ROB312 - TP4 : SLAM using Extended Kalman Filter

Dajing GU

January 2020

1 Introduction

In this practical work, we study a Simultaneous Localization and Mapping (SLAM) method that builds a map of an unknown environment using an Extended Kalman Filter (EKF).

1.1 Principle of EKF SLAM

For each landmark perceived, if it is already in the map, a filter of Kalman will be used to estimate its pose and the robot pose, whose fomulas are presented below.

$$\begin{aligned} X_t^* &= f\left(X_{t-1}, u_t\right) \\ P_t^* &= A.\hat{P}_{t-1} \cdot A^T + B \cdot Q \cdot B^T \\ Y_t^* &= h\left(X_t^*\right) \\ K &= P_t^* H^T \cdot \left(H \cdot P_t^* \cdot H^T + P_Y\right)^{-1} \\ \hat{X}_t &= X_t^* + K\left(Y_t - Y_t^*\right) \\ \hat{P}_t &= P_t^* - KHP_t^* \end{aligned} \tag{1.1}$$

If it's not in the map, it will be added to the state vector. When necessary, X with absolute landmark pose and P with the right covariances will be extended by calculating J_r and J_y.

1.2 Introduction of TP

The provided code simulates a robot moving on a given trajectory in an environment made up of punctual landmarks. A simple extended Kalman filtering method using the perception of the direction and distance of these landmarks is implemented, whose result is illustrated in the figure 1.1.

In the left figure, the red curve is the position estimated by the EKF, the green one presents the odometry trajectory and the black one is the true trajectory. While in the right figure, the blue line is the error between the position estimated and the true position, with the red lines the covariance.

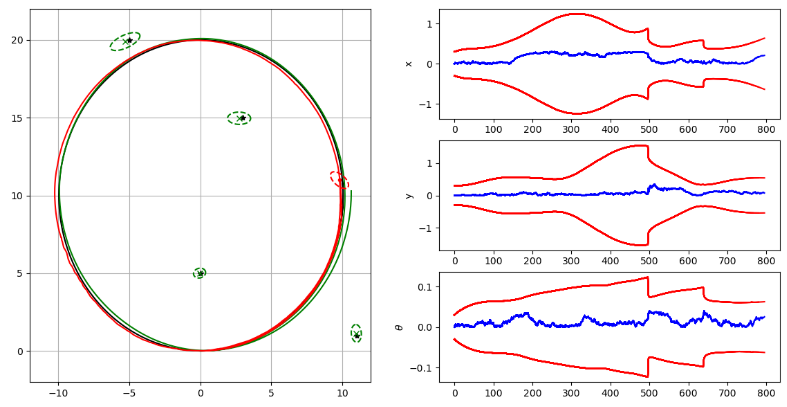


Figure 1.1: Illustration of the original condition

2 Influence of the environment

In this section, we will firstly analyze the influence of the environment on the performance of the algorithm. The number of positions are analyzed in question 1 and the Mahalanobis distance is used in question 2. By default, the data association is assumed to be known.

2.1 Question 1

The number and position of landmarks and the robot trajectory is firstly modified. The following three cases are studied. The parameter `yaw_rate` is related to the length of loop and `Landmarks` is the definition of landmarks. It should be noted that too close landmarks may cause some problems.

Case 1: A short loop and a dense map

In this case, a bigger `yaw_rate` $= 0.2$ is chosen to have a shorter loop. More landmarks are created to increase the density of the map.

A map of high quality is given in the figure 2.1. The red curve (true trajectory), and the black curve (estimated curve) overlap very well and the position of each landmark is well localized. However, it can be found that the landmarks near the start position is better localized than those far from the start position since the corresponding uncertainty is smaller. This is also shown in the right figure, where the error is accumulated during the loop and corrected at the start point, so as the the uncertainty.

In a short loop and a dense map, the robot is able to keep many landmarks inside it's perception radius during the movement. By doing data association and matrix updating, a good map can then be obtained.

