EKF-SLAM with Traffic sign images

Alaaeddine El Masri El Chaarani, Jomana Ashraf, and Rahaf Abu Hara

***Abstract*— The aim of this report is to perform the EKF-SLAM for localizing the robot and mapping the environment by using the traffic signs. Those traffic signs are detected by applying a recognition by the SSD Architecture and then sends the data to the localization model in order to perform the SLAM. The methodology used and how to deal with the perception data and prepare it to the localization is explained in this paper. Also, the problem faced and the solutions that are applied to overcome this problems are discussed with results of those solutions.**

I. INTRODUCTION

SLAM is the abbreviation for Simultaneous Localization and Mapping which is a method used in all the autonomous vehicles for building a map and locating the vehicle at the same time in unknown environment. It is used in many applications as controlling the navigation of a fleet of mobile robots to organize shelves in a warehouse, parking an autonomous car in an empty space, or delivering a package by piloting a drone in an unfamiliar environment. The SLAM consists of three main parts which are [1]:

- The robot moves to a new position and predict its location according to the Odometry sensor which gives the linear and angular velocity of the vehicle. Due to the noise came from this sensor, this motion increases the uncertainty on the robot's localization.
- The robot discovers new landmarks in the environment; the location of the landmarks will be uncertain as the robot is uncertain. Then a mathematical model to determine the position of the landmarks from the data obtained by the sensors is computed by a model called "Inverse observation model"
- The robot observes landmarks that already seen before to correct both the robot localization and the localization of all landmarks which is called "observation model" which is used in the correction or the update step to perform the localization

II. METHODOLOGY

A. Main Architecture

The architecture of the algorithm depends mainly on two topics which are the "/Odom" topic and the "/perception topic". When the Odom data is received,the apply prediction function from the EKF-SLAM is executed to compute the predicted position of the robot and its covariance. After that, if the perception data received, the data association is called to make sure that our perception model is working fine, So if the class id retrieved from the perception is same as the one got from the data association; so, it will check if this class id in the dictionary; so if yes, the state vector will be updated and the covariance matrix. If not, this new landmark will be added to the state vector. But, when our perception model predict the traffic sign wrongly, the right sign will be got from the data association and then the same process of updating or adding new landmark will be executed.

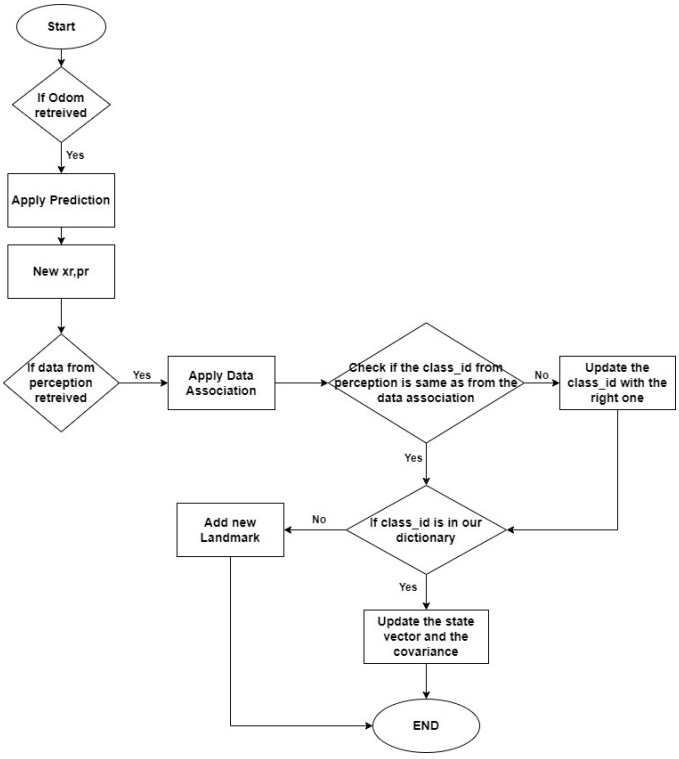

Fig. 1. shows the architecture of the algorithm used.

