EKF-BASED LOCALIZATION WITH LRF

Gonçalo Castilho (75305), Tiago Gomes (76079), Pedro Sousa (79205), and Markus Kühlen (85746)

***Abstract*—This project report deals with the implementation of an extended Kalman filter (EKF) on a mobile robot, which is equipped with a laser rangefinder (LRF). The goal of the project is to estimate the two dimensional pose of the mobile robot in real time. The available odometry information of the robot is used for the state prediction. For the observation model a map of the environment is pre-acquired and used. The laser rangefinder gives the observation information by scanning the robots environment in real time. An ICP algorithm then compares the two point clouds from the predicted and the real observation.**

***Index Terms*—Extended Kalman filter, Localization, Robotics, Laser Rangefinder, Point Clouds.**

I. Introduction

MOBILE autonomous systems are fundamentally dependent on localization. Their motion and task planing require knowledge about the current robot state. For mobile robots the current posture, which includes position and orientation, is an important part of their state.

For this reason the report for the group project of the experimental part within the "Autonomous Systems" class at Instituto Superior Técnico in 2016/2017 deals with EKF-based localization with a LRF. The objective for the students is to prove their theoretical knowledge on mobile robotics localization in a practical scenario and to gain first experiences with implementing in a robot operating system (ROS) environment.

The used hardware in this project consists of a Pioneer 3DX mobile robot, a Hokuyo URG-04LX-UG01 laser rangefinder and laptop running Ubuntu operating system. The Pioneer 3DX comes with implemented motion sensors that provide odometry information. The Pioneer's integrated sonar sensors are not used.

The available odometry information can be used straight away for localization. However, relying only on odometry is inaccurate, since the errors arising from the uncertainties of the odometry model and the measurement noise of the odometric sensor are accumulating over time. To improve its localization the robot can use available information from other sensors, such as a sonar, a camera or a laser rangefinder, each having different advantages and disadvantages. The task for this project is to use a laser rangefinder, which is more accurate in comparison to a sonar, but is not able to measure transparent objects. The additionally gained information has to be merged with the odometry information. Therefore, different kind of algorithms, so called filters, can be used. [1]

The original Kalman filter is the optimal estimate algorithm for linear system models with additive independent white noise in the prediction and measurement systems. To be able to apply the Kalman filter based filtering method to non-linear systems the extended Kalman filter uses linearization around a working point. As long as the system model is well known and accurate, the EKF is the most widely used estimation algorithm. Otherwise Monte Carlo methods, especially particle filters, will lead to better results, despite being computationally more expensive. [2]

As the considered robot system is a real world non-linear system the used filter for localization has to be robust to the influence of noise and non-linearities. The movement of a wheeled mobile robot in a two dimensional environment can be described by an accurate system model. Therefore, the EKF can be used for computationally efficient localization. [3]

This paper is organized as follows. In section II the used methods are introduced. Afterwards the implementation of the EKF is described in section III. The results of the algorithm running on the real Pioneer 3DX robot in a test environment are then discussed in section IV. The paper then is concluded in section V.

II. Methods

A. Robot Operating System

The Robot Operating System is a set of software libraries and tools that helps building robot applications. It is a modularized, portable and standard system which allows it to be developed by different system designers, and has a wide range of uses, from quadcopters to industrial-type robotic manipulators.

ROS is mainly composed by 4 elements:

- Roscore – main program, which acts primarily as a name server
- Node – process that uses ROS framework and performs a specific task
- Topic – mechanism to send messages from a node to one or more nodes. Follows publisher-subscriber design pattern
- Service – mechanism for a node to send a request to another node and receive a response in return.

In order to develop the project, several ROS packages were used as a foundation. A Package is a self-contained directory containing source code, CMake makefiles, builds and others. Below there is a list of the main packages used in this project.

- rosaria – ROS interface for the pioneer 3DX robot. It allows issuing commands to the robot wheel motors as well as retrieving information on the odometric sensors.
- teleop_twist_keyboard – Provides teleoperation using a keyboard.
- slam_gmapping – Provides laser-based SLAM (Simultaneous Localization and Mapping), elaborating a 2-D occupancy grid map from laser and pose data collected by the LRF and the Pioneer robot.
- map_server – Provides a map_server ROS Node, which offers map data as a ROS Service. It also provides the

map_saver command-line utility, which allows dynamically generated maps to be saved to file.
- rviz – visualizing tool for displaying sensor data and state information from ROS.
- tf – Used to keep track of the robots frame in relation to the static world reference
- urg_node – Used to make a connection between the Hokuyo LRF and the ROS core. It provides topics to read the information acquired by the laser beams.

