SLAM HOMEWORK 2

Sagar Sachdev (AndrewID: sagarsac)

Discussed with Troy Vicsik (AndrewID: tvicsik)

17 March 2022

1 Theory

1.1

Given:

$$P_t = \begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} \tag{1}$$

$$P_t = g(u_t, P_{t-1}) \tag{2}$$

$$=> P_{t+1} = g(u_{t+1}, P_t) \tag{3}$$

As a result, the matrix can be written as:

$$P_{t+1} = \begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{pmatrix} = \begin{pmatrix} x_t + d_t \cdot cos\,(\theta_t) \\ y_t + d_t \cdot sin\,(\theta_t) \\ \theta_t + \alpha_t \end{pmatrix} \tag{4}$$

This can be written as:

$$P_{t+1} = \begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{pmatrix} = \begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} + \begin{pmatrix} d_t \cdot cos\,(\theta_t) \\ d_t \cdot sin\,(\theta_t) \\ \alpha_t \end{pmatrix} \tag{5}$$

1.2

Given:

$$e_x \sim \mathcal{N}(0,\, \sigma_x^2)\,.$$

$$e_y \sim \mathcal{N}(0,\, \sigma_y^2)\,.$$

$$e_\alpha \sim \mathcal{N}(0,\, \sigma_\alpha^2)\,.$$

Computing the Jacobian (A) Matrix, we get:

$$\text{Jacobian (A)} = \begin{pmatrix} \frac{\partial}{\partial x_t}(x_{t+1}) & \frac{\partial}{\partial y_t}(x_{t+1}) & \frac{\partial}{\partial \theta_t}(x_{t+1}) \\ \frac{\partial}{\partial x_t}(y_{t+1}) & \frac{\partial}{\partial y_t}(y_{t+1}) & \frac{\partial}{\partial \theta_t}(y_{t+1}) \\ \frac{\partial}{\partial x_t}(\theta_{t+1}) & \frac{\partial}{\partial y_t}(\theta\ _{t+1}) & \frac{\partial}{\partial \theta_t}(\theta_{t+1}) \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & 0 & -d_t \cdot sin\,(\theta_t) \\ 0 & 1 & d_t \cdot cos\,(\theta_t) \\ 0 & 0 & 1 \end{pmatrix} \tag{6}$$

Let L = Rotation Matrix about z axis
Therefore,

$$L = \begin{pmatrix} cos\,(\theta) & -sin\,(\theta) & 0 \\ sin\,(\theta) & cos\,(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_{t+1} = \begin{pmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\alpha^2 \end{pmatrix}$$

To account for the frames, the following matrix multiplication is performed:

$$LR_{t+1}L^T$$

Finally the Covariance at time t+1 can be written as:

$$\overline{\Sigma}_{t+1} = A_{t+1}\overline{\Sigma}_t {A_{t+1}}^T + LR_{t+1}L^T \tag{7}$$

Thus the normal distribution with 0 mean can be written as:

$$\mathcal{N}(0,\, (A_{t+1}\overline{\Sigma}_t {A_{t+1}}^T + LR_{t+1}L^T))\,.$$

1.3

Total bearing angle = $\beta + n_\beta$
Total radius = $r + n_r$

$$l_x = x_t + (r + n_r)cos(\beta + n_\beta + \theta_t) \tag{8}$$
$$l_y = y_t + (r + n_r)sin(\beta + n_\beta + \theta_t) \tag{9}$$

1.4

From (8) we know:

$$l_x - x_t = (r + n_r) cos(\beta + n_\beta + \theta_t)$$

$$r + n_r = \frac{l_x - x_t}{cos\left(\beta + n_\beta + \theta_t\right)} \tag{10}$$

Similarly, from (9) we know:

$$l_y - y_t = (r + n_r) sin(\beta + n_\beta + \theta_t)$$

$$r + n_r = \frac{l_y - y_t}{sin\left(\beta + n_\beta + \theta_t\right)} \tag{11}$$