B. EKF-SLAM

1) Initialization Step:
At the initial time, the initial state only consists of the robot state, and there is still no landmark registered to the map. However, to avoid expensive computations the state is initialized as a zero's vector of a size of (mx1) where m is equal to equation1.

$$m = 3 + maximumlandmarks * landmarksize \quad (1)$$

to include the robot state and n number of landmarks as shown in equation 2.

$$X = \begin{bmatrix} X_R \\ f_1 \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} x_R \\ y_R \\ \gamma_R \\ x_{f1} \\ y_{f1} \\ \vdots \\ x_{fn} \\ y_{fn} \end{bmatrix} \tag{2}$$

Moreover, it is assumed that the initial state of the robot is known exactly, and there is no uncertainty. While the uncertainty of the landmarks is infinity since there is no knowledge about the position of the landmarks yet. So, a reserved place in the memory is initialized to be (mxm) as seen in equation 1 and the covariance of the robot will be initialized as zeros which are the first (3x3) matrix. Due to new initialization of landmark process, the covariance matrix will expand in size every time the robot detects new observed landmarks as in equation 3.

$$P = \begin{bmatrix} P_{RR} & P_{RM} \\ P_{MR} & P_{MM} \end{bmatrix} = \begin{bmatrix} P_{RR} & P_{Rf1} & \cdots & P_{Rfn} \\ P_{f1R} & P_{f1f1} & \cdots & P_{f1fn} \\ \vdots & \vdots & \ddots & \vdots \\ P_{fnR} & P_{fnf1} & \cdots & P_{fnfn} \end{bmatrix} \tag{3}$$

Since updating the whole covariance matrix and state vector can be computationally expensive, NumPy views are used to update only the affected parts of the state vector and covariance matrix.

2) Prediction Step:

First the prediction model is computed where the new position of the robot is required in the world frame which is the blue line in fig.2 so compounding is used to compute the different frames as shown in equation 4

$$X^N_{R_k} = X^N_{R_{K-1}} \oplus (u^{R_{K-1}}_{R_k} + w^{R_{K-1}}_{R_k}) \tag{4}$$

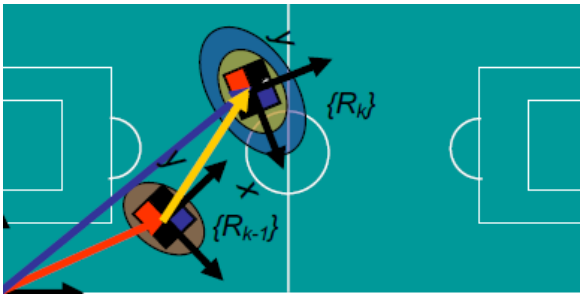

Fig. 2. shows the different frames for the prediction step

where the $u^{R_{K-1}}_{R_k}$ is the odometry readings and $w^{R_{K-1}}_{R_k}$ is the noise getting from the sensor. The equation computed from the compounding will be equal to:

$$X^N_{R_k} = \begin{bmatrix} cos(\gamma^N_{R_{K-1}})(x^{R_{K-1}}_{R_k} + w_x) - sin(\gamma^N_{R_{K-1}})(y^{R_{K-1}}_{R_k} + w_y) + x^N_{R_{K-1}} \\ sin(\gamma^N_{R_{K-1}})(x^{R_{K-1}}_{R_k} + w_x) + cos(\gamma^N_{R_{K-1}})(y^{R_{K-1}}_{R_k} + w_y) + y^N_{R_{K-1}} \\ \gamma^N_{R_{K-1}} + \gamma^{R_{K-1}}_{R_k} + w_\gamma \end{bmatrix} \tag{5}$$

