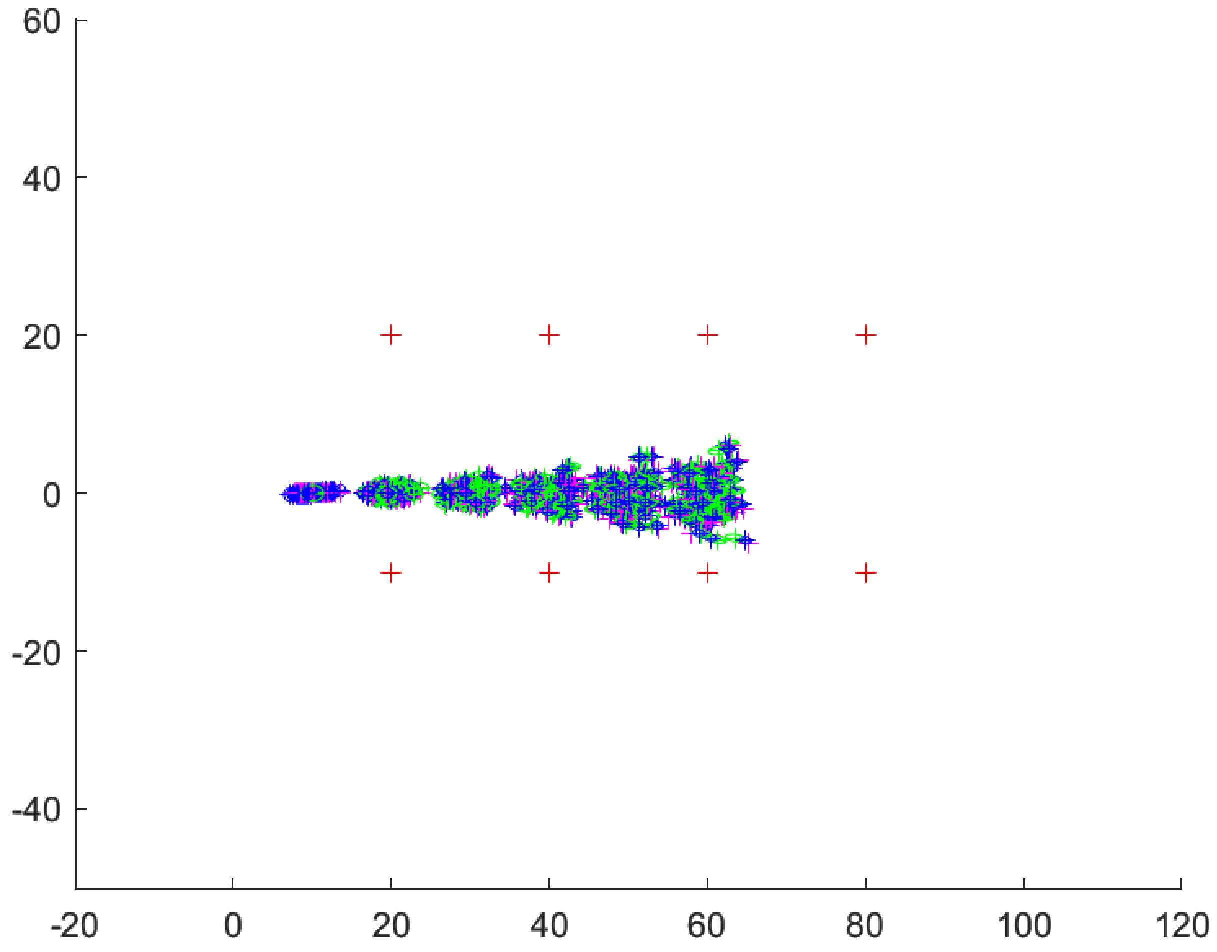
$\sigma^2_x = 5.452$

$\sigma^2_y = 5.646$

$\sigma^2_{bearing} = 23.288$

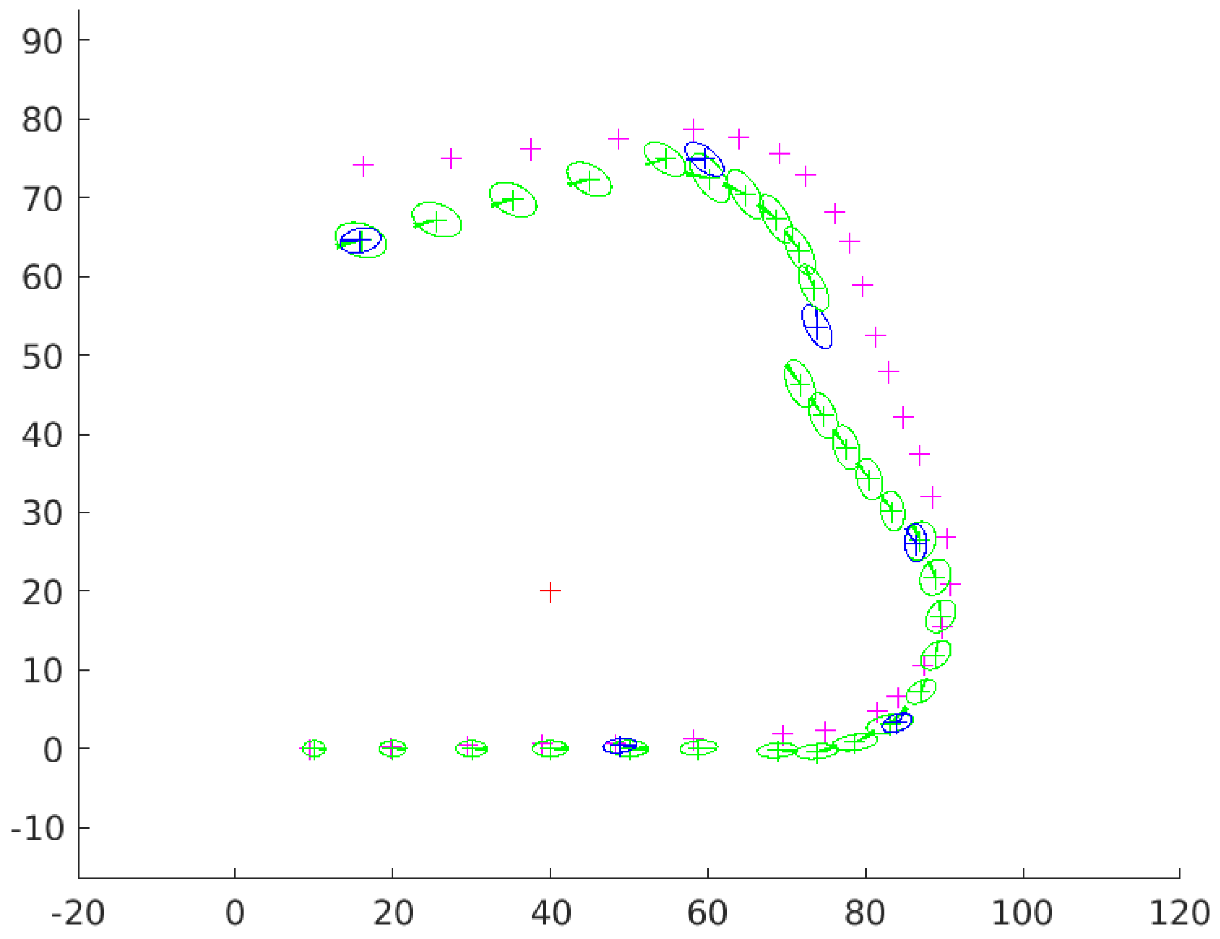
This is very close to what was expected in 1. And as expected with all things random, there is slight variation in the final values for the means and variances. However, it is expected that with more runs, these values will be closer to the true values. It is a good exercise