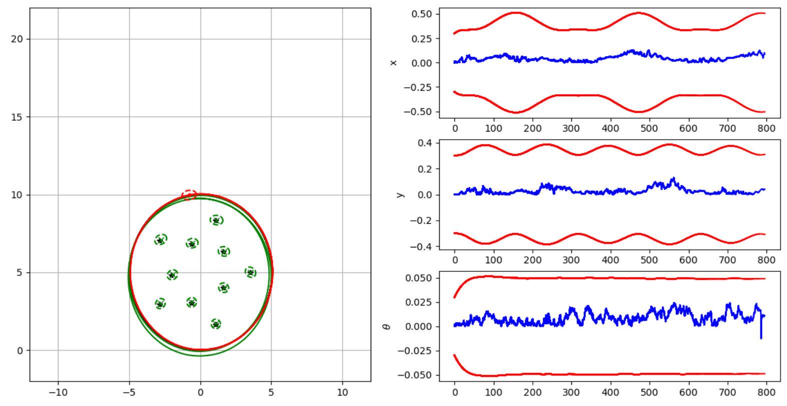


Figure 2.1: Illustration of case 1

Case 2: A long loop and a dense map

In this case, the value of `yaw_rate` is set as 0.1 to have a longer loop and the landmarks are added all along the loop.

The quality of the map is also good with the black curve (estimated position) and the red curve (true position) overlapping very well. The position of each landmark is also well localized. It can still be found that the landmarks near the start position is better localized than those far from the start position. In the right figure, the error and the uncertainty is also accumulated during the loop and corrected at the start point.

The big range of perception 10 allows the robot to keep many landmarks inside its perception radius during the movement, which give us finally a good map.

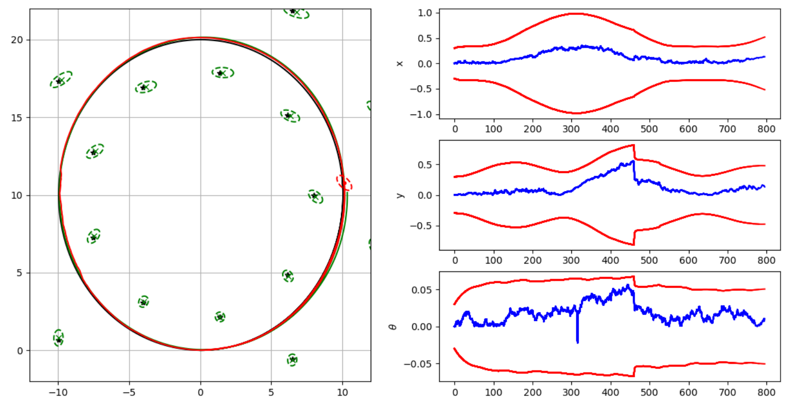


Figure 2.2: Illustration of case 2

Case3 : A long loop and a sparse map

In this case, there are only few landmarks near the start position. The landmarks in the map are all localized since they are near the start position but the three curves no longer overlap. which is also shown by the big error and big uncertainty in the right figure.

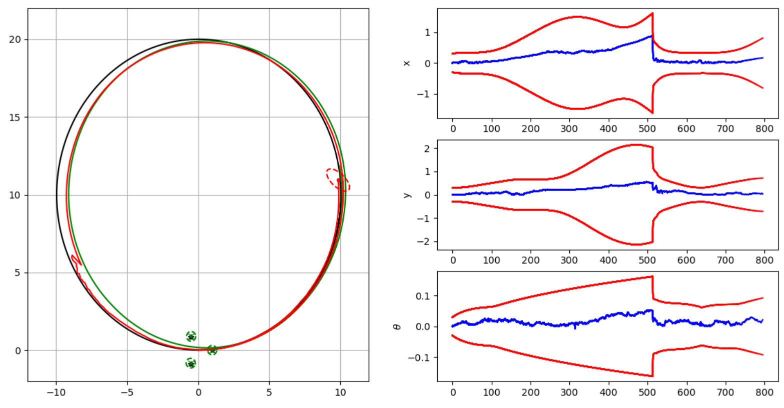


Figure 2.3: Illustration of case 3

The lack of landmarks makes the robot unable to correct and update its prediction, which causes this phenomenon.

2.2 Question 2

Then we study the case when the data association is performed using the Mahalanobis distance (`KNOWN_DATA_ASSOCIATION = 0`). The parameter `M_DIST_TH` shoud be modeified according to the environment.