Then the Jacobian of the function with respect to the state $x_{k_{-1}}$ and the noise are represented in equations 6,7 respectively

$$A = \begin{bmatrix} 1 & 0 & -sin(\gamma^N_{R_{K-1}})(x^{R_{K-1}}_{R_k}) - cos(\gamma^N_{R_{K-1}})(y^{R_{K-1}}_{R_k}) \\ 0 & 1 & cos(\gamma^N_{R_{K-1}})(x^{R_{K-1}}_{R_k}) - sin(\gamma^N_{R_{K-1}})(y^{R_{K-1}}_{R_k}) \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$W = \begin{bmatrix} cos(\gamma^N_{R_{K-1}}) & -sin(\gamma^N_{R_{K-1}}) & 0 \\ sin(\gamma^N_{R_{K-1}}) & cos(\gamma^N_{R_{K-1}}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

When the robot is moving, only the state of the robot, will be affected from this movement. The new estimated pose of the robot is computed as in equation 5. Since the robot movement only affects the states of the robot, the covariance matrix related to the robot states in addition to the cross-correlation between the robot and the map will be updated as shown in fig.3 [2].

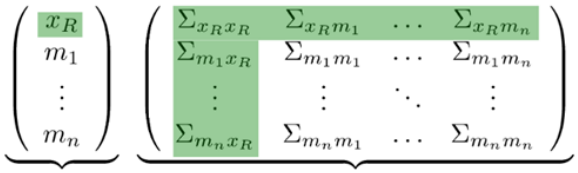

Fig. 3. shows the updated part in the state vector and covariance matrix in the prediction step

To update the covariance matrix as shown in fig.3, the following equations are used:

$$\begin{aligned} P_{RR} &= A * P_{RR} * A^T + W * Q @ W^T \\ P_{RM} &= A * P_{RM} \end{aligned} \tag{8}$$

Where Q is the covariance of the noise and in our case, it is equal to constant number which is:

$$Q = \begin{bmatrix} 0.025^2 & 0 & 0 \\ 0 & 0.025^2 & 0 \\ 0 & 0 & 0.025^2 \end{bmatrix} \tag{9}$$

3) Update based on Landmarks Observation:

When the robot is moving, the landmarks are observed around it. When known landmark are detected, the estimated state vector of the robot is then updated based on the difference in actual measurement which is the "y" in equation 15 is getting from the camera frame which are ρ_p and θ, and expected measurement of the correspond landmark.Hence, it is required to get the expected measurement from the camera to the landmark as shown in fig.4 by using the following equations where equation10 is the estimated measurement in the world frame but the actual measurement is in the

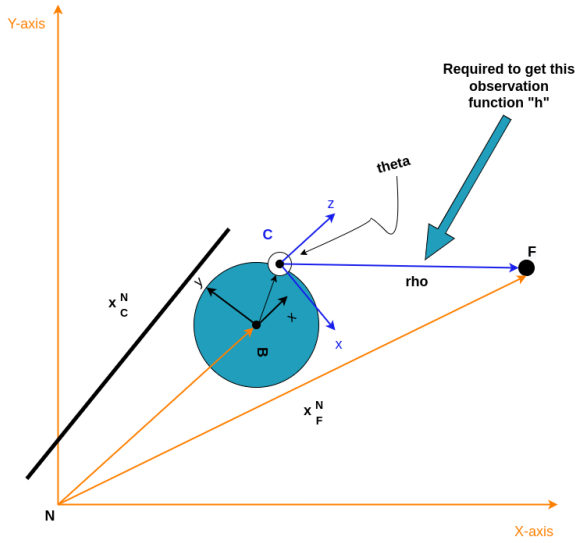

Fig. 4. shows the transformations between the frames to get the expected measurement

camera frame so, the estimated measurement is converted to the camera frame by using equation11

$$h_F^N = X_F^N - T_B^N X_C^B \tag{10}$$

$$h_F^C = R_N^C h_F^N \tag{11}$$

Therefore, the estimated measurement is equal to:

$$h_F^C = \begin{bmatrix} x_F^N sin(\gamma_B^N) - y_F^N cos(\gamma_B^N) - x_B^N sin(\gamma_B^N) + y_B^N cos(\gamma_B^N) + y_C^B \\ x_F^N cos(\gamma_B^N) + y_F^N sin(\gamma_B^N) - x_B^N cos(\gamma_B^N) - y_B^N sin(\gamma_B^N) + x_C^B \end{bmatrix} \tag{12}$$