at showing how some variation can be compounded through multiple steps of the same variation being applied.

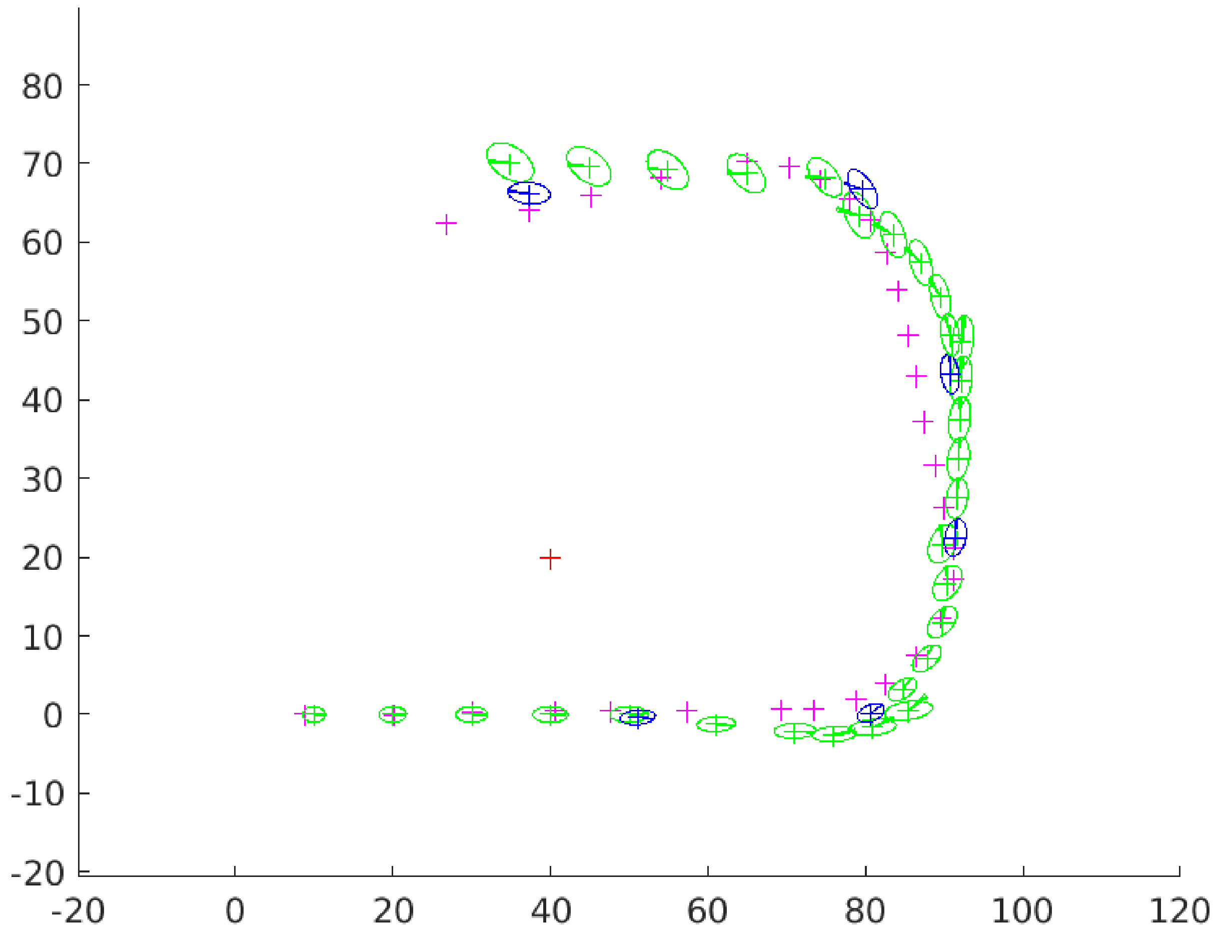
However, the variance of y in this simulation is non-zero as these 100 runs have introduced the bearing variance which impacts the y position at the end, and this y position is dependent on the forward variance based on that bearing, which explains why it has a variance of 6.