Equating (10) and (11), we get:

$$\frac{l_x - x_t}{cos\left(\beta + n_\beta + \theta_t\right)} = \frac{l_y - y_t}{sin\left(\beta + n_\beta + \theta_t\right)}$$

$$tan\left(\beta + n_\beta + \theta_t\right) = \frac{l_y - y_t}{l_x - x_t}$$

$$\beta + n_\beta + \theta_t = np.arctan2(l_y - y_t, l_x - x_t)$$

$$\beta = np.arctan2(l_y - y_t, l_x - x_t) - n_\beta - \theta_t$$

To ensure that beta is in the range $(-\pi, \pi]$, we can use the warp2pi function. Therefore:

$$\beta = warp2pi(np.arctan2(l_y - y_t, l_x - x_t) - n_\beta - \theta_t) \tag{12}$$

Similarly we know that:

$$(l_x - x_t)^2 + (l_y - y_t)^2 = (r + n_r)^2$$

$$r + n_r = \sqrt{(l_x - x_t)^2 + (l_y - y_t)^2}$$

$$r = \sqrt{(l_x - x_t)^2 + (l_y - y_t)^2} - n_r \tag{13}$$

1.5

$$\begin{pmatrix} r \\ \beta \end{pmatrix} = \begin{pmatrix} \sqrt{(l_x - x_t)^2 + (l_y - y_t)^2} - n_r \\ warp2pi(np.arctan2\left(l_y - y_t, l_x - x_t\right) - n_\beta \ - \ \theta_t) \end{pmatrix}$$

We need to compute Jacobian matrix (H_p) with respect to robot poses. This can be written as follows:

$$H_p = \begin{pmatrix} \frac{\partial}{\partial x_t}(r) & \frac{\partial}{\partial y_t}(r) & \frac{\partial}{\partial \theta_t}(r) \\ \frac{\partial}{\partial x_t}(\beta) & \frac{\partial}{\partial y_t}(\beta) & \frac{\partial}{\partial \theta_t}(\beta) \end{pmatrix}$$

Therefore, the H_p can be written as:

$$H_p = \begin{pmatrix} \frac{(-l_x+x_t)}{\sqrt{(l_x-x_t)^2+(l_y-y_t)^2}} & \frac{(-l_y+y_t)}{\sqrt{(l_x-x_t)^2+(l_y-y_t)^2}} & 0 \\ \frac{(l_y-y_t)}{\sqrt{(l_x-x_t)^2+(l_y-y_t)^2}} & \frac{(-l_x+x_t)}{\sqrt{(l_x-x_t)^2+(l_y-y_t)^2}} & -1 \end{pmatrix} \tag{14}$$

1.6

To find the measurement Jacobian (H_l) using its corresponding landmark:

$$H_l = \begin{pmatrix} \frac{\partial}{\partial l_x}(r) & \frac{\partial}{\partial l_y}(r) \\ \frac{\partial}{\partial l_x}(\beta) & \frac{\partial}{\partial l_y}(\beta) \end{pmatrix}$$

$$H_l = \begin{pmatrix} \frac{(l_x-x_t)}{\sqrt{(l_x-x_t)^2+(l_y-y_t)^2}} & \frac{(l_y-y_t)}{\sqrt{(l_x-x_t)^2+(l_y-y_t)^2}} \\ \frac{(-l_y+y_t)}{(l_x-x_t)^2+(l_y-y_t)^2} & \frac{(l_x-x_t)}{(l_x-x_t)^2+(l_y-y_t)^2} \end{pmatrix} \tag{15}$$

2 Implementation and Evaluation

2.1

The number of observed landmarks over the entire sequence is 6.