Then the Jacobian of expected measurement with respect to the robot state is computed which is a matrix of 2x3 by using equation 13

$$Jhxr = \begin{bmatrix} -sin(\gamma_B^N) & cos(\gamma_B^N) & x_F^N cos(\gamma_B^N) + y_F^N sin(\gamma_B^N) - x_B^N cos(\gamma_B^N) - y_B^N sin(\gamma_B^N) \\ -cos(\gamma_B^N) & -sin(\gamma_B^N) & -x_F^N sin(\gamma_B^N) + y_F^N cos(\gamma_B^N) + x_B^N sin(\gamma_B^N) - y_B^N cos(\gamma_B^N) \end{bmatrix} \tag{13}$$

Then the Jacobian of the expected measurement with respect to the landmark is computed which is a matrix of 2x2 by using equation 14

$$Jhxf = \begin{bmatrix} sin(\gamma_B^N) & -cos(\gamma_B^N) \\ cos(\gamma_B^N) & sin(\gamma_B^N) \end{bmatrix} \tag{14}$$

Accordingly, all the state vector and the covariance matrix are updated as shown in fig.5 [2].

$$\underbrace{\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{} \quad \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \dots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \dots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R} & \Sigma_{m_n m_1} & \dots & \Sigma_{m_n m_n} \end{pmatrix}}_{}$$

Fig. 5. shows the update of all the state vector and the covariance matrix

To make this update, first the difference between the measurements are computed by using equation 15 then the kalman gain is obtained by equation 17

$$z = y - h_F^C \tag{15}$$

$$Z = \begin{bmatrix} Jhxr & Jhxf \end{bmatrix} P \begin{bmatrix} Jhxr & Jhxf \end{bmatrix}^T + R \tag{16}$$

$$K = P \begin{bmatrix} Jhxr & Jhxf \end{bmatrix}^T Z^{-1} \tag{17}$$

Then the state vector and covariance matrix are updated by using equations 18,19 respectively.

$$X = X + Kz \tag{18}$$

$$P = P + KZK^T \tag{19}$$

4) Landmark Initialization:

When the robot observes a new landmark, the state of this landmark with respect to the world frame is estimated. In the inversion observation, it is required to get the location of the landmark which consists of x_f, y_f from the measurement which contains the distance ρ and the angle θ and the state of the robot that contains $x_b, y_b, and \gamma$ as shown in fig.6.

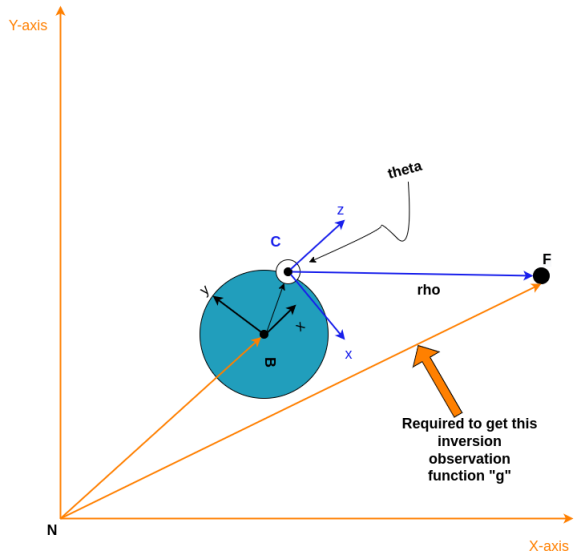

Fig. 6. shows the transformations to get the inversion observation with respect to the world frame