100 runs of 6 steps.

3





By using only 1 beacon with the initial measurement standard deviations of 5 and 10, this above plots are the result. From this we can see that the posterior is primarily uncertain about the heading estimate as it moves forward, and the measurements help the model adjust for what it is actually doing.
The measurements help constrain the estimates to within the measurement error, which is better shown by the first image as the estimate starts going wildly off course, but the measurement brings the estimate much closer to the true path.
It is also helpful at adjusting the bearing of the estimate, which is present in both plots where the estimate would start going off course but the measurement changes the ellipse and covariance, which shifts the direction the estimate starts moving.

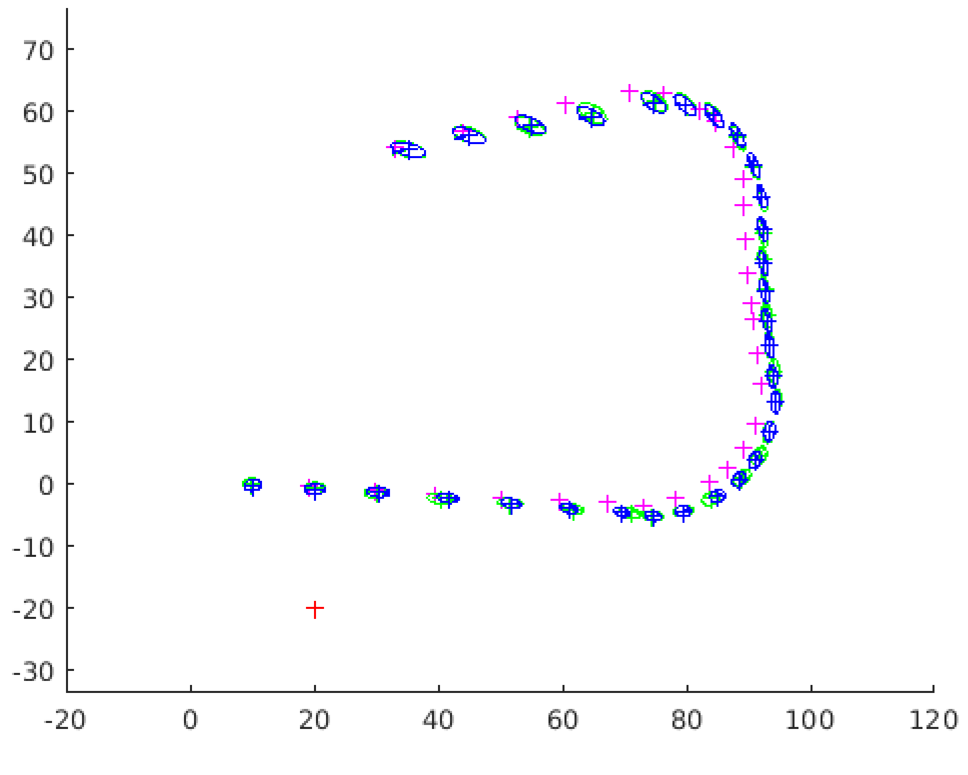
4

The minimum is 1, if using both range and heading measurements, and the placement of it doesn't seem to matter, however the error does appear to get worse as the robot moves away from the beacon. However, during the majority of the path it was found that a single beacon was capable of making accurate measurements with the measurement standard deviation being the original 1 and 2 for range and bearing.
However, increasing the number of beacons reduces the measurement error as shown in figures 4 and 5 of this section and again placement of these doesn't seem to matter, the only major change is that the tracking is overall better and the error ellipse is smaller.

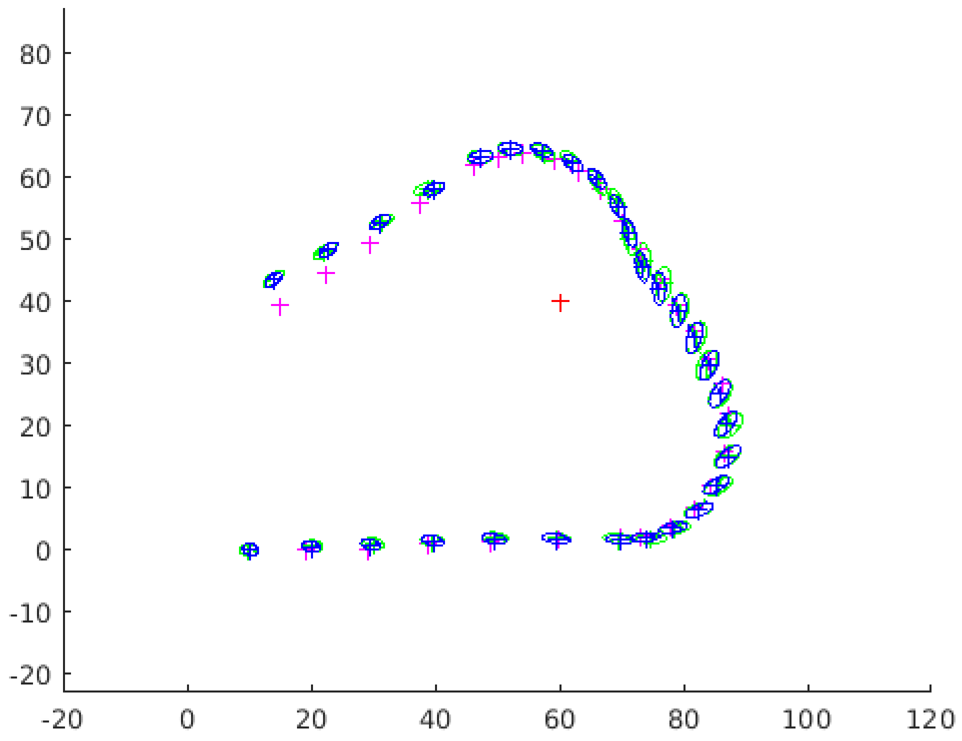
However, with more landmarks likely comes more resources to measure and process the measurements, and after a certain point there will be diminishing returns on the error improvement.
If the measurements were range dependent in their error, then you would want to increase the number of beacons to cover the major areas of your robot path. So for this example, it would be beneficial to stick a beacon near the start and end location, along with the location of the bends. As the robot distances itself from 1 beacon, it's already making its way towards another. See figure 6 of this section.
If only range measurements could be used, every turn would result in a slight transient response where the model tries to catch up. But after a few measurements in what the model considers the wrong direction, it would be able to infer a bearing and catch up, meaning that no real change in beacon placement would need to be made.
If only heading measurements could be used then the model would have to infer speed based on the measurements. This should have the same initial transient change that using only range would have, however it is likely more susceptible to error as it won't ever be able to accurately predict the distance travelled between measurements. You pretty much need to trust your model cause you no longer know where the robot actually is.