2.2

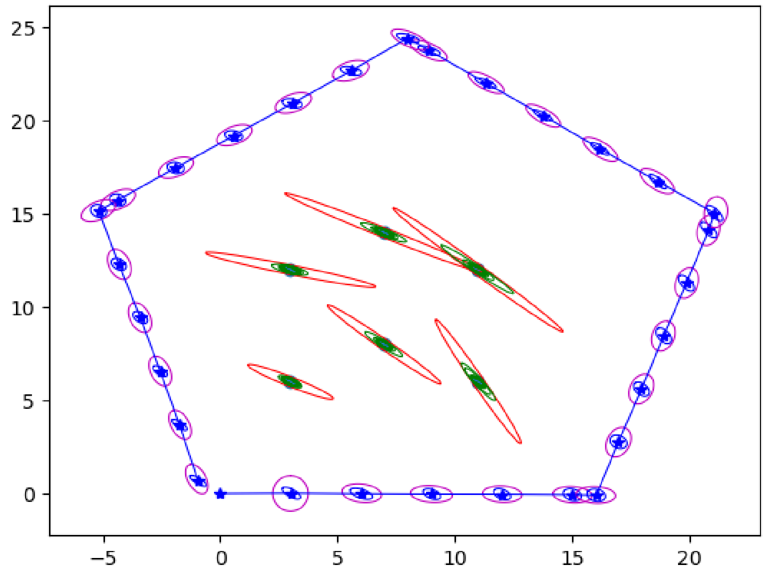


Figure 1: Visualization

2.3

In the above visualization, with every iteration, the measurement of the landmarks relative to the robot position is changing and this induces some errors which the Kalman gains account for. This can be seen by the sizes of the uncertainty ellipses with the blue ellipses remaining somewhat the same size and the magenta ellipses in general decreasing in size highlighting lower uncertainties and thus improved estimation of the trajectory. Similarly the green ellipses are expected to be bounded by the red ellipses and thus depict the Kalman gains accounting for large covariances in terms of the landmarks and thus an improved estimation of the map.

2.4

Euclidean Distances = [0.01358596,0.02072718,0.00875565,0.01981848,0.01145865,0.01968745]
Mahalanobis Distances = [0.00114841,0.00169644,0.00101575,0.00202432,0.00124972,0.00231084]

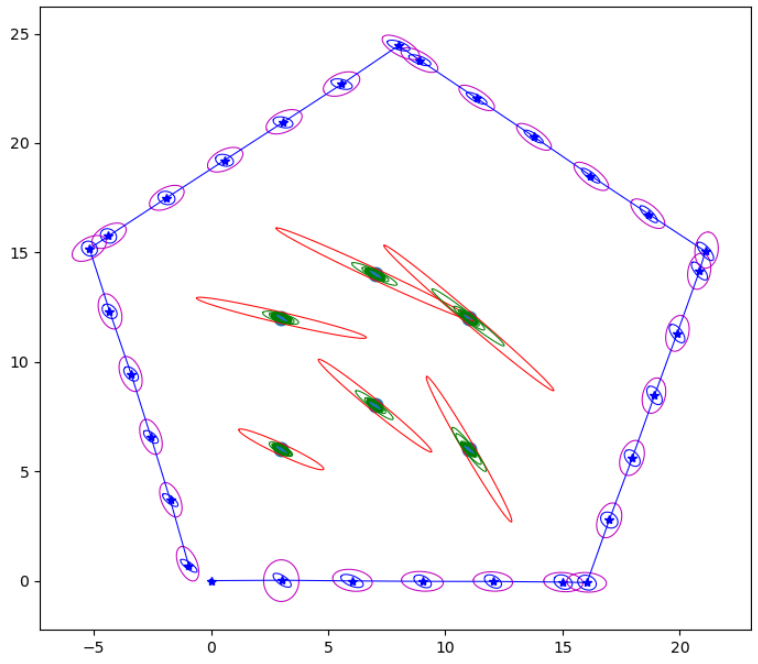


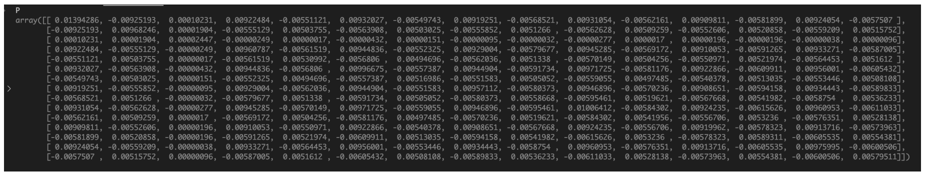
Figure 2: Showing landmarks more prominently

Yes each one of the landmarks are inside the smallest corresponding ellipse. This implies that the 3-σ ellipse is a great estimate for the landmark positions.