So, the inversion observation is computed using the following equation where z_F^C is the measurement retrieved from the camera

$$g_F^N = T_C^N z_F^C$$

$$= \begin{bmatrix} x_b + x_c cos(\gamma_B^N) + \rho_p cos(\gamma_B^N) cos(\theta) + \rho_p sin(\gamma_B^N) sin(\theta) \\ y_b + x_c sin(\gamma_B^N) - \rho_p cos(\gamma_B^N) sin(\theta) + \rho_p sin(\gamma_B^N) cos(\theta) \end{bmatrix} \tag{20}$$

Then the Jacobian of the inversion observation with respect to the robot state which is a matrix of 2x3 is computed by using equation 21.

$$Jgxr = \begin{bmatrix} 1 & 0 & -x_c sin(\gamma_B^N) - \rho_p sin(\gamma_B^N) cos(\theta) + \rho_p cos(\gamma_B^N) sin(\theta) \\ 0 & 1 & x_c cos(\gamma_B^N) + \rho_p sin(\gamma_B^N) sin(\theta) + \rho_p cos(\gamma_B^N) cos(\theta) \end{bmatrix} \tag{21}$$

After that the Jacobian of the inversion observation with respect to the measurement which consists of 2x2 by using equation 22.

$$Jgxf = \begin{bmatrix} cos(\gamma_B^N)cos(\theta)+sin(\gamma_B^N)sin(\theta) & \rho_p sin(\gamma_B^N)cos(\theta)-\rho_p cos(\gamma_B^N)sin(\theta) \\ sin(\gamma_B^N)cos(\theta)-cos(\gamma_B^N)sin(\theta) & -\rho_p cos(\gamma_B^N)cos(\theta)-\rho_p sin(\gamma_B^N)sin(\theta) \end{bmatrix} \quad (22)$$

Consequently as shown in fig.7, the new landmark is added to the state vector and the covariance and its cross-correlation are also added.

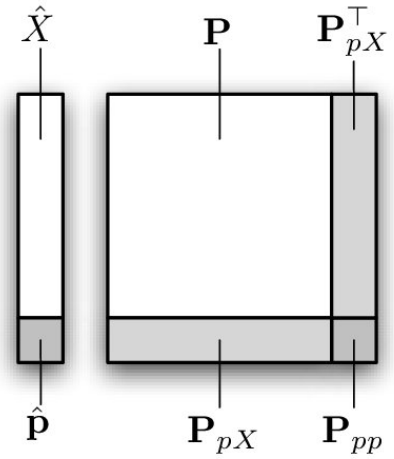

Fig. 7. shows the new covariance and state vector of the added new landmark

So to add the new landmark in the state vector which is in x,y with respect to the world frame equation 20 is used. Then to add the new covariance, the following equations are used:

$$P_{pp} = JgxrP_{RR}Jgxr^T + JgxfRJgxf^T \quad (23)$$

$$P_{pX} = Jgxr\begin{bmatrix} P_{RR} & P_{RM} \end{bmatrix} \quad (24)$$

C. Perception Preparation Data

In order to perform the update or adding the new landmark, features are required to be identified so, this is the role of our perception model. It detects the traffic signs by sending the class id of this predicted sign with the center of the bounding box in pixels. Therefore, the localization model took this data and preform some conversions to fit this data with our localization model.

As mentioned before that our model take from the camera the distance ρ and the angle θ hence the distance is the depth from the camera where the pixels of the center of the bounding box is used in the depth image to get the depth of this predicted traffic sign. Then as shown in fig. 8, the center of the bounding box is retrieved from the pixel frame of the camera so, to convert it to the center of the frame; equation 25 is used where the u_0v_0 are the principle points retrieved from the intrinsic of the camera.

$$X = i_u - u_0, Y = i_v - v_0 \quad (25)$$

In order to get the theta, equation 26 is used where the focal length f got from the intrinsic parameters of the camera.

