ROBOT LOCALIZATION AND MAPPING

16-833

HOMEWORK - 2

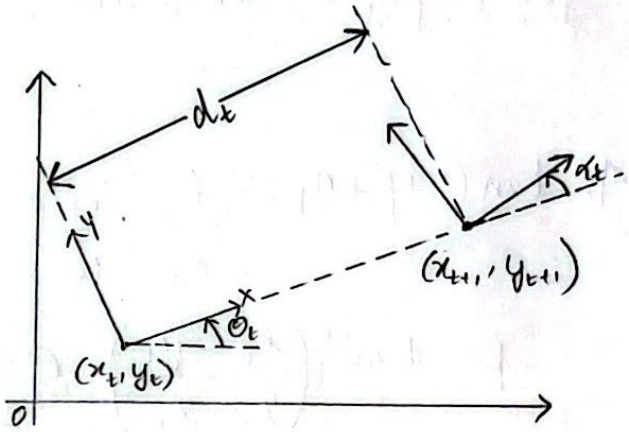
Name: Akshay Bodagabettu
Andrew ID: abadagab

1. Theory

1. Assuming no noise in the control system,

$$x_{t+1} = g(x_t, u_{t+1})$$

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t + d_t \cos\theta_t \\ y_t + d_t \sin\theta_t \\ \theta_t + \alpha_t \end{bmatrix} \begin{matrix} \rightarrow g_1 \\ \rightarrow g_2 \\ \rightarrow g_3 \end{matrix}$$



2. The predicted uncertainty of the robot at time t+1 is nothing but the covariance matrix at time t+1

$$\Sigma_{t+1} = G_t \Sigma_t G_t^T + T R_{t+1} T^T$$

where, $G_t = \frac{\partial g}{\partial x} = \begin{bmatrix} \partial g_1/\partial x & \partial g_1/\partial y & \partial g_1/\partial \theta \\ \partial g_2/\partial x & \partial g_2/\partial y & \partial g_2/\partial \theta \\ \partial g_3/\partial x & \partial g_3/\partial y & \partial g_3/\partial \theta \end{bmatrix} = \begin{bmatrix} 1 & 0 & -d_t \sin\theta_t \\ 0 & 1 & d_t \cos\theta_t \\ 0 & 0 & 1 \end{bmatrix}$

$\Sigma_t \rightarrow$ Covariance matrix at time t

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

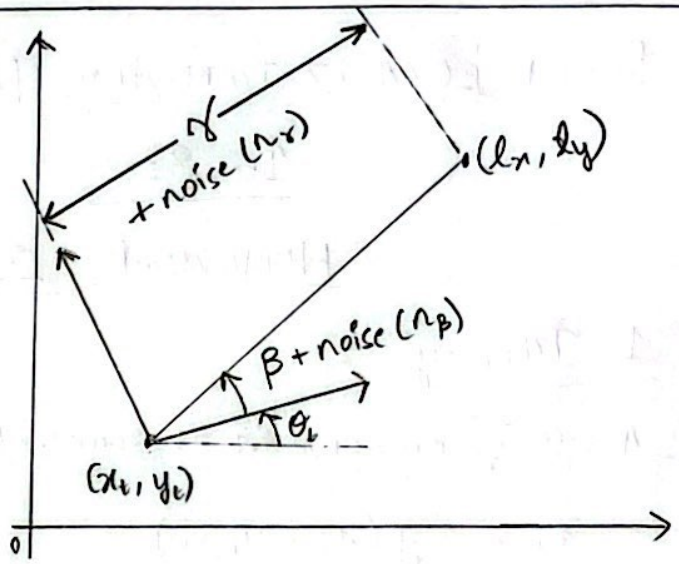
As this is in the robot frame, to bring it to world frame, we multiply it by an SE(2) transformation matrix (T)

$$T = \begin{bmatrix} \cos\theta_t & -\sin\theta_t & 0 \\ \sin\theta_t & \cos\theta_t & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Σ_{t+1} can now be calculated by plugging in these matrices.