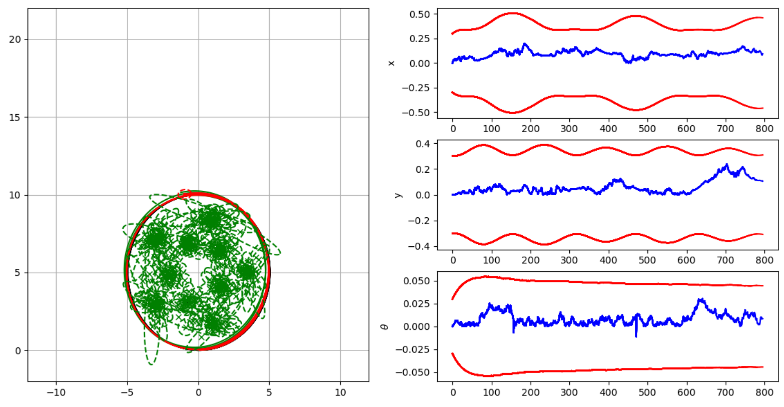


Figure 2.4: Case of `M_DIST_TH` = 1

`M_DIST_TH` is the threshold of Mahalanobis distance for data association. It can be seen in the figure 2.4 that when the threshold is too small, more points than reality will be misidentified as landmarks. However, when the threshold is too big, some landmarks will be ignored and there will be big errors between the real position and the estimated position, which is presented in the figure 2.5. The landmarks are not located either. Thus, an appropriate value of threshold should be chosen to ensure the normal functionality of the algorithm. Here the threshold is set 9.

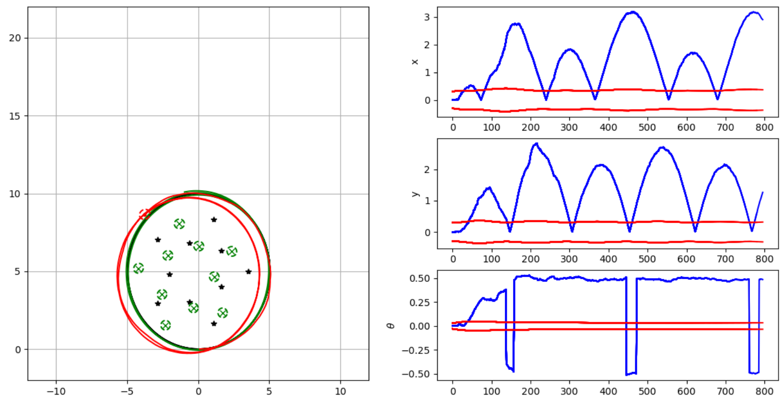


Figure 2.5: Case of `M_DIST_TH` = 100

When an appropriate threshold is chosen, the figures obtained are very similar to those obtained in Question 1. When there are enough landmarks inside the perception area of the robot, the correction and the update of the prediction can be made, which brings a good map, like the figure 2.6 and 2.7. However, when there aren't enough landmarks, the red curve (true position) and the black curve (estimated position) no longer overlap, which is also shown by the big error and uncertainty in the right figure of the figure 2.8.