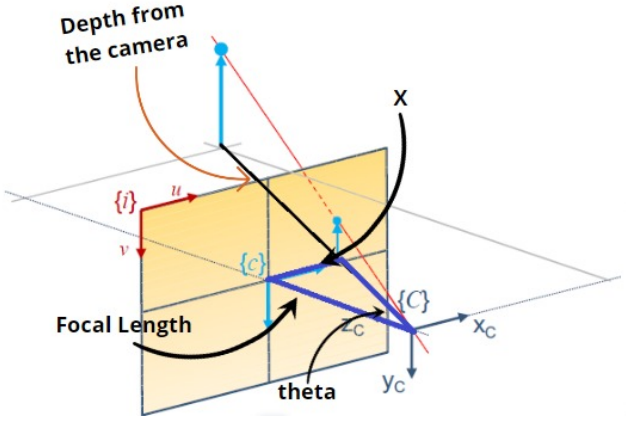

Fig. 8. shows the conversions made to get the ρ and θ

Eventually, the depth and the theta are published to the specified topic to send this data to the localization model to perform the SLAM algorithm.

$$\theta = tan^{-1}(\frac{X}{f}) \quad (26)$$

III. Results and Discussion

A. Results of the traffic signs with EKF-SLAM

First, the prediction model of the EKF-SLAM is tested in the simulation environment. As shown in fig. 9, the uncertainty of the robot was small and then after the robot moves and the Odom receive data; the uncertainty begin to increase as shown in fig. 10 and the predicted position of the robot is being computed.

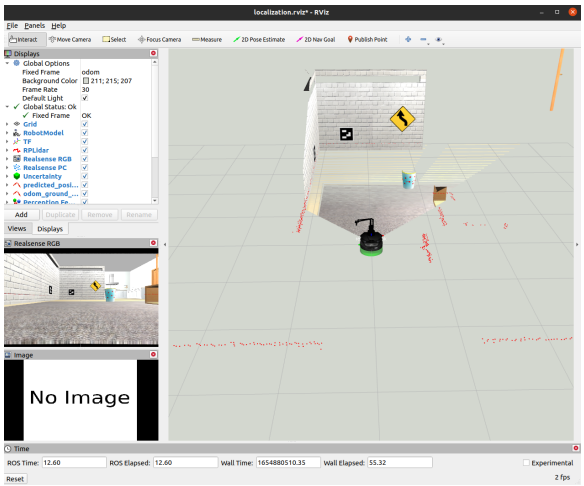

Fig. 9. shows the uncertainty of the robot at the beginning

B. Problems Faced

As the prediction of the perception model in detecting the traffic signs is not working well because our data set is taken from a landscape with large resolution; But our testing environment is an indoor environment with small resolution. Thus, the localization model was not able to be

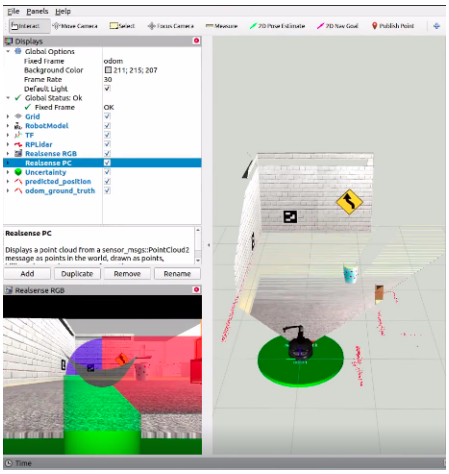

Fig. 10. shows the increasing in the uncertainty of the robot while moving

tested with the update and landmark initialization steps; so some solutions are performed to overcome this problem and test our model.

C. Solution Proposed

1) Data Association: In order to solve the mentioned problem, data association is performed by adding the locations of the traffic signs in our environment in a dictionary and then after applying the perception the locations retrieved is compared with the output of the data association and then perform the same procedure of the EKF-SLAM and the algorithm shown in fig. 11 for the data association is used.