3. The estimated position of landmark l in the global frame is

$$\begin{bmatrix} l_x \\ l_y \end{bmatrix} = \begin{bmatrix} x_t + (r+n_r)\cos(\theta+\beta+n_\beta) \\ y_t + (r+n_r)\sin(\theta+\beta+n_\beta) \end{bmatrix}$$



4. $$\tan(\theta+\beta+n_\beta) = \frac{l_y - y_t}{l_x - x_t}$$

$$\beta = \tan^{-1}\left(\frac{l_y - y_t}{l_x - x_t}\right) - \theta - n_\beta = \text{warp2pi}(np.\arctan2(l_y - y_t, l_x - x_t) - \theta - n_\beta$$

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

$$\begin{bmatrix} \beta \\ r \end{bmatrix} = \begin{bmatrix} \text{warp2pi}(np.\arctan2(l_y - y_t, l_x - x_t) - \theta - n_\beta) \\ \sqrt{(l_x - x_t)^2 + (l_y - y_t)^2} - n_r \end{bmatrix}$$

5. The measurement Jacobian H_p with respect to robot pose can be represented as

$$H_p = \begin{bmatrix} \partial\beta/\partial x & \partial\beta/\partial y & \partial\beta/\partial\theta \\ \partial r/\partial x & \partial r/\partial y & \partial r/\partial\theta \end{bmatrix}$$

$$H_p = \begin{bmatrix} \dfrac{l_y - y_t}{(l_x - x_t)^2 + (l_y - y_t)^2} & \dfrac{x_t - l_x}{(l_x - x_t)^2 + (l_y - y_t)^2} & -1 \\ \dfrac{x_t - l_x}{\sqrt{(l_x - x_t)^2 + (l_y - y_t)^2}} & \dfrac{y_t - l_y}{\sqrt{(l_x - x_t)^2 + (l_y - y_t)^2}} & 0 \end{bmatrix}$$

6. The measurement Jacobian H_ℓ with respect to its corresponding landmark ℓ is

$$H_\ell = \begin{bmatrix} \partial\beta/\partial\ell_x & \partial\beta/\partial\ell_y \\ \partial r/\partial\ell_x & \partial r/\partial\ell_y \end{bmatrix}$$

$$H_\ell = \begin{bmatrix} \dfrac{y_t - \ell_y}{(\ell_x - x_t)^2 + (\ell_y - y_t)^2} & \dfrac{\ell_x - x_t}{(\ell_x - x_t)^2 + (\ell_y - y_t)^2} \\ \dfrac{\ell_x - x_t}{\sqrt{(\ell_x - x_t)^2 + (\ell_y - y_t)^2}} & \dfrac{\ell_y - y_t}{\sqrt{(\ell_x - x_t)^2 + (\ell_y - y_t)^2}} \end{bmatrix}$$