So from this, something like figure 6 seems optimal. Beacons around the area that the robot can travel in. There obviously needs to be a maximum range on a case by case basis that determines how many intermediary beacons are needed to account for the range issue, but being able to have the robot navigate between beacons in an area will reduce the overall error, especially as the robot gets further away from beacons.

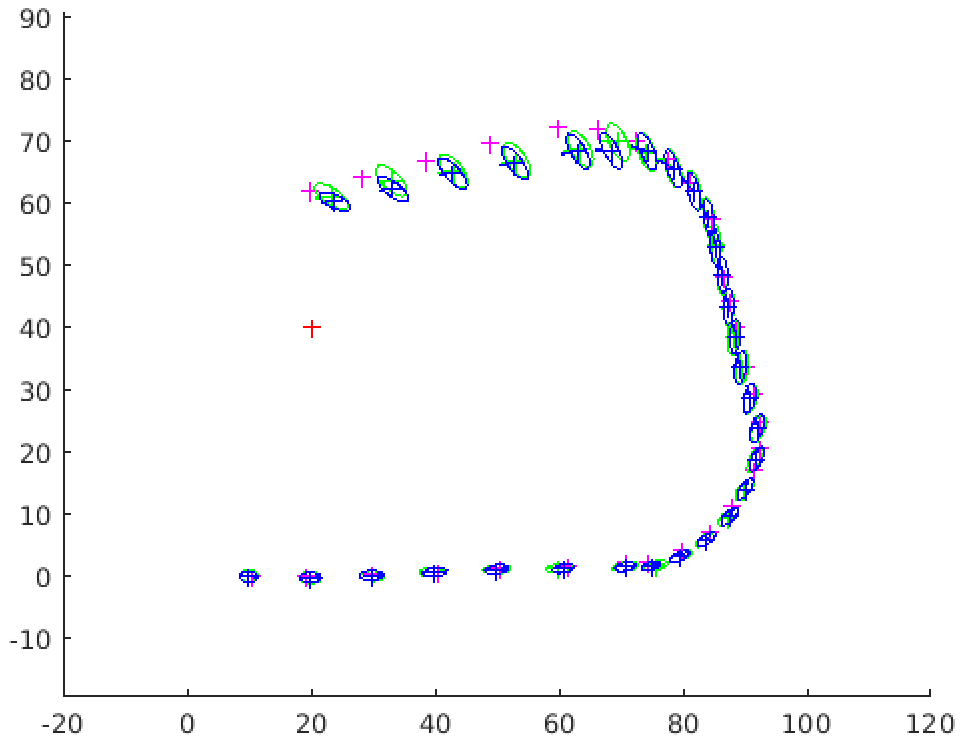
1. 1 beacon, bottom left



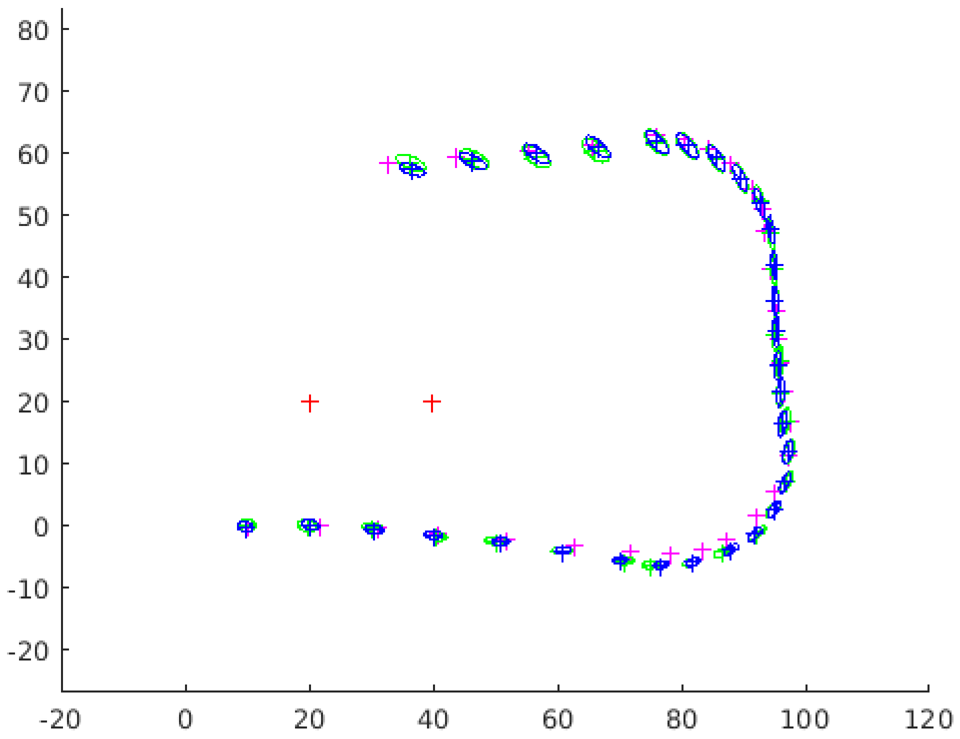
2. 1 beacon, upper mid right



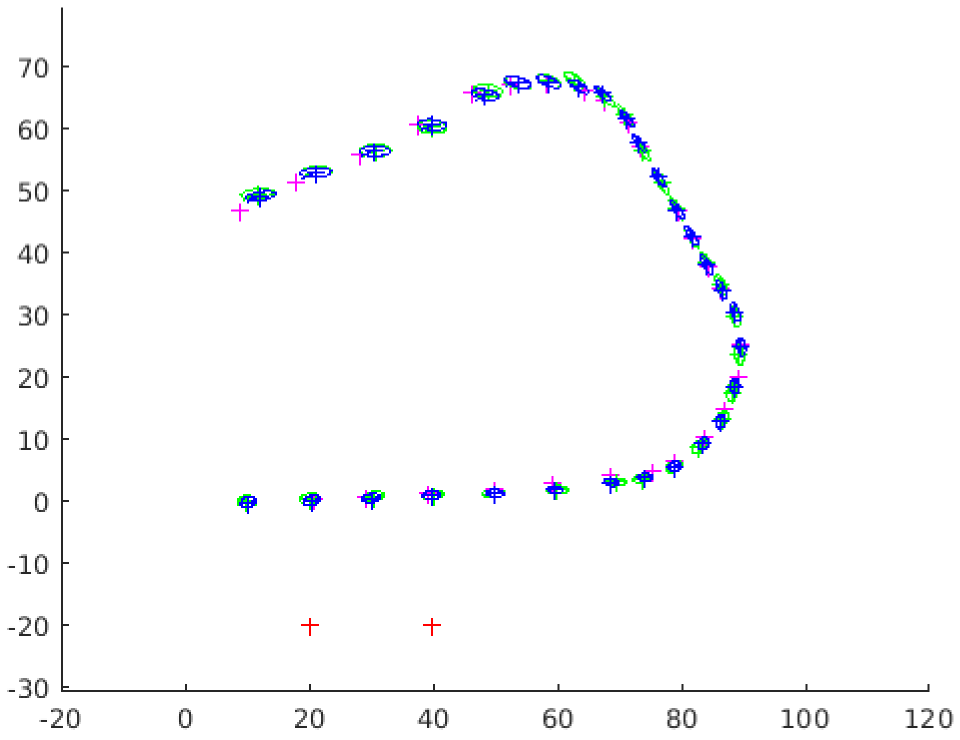
3. 1 beacon, mid left.