From the numbers it can be seen that the Mahalanobis distance is always smaller than the Euclidean distance. Overall the errors are quite small and this shows that the EKF is a great tool to eliminate noise and provide an accurate estimate of the landmark positions.

3 Discussion

3.1



```
P
array([[ 0.01394286, -0.00925193,  0.00010231,  0.00922484, -0.00551121,  0.00932027, -0.00549743,  0.00919251, -0.00568521,  0.00931054, -0.00562161,  0.00909811, -0.00581899,  0.00924054, -0.0057507 ],
       [-0.00925193,  0.00968246,  0.00001904, -0.00555129,  0.00503755, -0.00563908,  0.00503025, -0.00555852,  0.0051266 , -0.00562628,  0.00509259, -0.00552606,  0.00520858, -0.00559209,  0.00515752],
       [ 0.00010231,  0.00001904,  0.00002447, -0.00000249,  0.00000017, -0.00000432,  0.00000151, -0.00000095, -0.00000032, -0.00000277,  0.0000017 ,  0.00000196, -0.00000196, -0.00000038,  0.00000096],
       [ 0.00922484, -0.00555129, -0.00000249,  0.00960787, -0.00561519,  0.00944836, -0.00552325,  0.00929004, -0.00579677,  0.00945285, -0.00569172,  0.00910053, -0.00591265,  0.00933271, -0.00587005],
       [-0.00551121,  0.00503755,  0.00000017, -0.00561519,  0.00530992, -0.0056806 ,  0.00494696, -0.00562036,  0.0051338 , -0.00570149,  0.00504256, -0.00550971,  0.00521974, -0.00564453,  0.0051612 ],
       [ 0.00932027, -0.00563908, -0.00000432,  0.00944836, -0.0056806 ,  0.00996675, -0.00557387,  0.00944904, -0.00591734,  0.00971725, -0.00581176,  0.00922866, -0.00609911,  0.00956001, -0.00605432],
       [-0.00549743,  0.00503025,  0.00000151, -0.00552325,  0.00494696, -0.00557387,  0.00516986, -0.00551583,  0.00505052, -0.00559055,  0.00497485, -0.00540378,  0.00513035, -0.00553446,  0.00508108],
       [ 0.00919251, -0.00555852, -0.00000095,  0.00929004, -0.00562036,  0.00944904, -0.00551583,  0.00957112, -0.00580373,  0.00946896, -0.00570236,  0.00908651, -0.00594158,  0.00934443, -0.00589833],
       [-0.00568521,  0.0051266 , -0.00000032, -0.00579677,  0.0051338 , -0.00591734,  0.00505052, -0.00580373,  0.00558668, -0.00595461,  0.00519621, -0.00567668,  0.00541982, -0.0058754 ,  0.00536233],
       [ 0.00931054, -0.00562628, -0.00000277,  0.00945285, -0.00570149,  0.00971725, -0.00559055,  0.00946896, -0.00595461,  0.01006412, -0.00584302,  0.00924235, -0.00615626,  0.00960953, -0.00611033],
       [-0.00562161,  0.00509259,  0.0000017 , -0.00569172,  0.00504256, -0.00581176,  0.00497485, -0.00570236,  0.00519621, -0.00584302,  0.00541956, -0.00556706,  0.0053236 , -0.00576351,  0.00528138],
       [ 0.00909811, -0.00552606,  0.00000196,  0.00910053, -0.00550971,  0.00922866, -0.00540378,  0.00908651, -0.00567668,  0.00924235, -0.00556706,  0.00919962, -0.00578323,  0.00913716, -0.00573963],
       [-0.00581899,  0.00520858, -0.00000196, -0.00591265,  0.00521974, -0.00609911,  0.00513035, -0.00594158,  0.00541982, -0.00615626,  0.0053236 , -0.00578323,  0.00589311, -0.00605535,  0.00554381],
       [ 0.00924054, -0.00559209, -0.00000038,  0.00933271, -0.00564453,  0.00956001, -0.00553446,  0.00934443, -0.0058754 ,  0.00960953, -0.00576351,  0.00913716, -0.00605535,  0.00975995, -0.00600506],
       [-0.0057507 ,  0.00515752,  0.00000096, -0.00587005,  0.0051612 , -0.00605432,  0.00508108, -0.00589833,  0.00536233, -0.00611033,  0.00528138, -0.00573963,  0.00554381, -0.00600506,  0.00579511]])
```

Figure 3: Final P Matrix

The zero terms in P get updated since we are applying an update after the initial EKF iteration. The assumption we are making is that the initial covariance between the robot position and the landmarks is 0. This is incorrect because we are using the robot's initial state/position to approximate the locations of the landmarks.