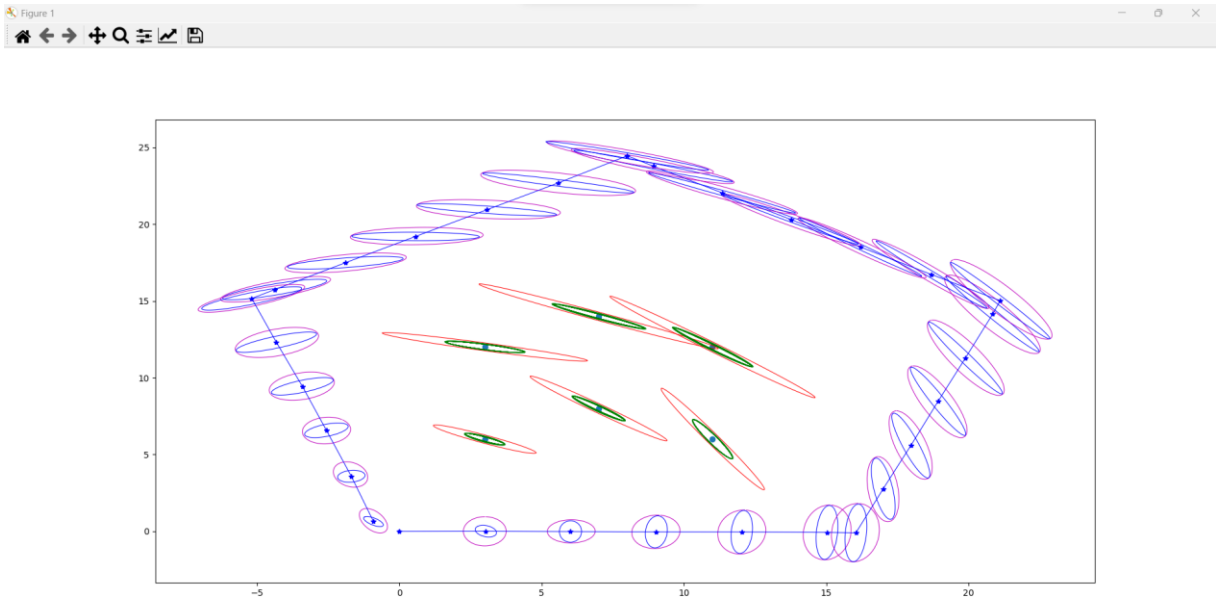
We assume that all the landmarks are independent of each other. This is why we only need to calculate the measurement Jacobian with respect to only the corresponding landmark and not all other landmarks

Name: Akshay Badagabettu
Andrew ID: abadagab

IMPLEMENTATION AND EVALUATION

PART 2

1. There are six fixed landmarks being observed over the entire sequence. I determined this by counting the number of logs in the data log file. There were 12 records, which correspond to beta and range values for each landmark. Also, the value of k was printed and checked in the init_landmarks part of the code.
2. The code was filled as requested and the final visualization with the given sigma values is shown below.



3. From the above figure, we see that the uncertainty of the robot position and landmark positions have been reduced. The robot starts at (0,0), observes the environment (6 landmarks in this case), and initializes the landmarks. It then moves according to the control input (3,0) in our case. This movement is not perfect and has noise associated with it. The corresponding covariance ellipse is shown by the magenta line. The robot then observes the environment again (the same six landmarks in our case). The EKF algorithm then uses this previously mapped landmark to correct both its self-localization and localization of the landmarks. Hence, the covariance matrix and state vector keep getting updated as the robot keeps moving and observing the landmarks. In the figure attached above, we see that the covariance ellipse (blue and green ellipses) has reduced from the initial covariance ellipse (magenta and red ellipses). This shows that the EKF algorithm is working well.
4. The ground truth landmark positions were plotted, and all of them are inside their smallest green ellipse. This means that the EKF algorithm has managed to localize the position of the landmarks very well.
   The Euclidean and Mahalanobis distance has been computed and attached below.

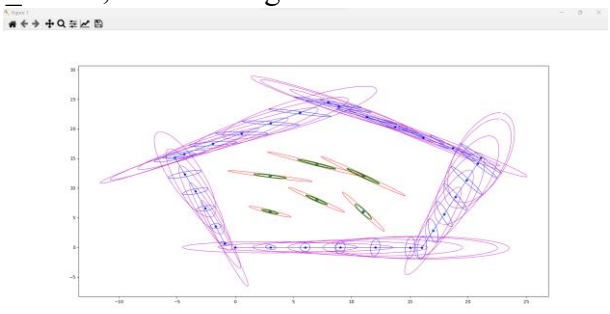
Name: Akshay Badagabettu
Andrew ID: abadagab

```
The Euclidean distance for landmark 1 is:  0.0024154708370622657
The Mahalanobis distance for landmark 1 is:  0.05452351374945245
The Euclidean distance for landmark 2 is:  0.005808771104541731
The Mahalanobis distance for landmark 2 is:  0.06477658562615622
The Euclidean distance for landmark 3 is:  0.001410751094012391
The Mahalanobis distance for landmark 3 is:  0.03507635123882815
The Euclidean distance for landmark 4 is:  0.00324188011860893
The Mahalanobis distance for landmark 4 is:  0.06410328613395173
The Euclidean distance for landmark 5 is:  0.001782335072325175
The Mahalanobis distance for landmark 5 is:  0.021948929144618753
The Euclidean distance for landmark 6 is:  0.005564322137017677
The Mahalanobis distance for landmark 6 is:  0.09436064748531146
```