Algorithm 23: Individual Compatibility Nearest Neighbour (ICNN)

Function ICNN$(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{P}}_k, \boldsymbol{z}_{f_1 \ldots m}, \boldsymbol{R}_{f_1 \ldots m})$
  for $i = 1$ *to* m **do** // Observation $\boldsymbol{z}_{f_i}$
    $D^2_{ij} = Mahalanobis(\boldsymbol{z}_{f_i} - \hat{\boldsymbol{h}}_{F_j}, \boldsymbol{P}_{F_j} + \boldsymbol{R}_{f_i})$;
    $nearest = 0$; // 0 ≡ non compatible with any feature
    $\mathscr{D}^2_{min} = \infty$;
    for $j = 2$ *to* n **do** // Feature $\boldsymbol{x}_{F_j}$
      $\mathscr{D}^2_{ij} = Mahalanobis(\boldsymbol{z}_{f_i} - \hat{\boldsymbol{h}}_{F_j}, \boldsymbol{P}_{F_j} + \boldsymbol{R}_{f_i})$;
      if $IndividuallyCompatible(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{P}}_k, \boldsymbol{z}_{f_i}, \boldsymbol{R}_{f_i}, \boldsymbol{x}_{F_j}, \alpha)$ **and** $\mathscr{D}^2_{ij} < \mathscr{D}^2_{min}$ **then**
        $nearest = j$;
        $\mathscr{D}^2_{min} = \mathscr{D}^2_{ij}$;
    $\mathscr{H}_p[i] = nearest$;
  return $\mathscr{H}_p$

Fig. 11. shows the Individual Compatibility Nearest Neighbor algorithm to perform data association

2) Results of the ArUco with EKF-SLAM: However the data association is performed, the perception model still does not recognize the signs while moving in the environment due to the difference in resolution. Therefore, a test for our localization model with ArUco markers is used. The perception model is tested by using the ArUco features, so as shown in fig. 12,the position of the ArUco markers are visualized in Rviz after making the localization conversions required as discussed in section II-C. Then the prediction model is tested again and the same results as before appeared that the uncertainty of the robot increases while moving as shown in fig. 13.

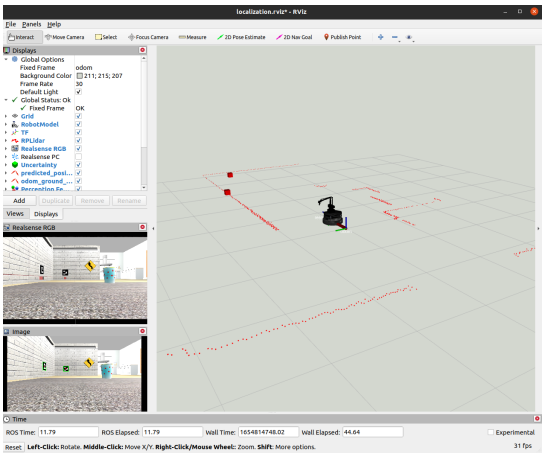

Fig. 12. shows the location of the detected ArUco markers in red cubes after making the conversions in the perception node

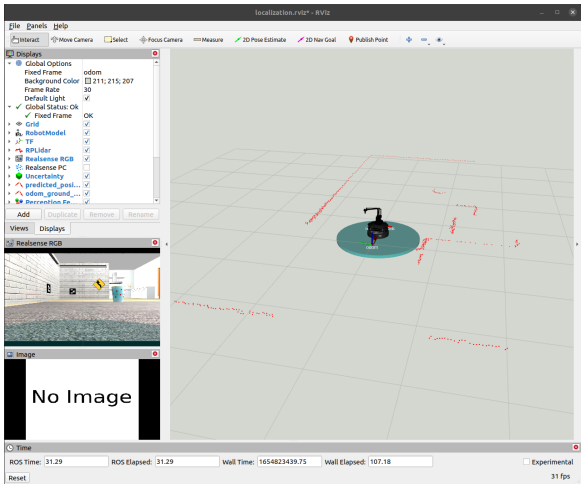

Fig. 13. shows the increasing in the uncertainty of the robot while moving in the ArUco model