3.2

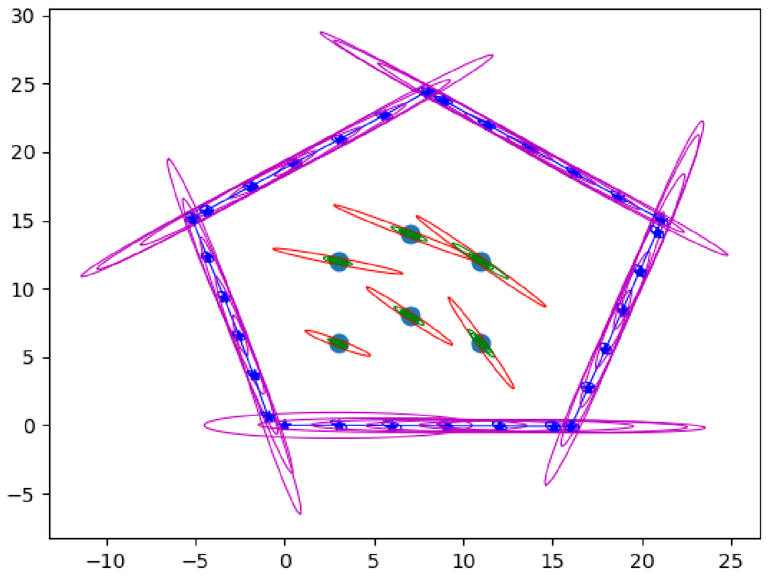


Figure 4: 10 times σ_x

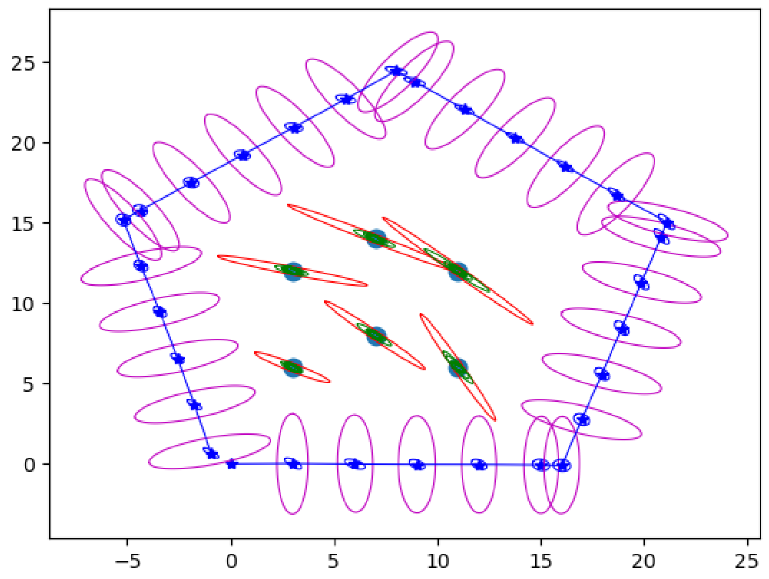


Figure 5: 10 times σ_y

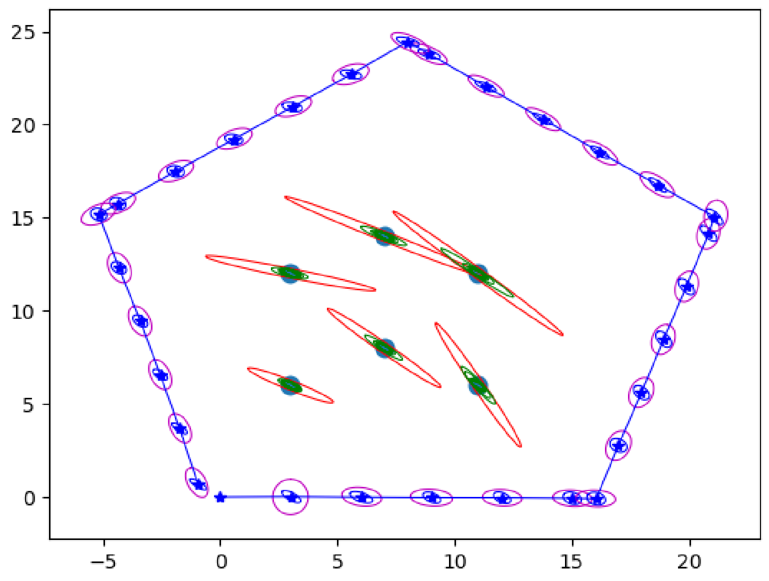


Figure 6: 10 times σ_α

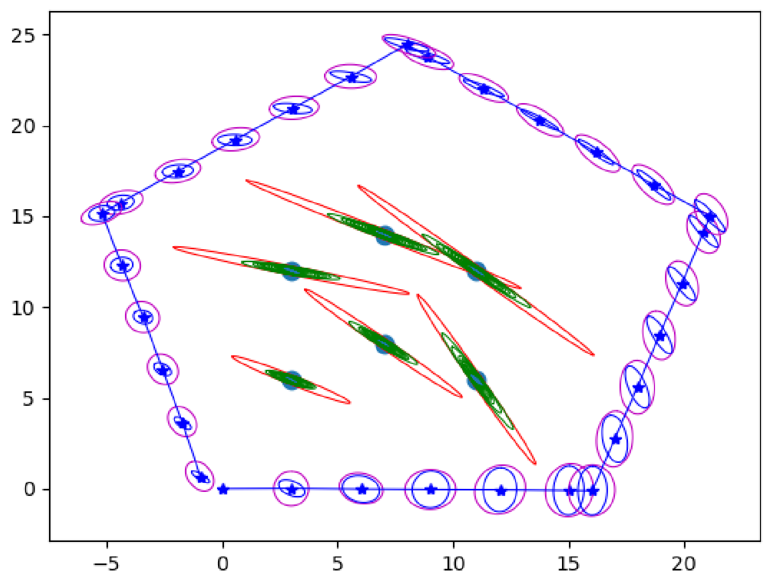


Figure 7: 10 times σ_b

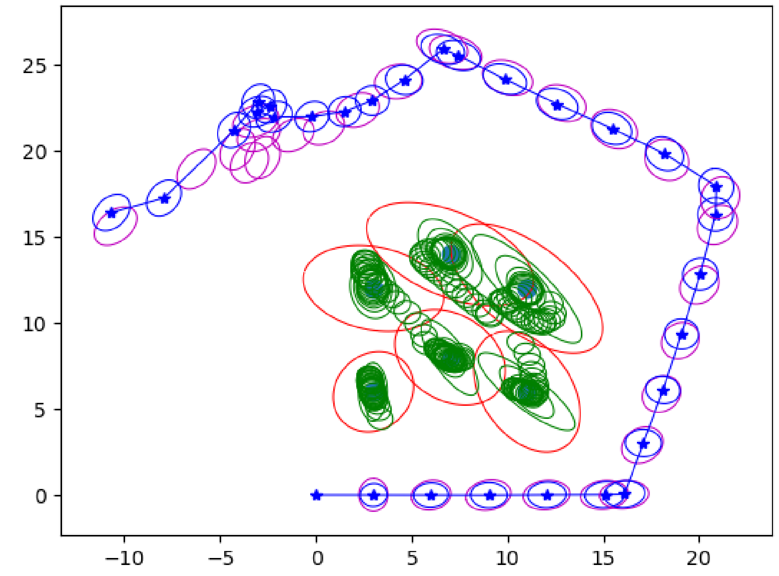


Figure 8: 10 times σ_r

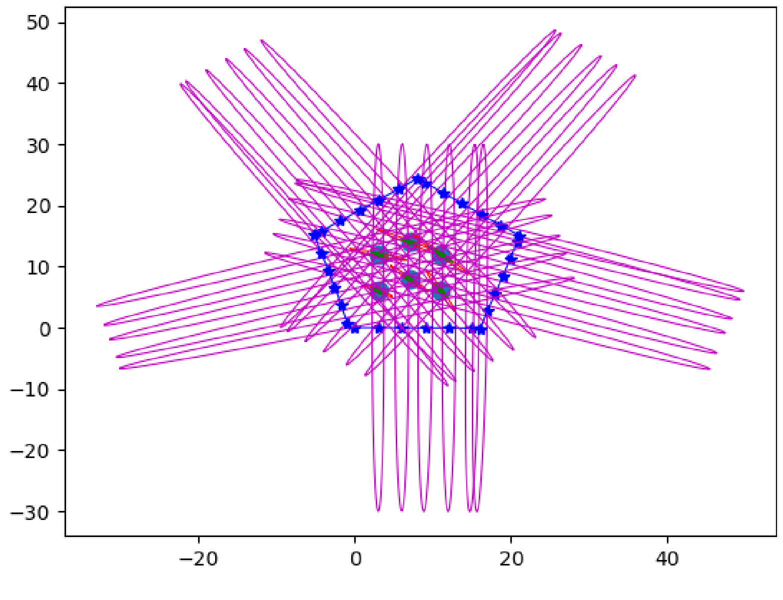


Figure 9: 100 times σ_y

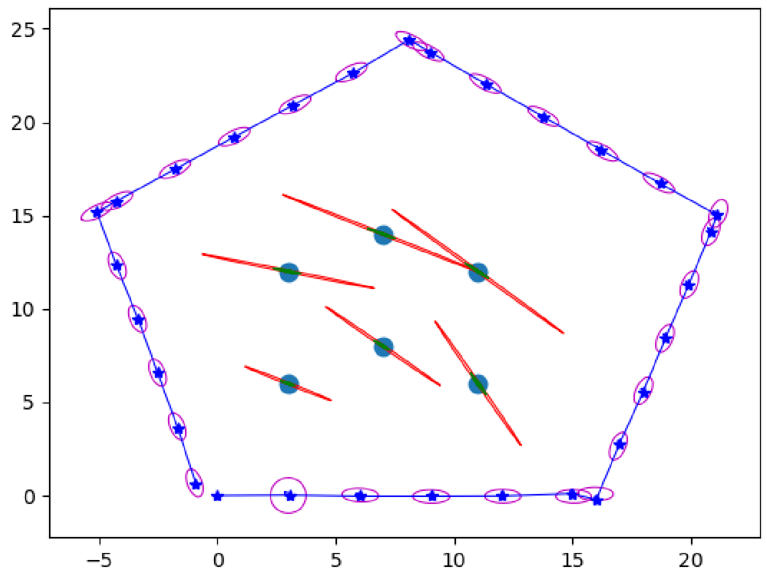


Figure 10: 0.01 times σ_r

From the above figures it can be seen that when the uncertainty parameters are multiplied by 10, the magenta ellipses become more larger in both the σ_x and in the σ_y scenarios. However the magenta ellipses become smaller in the σ_α and in the σ_b directions and in the σ_r direction the "pentagon" shape is not completed arising from huge uncertainties. I also tested out some extreme cases where I multiplied σ_y by 100 and noticed the magenta ellipses to be even more stretched whereas multiplying σ_r resulted in the landmarks uncertainties in red being stretched out.

3.3

Some solutions that could be explored to make EKF SLAM framework faster are: 1) Use of multi-threading on a single core on a computer 2) Multi-core processing on a multi-core computer 3) The algorithm could possibly be written in C which compiles much faster than Python 4) Limit the use of "for" loops in the python code 5) All the landmarks do not need to be saved, especially for computations involving circular patterns. This is because landmarks would circle back to their original locations as time proceeds and the older stored landmarks are not very helpful. This could be implemented by overwriting landmarks that have not been called in a certain number of iterations.

4 References

1) Probabilistic Robotics textbook
2) 16-833 Lecture Notes
3) My classmate Troy Vicsik helped me troubleshoot/debug some of my code