Both the Euclidean and Mahalanobis distances are very small, which tells us that the algorithm has localized the landmark positions quite accurately. The Euclidean distance is not really a good measure of knowing whether the localization was correct. This is because, there may exist a point P that is outside the covariance ellipse but with the same Euclidean distance as a point X which lies inside the covariance ellipse. The Mahalanobis distance solves this problem by changing the axes to the eigenvectors. This makes the axis directions to be somewhere along the bulges of the ellipse. Then the distance is calculated. Therefore, the Mahalanobis distance gives the distance between a point and a distribution very accurately. In our case, the Mahalanobis distance is also very low, which shows that the predicted location of the landmarks is very accurate.

PART 3 – DISCUSSION

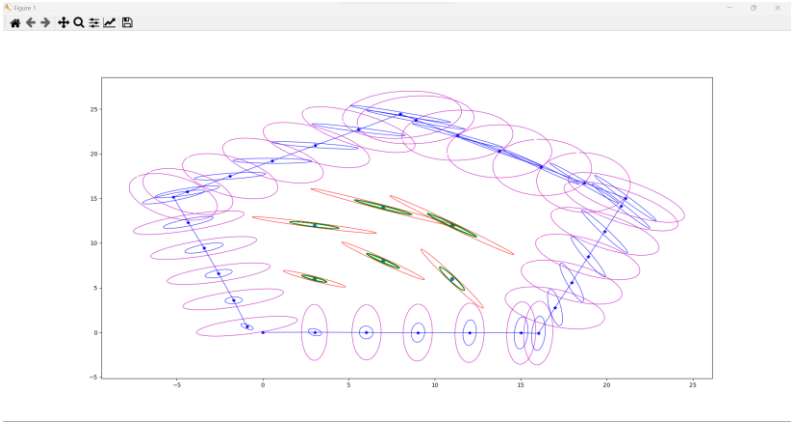
1. The zero term in the initial landmark covariance matrix became non-zero in the end because, during the computation of the Kalman Gain, the Kt matrix is a completely filled matrix and has no zero elements. This Kalman Gain is used to update the covariance matrix, which is why even the zero elements of the initial covariance matrix get altered.
   When setting the initial covariance matrix, we assumed that the state vector elements are independent of each other. In simple words, the poses are independent of each other and the landmarks, and even the landmarks are independent of each other and the robot pose. But this assumption is not correct. All the state vector elements are dependent on each other.
2. The parameters were changed one by one and the results were observed.
   a. Sigma_x = Sigma_x * 10, rest unchanged



We see that the robot pose uncertainty has increased in the x direction i.e, the covariance ellipse is wider but shorter. But still, the algorithm is able to localize its pose well, as seen by the blue ellipse.

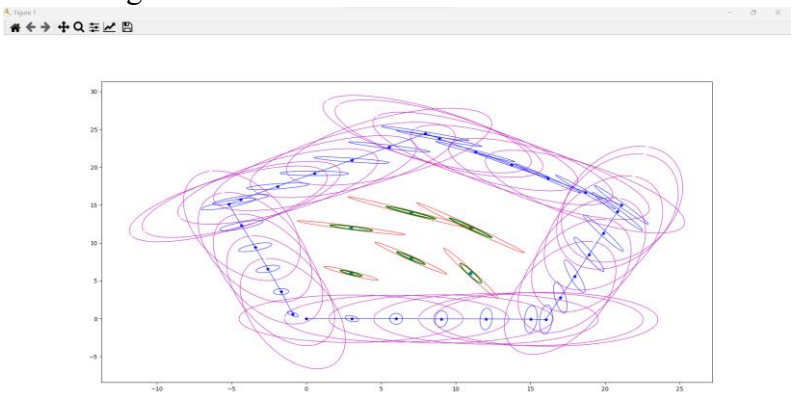
Name: Akshay Badagabettu
Andrew ID: abadagab