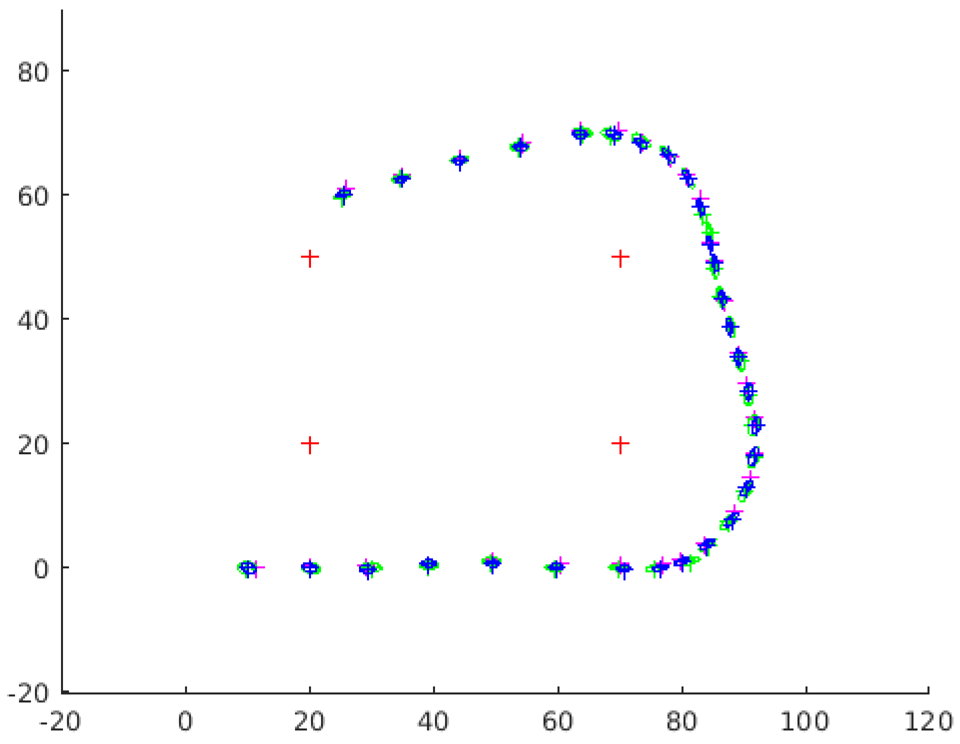
4. 2 beacons, lower mid left



5. 2 beacons, bottom



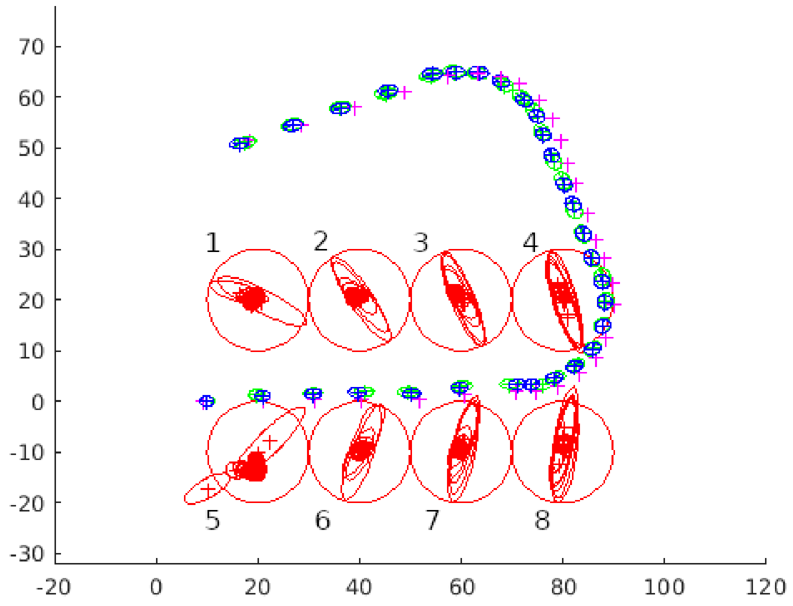
6. 4 beacons arranged for range dependent measurement error



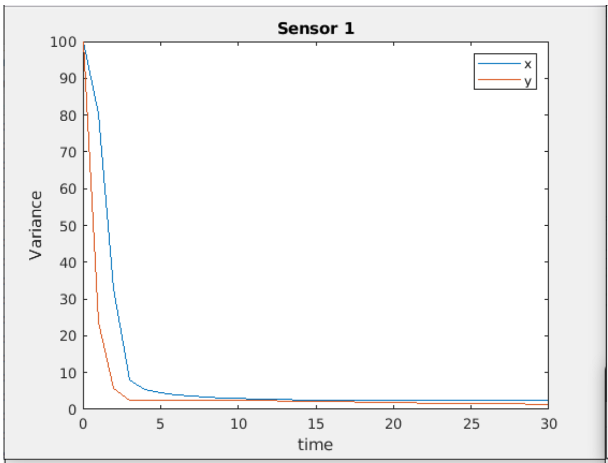
5

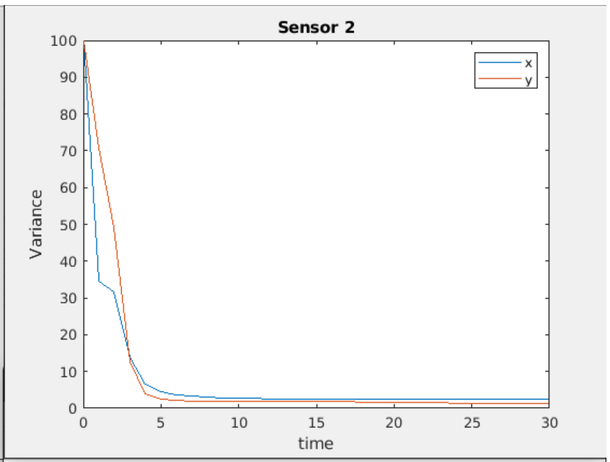
A

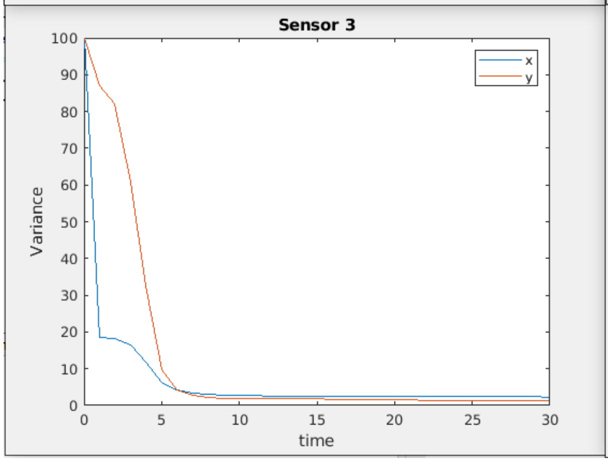
The below image shows the setup used. These are the original 8 beacons initially set up in the code.