Case 1 : A short loop and a dense map

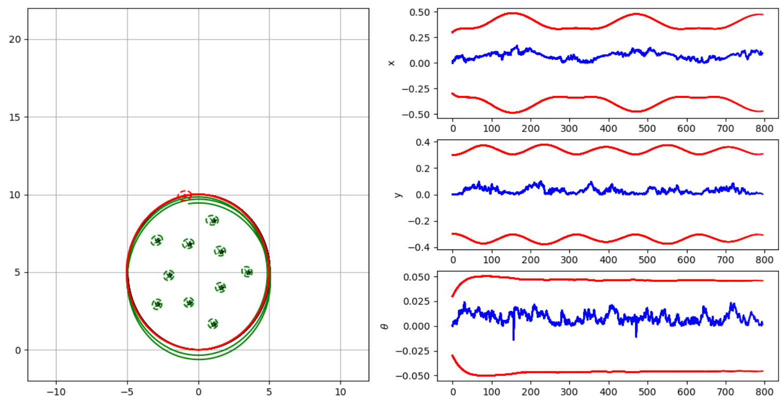


Figure 2.6: Illustration of case 1

Case 2 : A long loop and a dense map

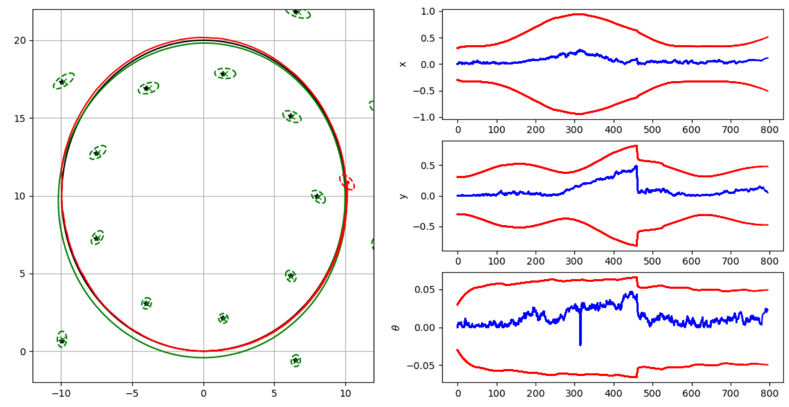


Figure 2.7: Illustration of case 2

Case3 : A long loop and a sparse map

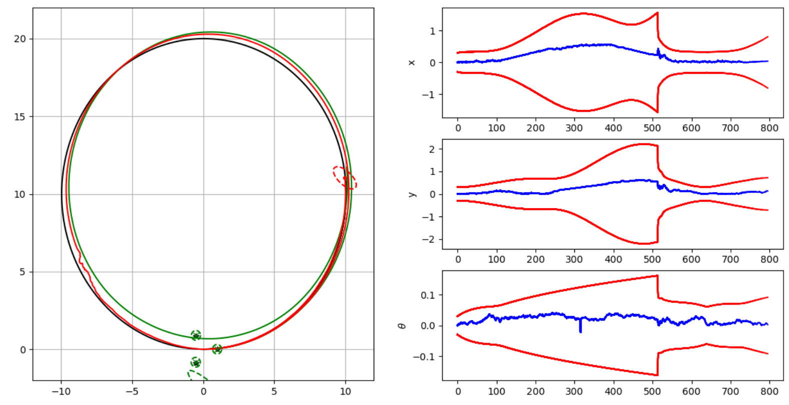


Figure 2.8: Illustration of case 3

3 Probabilistic models

In this question, we keep the configuration with unknown data association, which means `KNOWN_DATA_ASSOCIATION` $= 0$, and an environment with a large loop and a sparse map. According to the question 2, the threshold of Mahalanobis distance for data association is set 9.

3.1 Question 3

The estimated noise values `Q` and `Py` are changed in order to analysed the influence of these two parameters when they are $\star$ smaller, $\star$ equal or $\star$ larger than the values used for simulation (`Q_Sim` and `Py_Sim`).

Case 1 : `Q` = `Q_Sim`/10, `Py` = `Py_Sim`/10

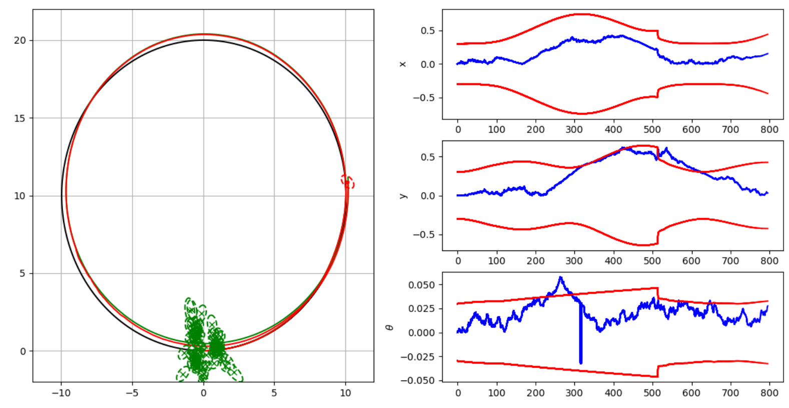


Figure 3.1: Illustration of case 1

It can be seen from the figure 3.1 that, when `Q` and `Py` is smaller than `Q_Sim` and `Py_Sim`, the positions of landmarks will be misidentified. The red curve (true position) and the black curve (estimated position) don't overlap and the error between the true position and the estimated position is really big.

Case 2 : `Q` = `Q_Sim`, `Py` = `Py_Sim`

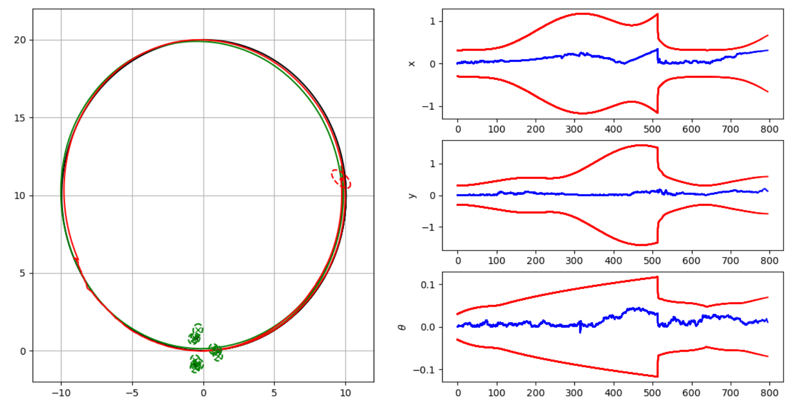


Figure 3.2: Illustration of case 2

According to the figure 3.2, When `Q` and `Py` is equal to `Q_Sim` and `Py_Sim`, we can find that the positions of landmarks are better identified than before but still not very good. The error is smaller in this case but the uncertainty increase. The red curve (true position) and the black curve (estimated position) overlap better.