b. Sigma_y = Sigma_y * 10, rest unchanged



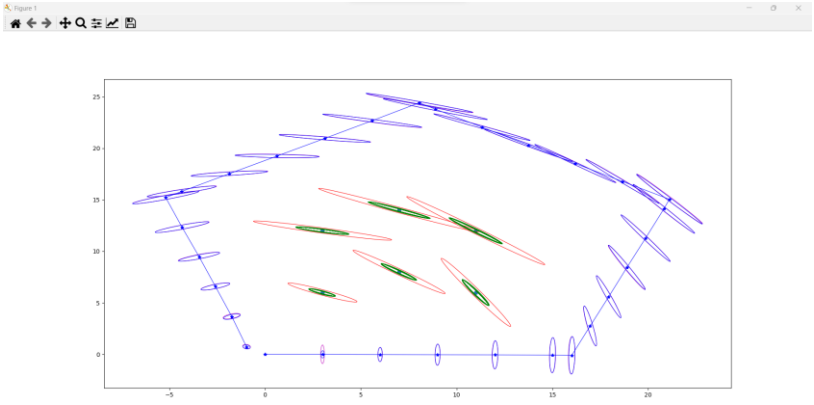
The uncertainty in the y-direction increases, and the covariance ellipse is taller but less wide. Still, the algorithm is able to localize its pose quite well.

c. Sigma_x = Sigma_x * 10, Sigma_y = Sigma_y * 10, Sigma_alpha = Sigma_alpha * 1000, rest unchanged



As expected, the covariance after the predict step is large, but during the update step, the algorithm is able to localize the pose well, and a smaller covariance ellipse is seen. The predicted uncertainty is so high because we are increasing the sigma values of the control parameters. This is increasing the values in the 'control_cov' matrix.

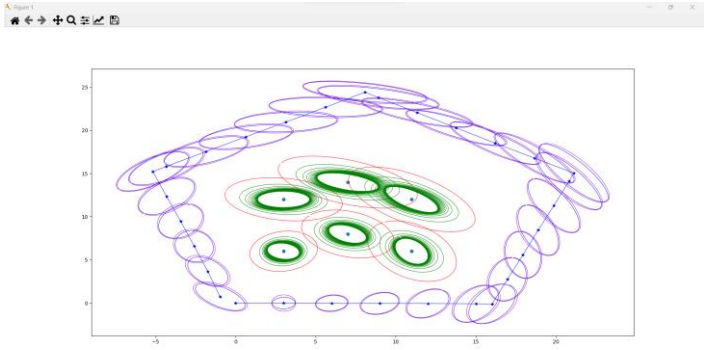
d. Sigma_x = Sigma_x / 10, Sigma_y = Sigma_y / 10, Sigma_alpha = Sigma_alpha / 10, rest unchanged



There is very little uncertainty in the robot pose prediction as we have reduced the control noise sigma values.

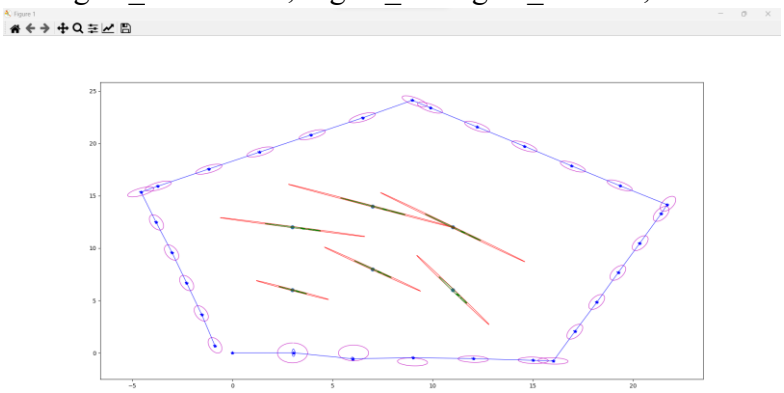
Name: Akshay Badagabettu
Andrew ID: abadagab

e. Sigma_beta = Sigma_beta * 10, Sigma_r = Sigma_r * 10, rest unchanged



Increasing the measurement noise sigma values causes the uncertainty of landmarks localization to increase. The initial covariance is huge as seen by the red ellipse, but as we keep providing new information, the uncertainty ellipse keeps reducing.