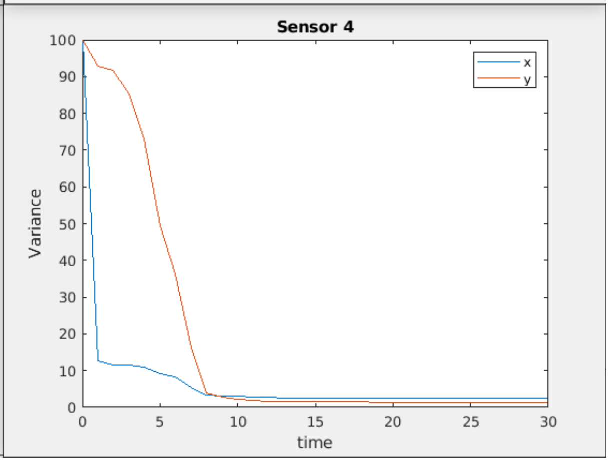


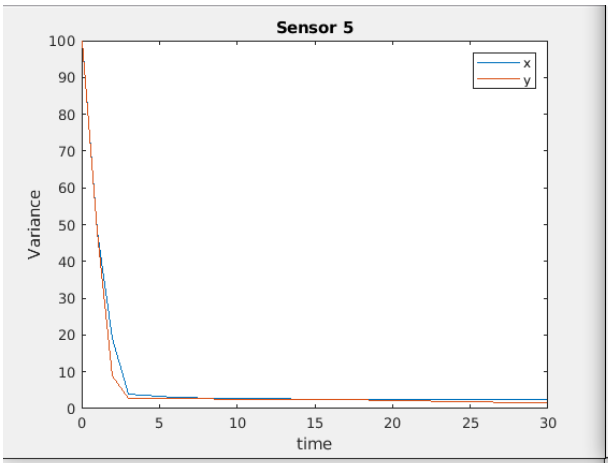
From the below plot, we can see that the found in a somewhat exponential fashion, and once it's "found" it's not made more uncertain.
What these graphs also show is that the beacons are found faster when the robot is closer to them. This is indicated by beacons 3,4,7 and 8 taking significantly longer than 1,2,5 and 6. We can see when the robot approaches the beacons as the uncertainty drastically decreases. This is further emphasised by the final plot in this section where the y component is never found because from where the robot is there is no definitive way to know this when there is uncertainty in the measurement. However they will eventually be found over many measurements. These ones are found more linearly.
So the limiting factors are the distance between the robot and the beacons and time.