Subsequently, the perception and the EKF-SLAM nodes are running together where when the ArUco detected by the perception node and publishing the ρ and θ; the other node will subscribe and will publish the location of the detected feature in green after applying the landmark initialization as shown in fig. 14.
When the robot observes the same landmark again and check the dictionary if containing this landmark; so, the update step is executed where the uncertainty of the robot decreased as shown in fig. 15

IV. CONCLUSIONS AND FUTURE WORKS

To sum up, the model is tested with different features but with the same model explained in section II-B. Our future work is to test our model in a street that having traffic signs to be as same as the trained data provided by the perception model in order to achieve the success of the localization model.

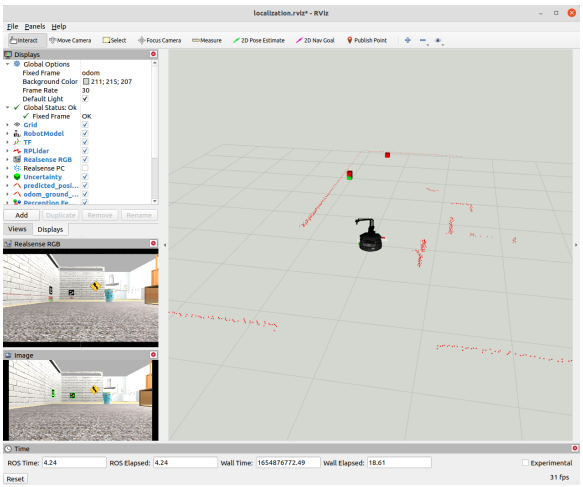

Fig. 14. shows the location of the ArUco markers in green cubes after applying the landmark initialization

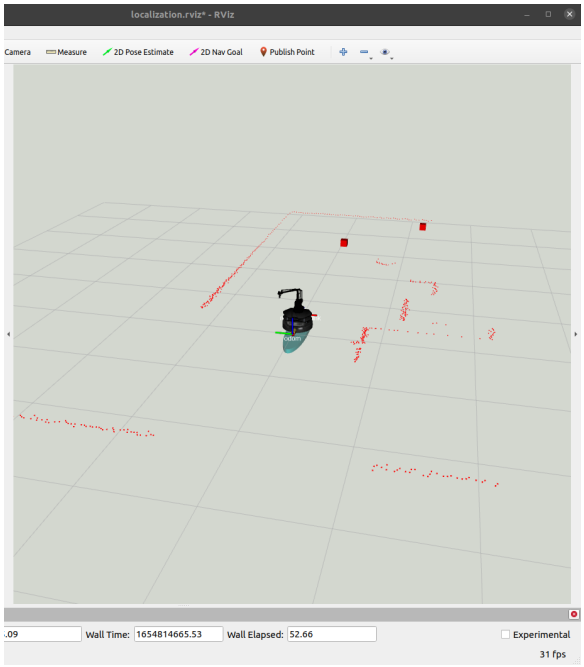

Fig. 15. shows the uncertainty of the robot after applying the update

REFERENCES

[1] J. Sol‘a, Iri.upc.edu, 2014.Available: https://www.iri.upc.edu/people/jsola/JoanSola/objectes/cursSLAM/-SLAM2D/SLAM/20course.pdf.

[2] Cyril Roussillon, Aur´elien Gonzalez, Joan Sol‘a, Jean Marie Codol, Nicolas Mansard, Simon Lacroix, and Michel Devy. RT-SLAM: a generic and real-time visual SLAM implementation. In Int. Conf. on Computer Vision Systems (ICVS), Sophia Antipolis, France, Sept. 2011