f. Sigma_beta = Sigma_beta / 1000, Sigma_r = Sigma_r / 1000, rest unchanged



When we decrease the measurement noise, the algorithm becomes very sure of the landmark positions from the beginning, and the covariance ellipse is not big enough. As EKF uses the known landmarks to localize both its pose and landmark location, this surety in the landmark position is detrimental to the model. The predicted robot pose deviates from the actual trajectory. Also, the landmarks are not localized correctly in the end, as can be evident from the high Mahalanobis distances, as shown in the picture below.

```
The Euclidean distance for landmark 1 is:  0.27732574277426675
The Mahalanobis distance for landmark 1 is:  10.365227175459472
The Euclidean distance for landmark 2 is:  0.5106187641428336
The Mahalanobis distance for landmark 2 is:  26.612092248373024
The Euclidean distance for landmark 3 is:  0.4313512459351797
The Mahalanobis distance for landmark 3 is:  14.515385393123427
The Euclidean distance for landmark 4 is:  0.6380195322517309
The Mahalanobis distance for landmark 4 is:  23.027342787254092
The Euclidean distance for landmark 5 is:  0.5056788457449197
The Mahalanobis distance for landmark 5 is:  29.848260411512936
The Euclidean distance for landmark 6 is:  0.6626209381560783
The Mahalanobis distance for landmark 6 is:  23.0818790476987
```

It should also be noted that this is a good example of why we need to use Mahalanobis distance instead of Euclidean distance. Even though the ground truth landmark position was way outside the predicted covariance ellipse, the Euclidean

distance is less. This may lead us to think that our model works very well, but the high Mahalanobis distances very accurately show that the model is not good. This is why Mahalanobis distance should always be used when measuring the distance between a point and a distribution.

3. Some changes that can be done to achieve constant computation time in each cycle or speed up the process are:
    - Parallelize the code so that it runs faster.
    - Implement landmark management techniques to limit the number of landmarks. As discussed in [2], M/N logic can be used to confirm a new landmark. This means that if during M consecutive time steps, the cluster is observed for at least N times, then it is considered as a landmark. This, of course, will not achieve constant computation time in each cycle, but the process might get sped up.
    - To achieve constant computation time, we have to have the same number of landmarks in each cycle. So, we need to adopt a scheme of dropping some landmarks in each cycle. This dropping can be random, or only the nearest N landmarks to the robot can be considered. This will most probably not yield good results.
    - Some matrices in the EKF algorithm have a lot of zero elements. For example, in the ‘init_landmarks’ covariance matrix, ‘control_cov’ matrix, and ‘Gt’ matrix in the ‘predict’ step. Instead of storing the entire matrix, sparse matrix representations can be used, which saves a lot of computational time. The effect will not be evident for 15x15 matrices, but when the dimensions of the matrices are in thousands, sparse matrices will save a considerable amount of computation time.
    - The update step of EKF has a lot of heavy matrix computations, which take a lot of time, especially the inverse term. We can try and approximate some values in the matrices, if after approximation, we have a lot of zero elements, we can use sparse matrix representations in this step too.

REFERENCES

[1] Thrun, S., 2002. Probabilistic robotics. *Communications of the ACM*, *45*(3), pp.52-57.

[2] Sun, S., Jelfs, B., Ghorbani, K., Matthews, G. and Gilliam, C., 2022, November. Landmark management in the application of radar slam. In *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)* (pp. 903-910). IEEE.