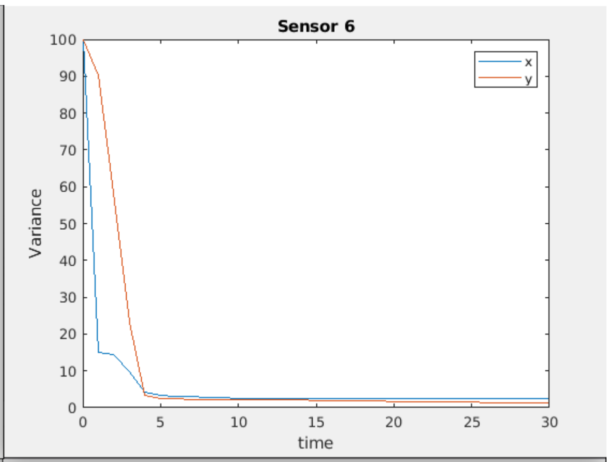

Sensor 1
Variance
time
x
y
0
5
10
15
20
25
30
100
90
80
70
60
50
40
30
20
10

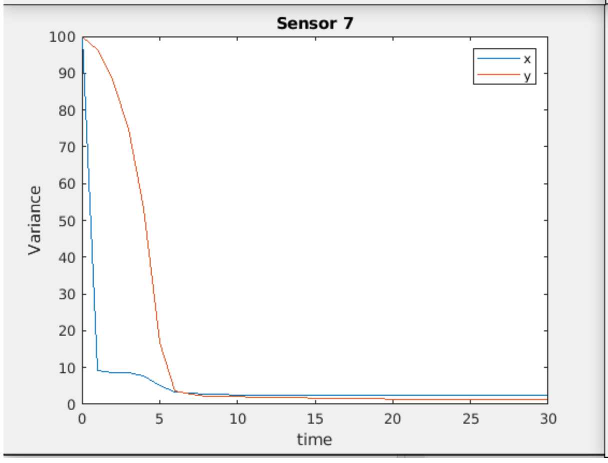

Sensor 2
Variance
time
x
y
0
5
10
15
20
25
30
100
90
80
70
60
50
40
30
20
10

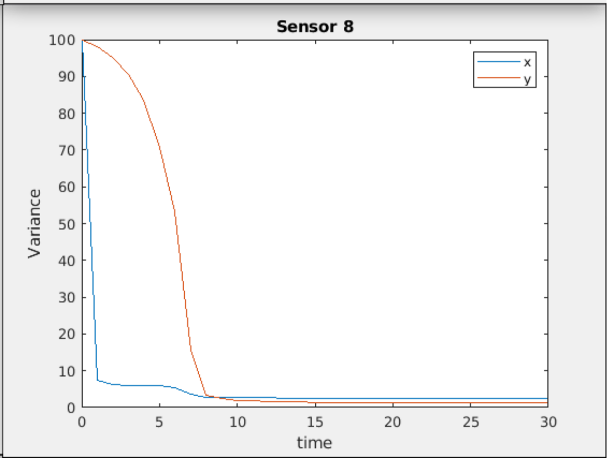

Sensor 3
Variance
time
x
y
0
5
10
15
20
25
30
100
90
80
70
60
50
40
30
20
10

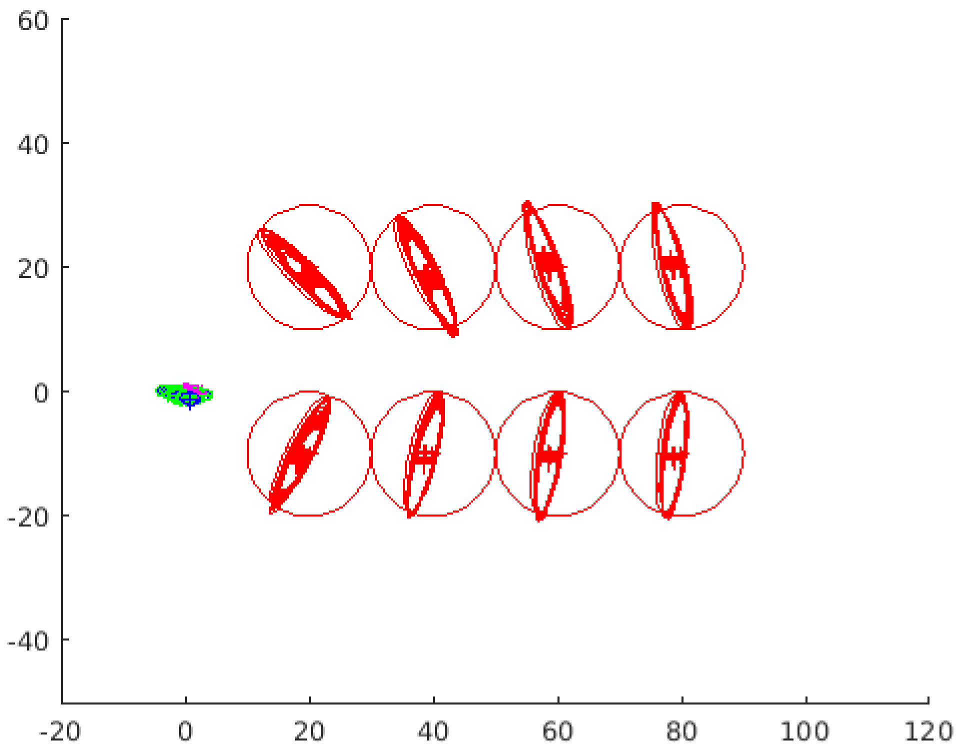

Sensor 4
Variance
time
x
y
0
5
10
15
20
25
30
100
90
80
70
60
50
40
30
20
10


Sensor 5
Variance
time
x
y
0
5
10
15
20
25
30
100
90
80
70
60
50
40
30
20
10

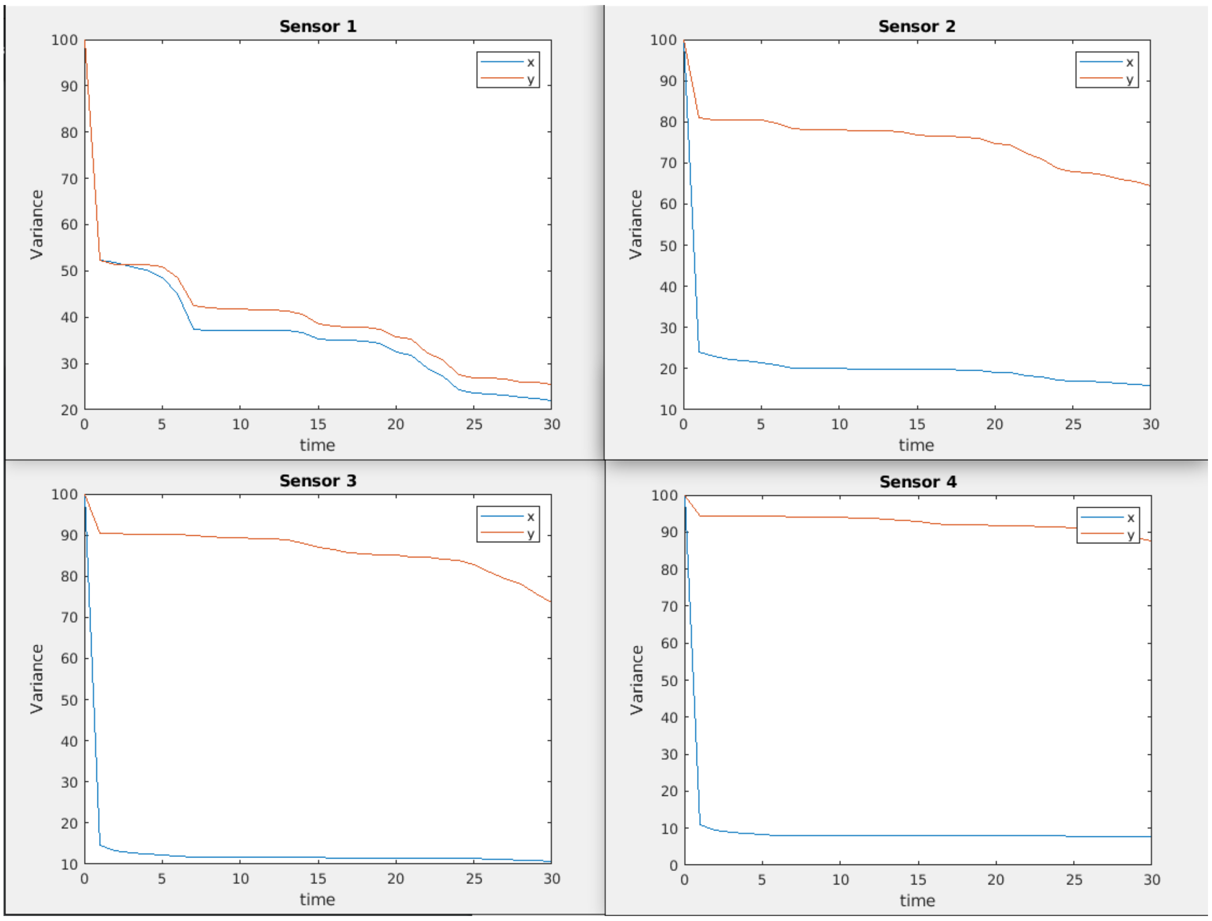

Sensor 6
Variance
time
x
y
0
5
10
15
20
25
30
100
90
80
70
60
50
40
30
20
10


Sensor 7
Variance
time
x
y
0
5
10
15
20
25
30
100
90
80
70
60
50
40
30
20
10


Sensor 8
Variance
time
x
y
0
5
10
15
20
25
30
100
90
80
70
60
50
40
30
20
10



60
40
20
0
-20
-40
-20
0
20
40
60
80
100
120



Sensor 1
Sensor 2
Sensor 3
Sensor 4
Variance
time
x
y

B

In the case of a single landmark, the robot should be programmed to drive towards where it thinks the landmark is.
As more landmarks are added, some path planning is required depending on where the landmarks are thought to be located, but the robot can drive towards them one by one or start driving in between them, because what matters is being able to get multiple measurements from different angles to narrow down where the landmarks are.