Case 3 : `Q` **= 10** · `Q_Sim`**,** `Py` **= 10** · `Py_Sim`

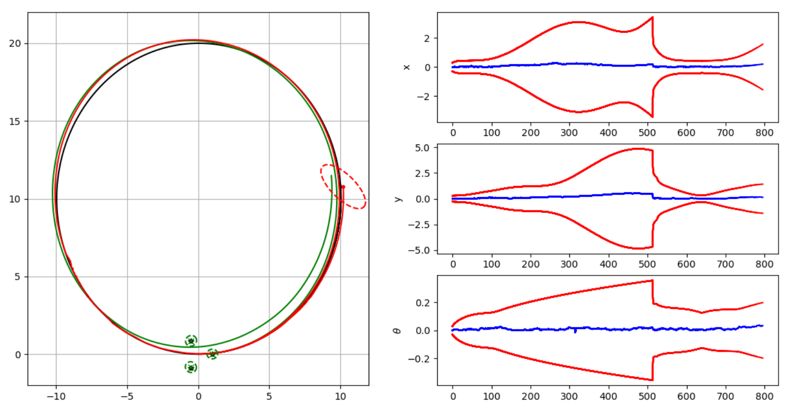


Figure 3.3: Illustration of case 3

According to the figure 3.3, when `Q` and `Py` is much bigger than `Q_Sim` and `Py_Sim`, we find that the positions of landmarks are well identified and the red curve (true position) and the black curve (estimated position) overlap better. The error is also smaller in this case but the uncertainty is really very big.

Best configuration : `Q` **= 2** · `Q_Sim`**,** `Py` **= 2** · `Py_Sim`

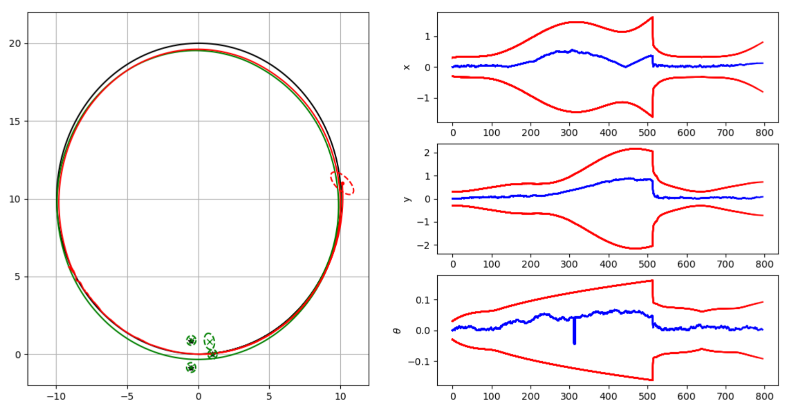


Figure 3.4: Case when `Q` = 2 `Q_Sim`, `Py` = 2 `Py_Sim`

From the figure 3.1, 3.2 and 3.3, we find that in comparison with `Q_Sim` and `Py_Sim`, the bigger the `Q` and `Py`, the better the landmarks are identified and the smaller the errors between the true position and the estimated position. However, the uncertainty increase. So the best configuration is that `Q` and `Py` is a little bigger than the `Q_Sim` and `Py_Sim`.

Just like the case presented in the figure 3.4, the landmarks are relatively well localized and the corresponding uncertainty is relatively small, the errors are also not too big.

4 Undelayed initialization

In this section, we will perform SLAM using only the direction of landmarks, not their distance. The method of undelayed initialisation will be used, which initializes several landmarks in a cone corresponding to the perceived direction and removes the useless landmarks afterwards. This method helps us to solve the problem of the difficult estimation of their position from a single perception. **It should be noted that the code in this section is written in the help of my classmate *Iad Abdul Raouf*.**

4.1 Question 4

4.1.1 Using only the direction of landmarks

The code of the filter is modified to use only the landmark direction in the Kalman correction. This time only the direction of landmarks are used so we modify firstly the noise of measurement `Py_sim` and `Py` from shape 2×2 to 1×1. All the indexes associated are also changed from 1 to 0 or from 2 to 1.

```
Py_sim = (1 * np.diag([np.deg2rad(5)])) ** 2  # modification
```

Then we change `y_i` in the function `observation` in order to eliminate the observation of distance.

```
yi = np.array([angle_n, i])  # modification
```

Then `y_p` in the function `calc_innovation` :

```
yp = np.array([[pi_2_pi(y_angle)]]) # modification
```

and also for `G` in the function `jacob_h`, we reserve only the second line of angle :

```
G = np.array([[delta[1, 0], -delta[0, 0], -q,  -delta[1, 0], ...
                                    delta[0, 0]]]) # modification
```

4.1.2 Implementation of a simple FIS-SLAM

A simple undelayed initialization is implemented for new landmarks by adding several landmarks along the perception direction with growing covariances.

It is the function `ekf_slam` that is modified in this subsection, only a part of the code modified is cited here.

According to the formula:

$$s_1 = (1 - \alpha)^{-1} \cdot s_{\min} \tag{4.1}$$

$$s_i = \beta^{i-1} s_1 \tag{4.2}$$

we can obtain that

$$s_i = \frac{\beta^{i-1} s_{min}}{1-\alpha} \tag{4.3}$$

Then we update the landmark hypothesis by stacking one by one iteratively according to the formula 4.4, the indexes associated are also changed.

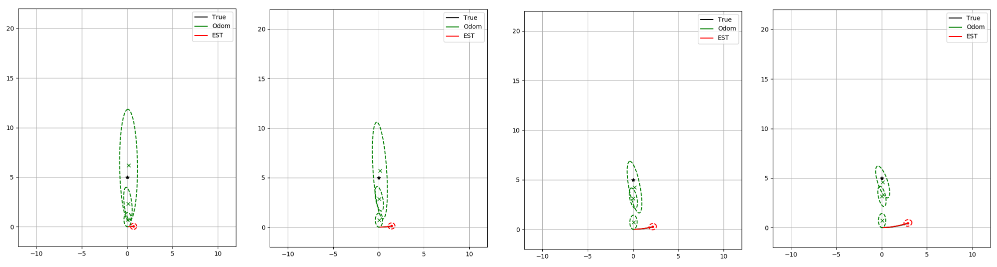
$$\hat{X}^{+} = \begin{bmatrix} \hat{X} \\ \hat{\mathbf{x}}_p^1 \\ \vdots \\ \hat{\mathbf{x}}_p^{N_g} \end{bmatrix} \qquad \mathbf{P}^{+} = \begin{bmatrix} \mathbf{P} & {\mathbf{P}_{pX}^1}^\top & \cdots & {\mathbf{P}_{pX}^{N_g}}^\top \\ \mathbf{P}_{pX}^1 & \mathbf{P}_{pp}^1 & & \\ \vdots & & \ddots & \\ \mathbf{P}_{pX}^{N_g} & & & \mathbf{P}_{pp}^{N_g} \end{bmatrix} \tag{4.4}$$

According to the formulas mentioned above, we can modify the code:

```python
for iRayInit in range(Ng):
    # create hypothetical new amer range and range noise
    d_hypo = (beta**iRayInit)*dmin/(1-alpha)
    y_hypo = np.array([d_hypo, y[iy, 0], y[iy, 1]])
    Py_hypo = np.diag([(alpha*d_hypo)**2, Py[0,0]])

    # Extend state and covariance matrix
    xEst = np.vstack((xEst, calc_landmark_position(xEst, y_hypo)))
    Jr, Jy = jacob_augment(xEst[0:3], y_hypo)
    bottomPart = np.hstack((Jr @ PEst[0:3, 0:3], Jr @ PEst[0:3, 3:]))
    rightPart = bottomPart.T
    PEst = np.vstack((np.hstack((PEst, rightPart)), ...
                                    np.hstack((bottomPart, Jr ...
                                    @ PEst[0:3, 0:3] @ Jr.T + ...
                                    Jy @ Py_hypo @ Jy.T))))
```

4.1.3 Results



(a) (b) (c) (d)

Figure 4.1: Illustration of the undelayed initialization from showing the initialization of three new landmarks in the map when a new landmark is detected in the environment

As is shown the figure 4.1, we have finished the test of the algorithm of bearing only SLAM. The procedure of the appearance and the update of the of the landmarks detected is clearly shown.