B. Mapping

The LRF was fitted on top of the Pioneer P3-DX unit, both connected to one computer using the rosaria, slam_gmapping, map_server, rviz, urg_node and tf packages. Through the slam_gmapping the map was generated, being stored with the map_server package. The rviz was used to visualize the mapping and the current robot position, based on the odometry, and the tf was used to keep the localization of the robot in the map reference. A second computer was used with the teleop_twist_package to control the robot from a distance. The final map was obtained using certain landmarks, which made it possible to go around the odometry imprecision.

C. Simulation

Gazebo software was used during the development of the code to test the behavior of the robot in a controlled environment without having to use the real robot. This expedited the code testing.

III. Implementation

This section describes how the EKF with its three basic steps has been implemented for the faced robot localization problem with a LRF. In subsection III-A the motion model and the observation model are discussed. The subsections III-B and III-C deal with the matching and the update step respectively. The resulting scheme of the implementation is shown in figure 1.

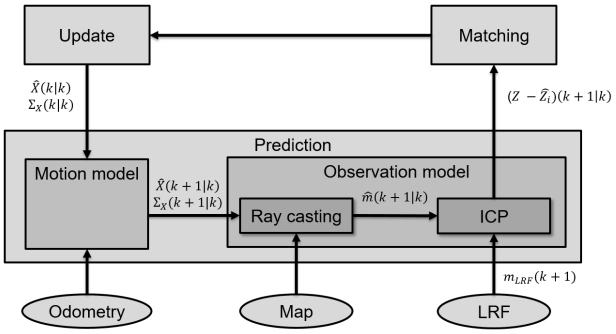


Fig. 1: Scheme of the implemented EKF

The robot's state X is defined as its current two dimensional pose. The observation Z used for the Extended Kalman filter, like explained in section III-B, also includes variables to describe a pose.

$$X = \begin{pmatrix} x_s & y_s & \theta_s \end{pmatrix}^T \quad (1)$$

$$Z = \begin{pmatrix} x_{obs} & y_{obs} & \theta_{obs} \end{pmatrix}^T \quad (2)$$

$$V = Z(k+1) - \hat{Z}(k+1) = \begin{pmatrix} \Delta x_v & \Delta y_v & \Delta\theta_v \end{pmatrix}^T \quad (3)$$

With the state and the observation being a 3 by 1 vector, the dimensions of all the variables used in the EKF are defined. Table I lists the used variables with their symbols, dimensions and a short description.

Symbol	Dimension	Description
X	3 x 1	State vector - Robot pose
V	3 x 1	ICP observation difference
Z	3 x 1	Observation
Σ	3 x 3	Robot state covariance
f	3 x 1	Motion model function
F	3 x 3	Motion model Jacobian
Q	3 x 3	Motion model noise
U	2 x 1	Motion model input
m^i	2 x 1	Laser beam measurement
h	3 x 1	Observation model function
H	3 x 3	Observation model Jacobian
R	3 x 3	Observation model noise
K	3 x 3	Kalman gain

TABLE I: Variables and functions used for the EKF

A. Prediction

The aim of the EKF prediction step is to receive the updated state and covariance and from there on determine the predicted observation for the next time step $\hat{Z}_i(k+1|k)$. To do so, the motion model predicts the future state in the next time step $\hat{X}(k+1|1)$ and the corresponding covariance matrix $\Sigma_X(k+1|k)$. From there on it is possible for the observation model to obtain the predicted observation $\hat{Z}_i(k+1|k)$. The motion and observation model need to be defined by the user depending on the problem and solution approach.

1) State Prediction: One of the main ideas of the EKF lies in linearising the motion and measurement model. For the state prediction, the motion model is decomposed as the noise-free part in (4) and a random noise component with zero mean in equation (5).

$$\hat{X}(k+1|k) = f(\hat{X}(k), U(k)) \quad (4)$$

$$\Sigma_X(k+1|k) = F\Sigma_X(k|k)F^T + Q(k) \quad (5)$$

As the considered robot is a three-wheeled vehicle that drives in a planar two dimensional environment, a motion model can be developed specifically for this case. The Pioneer 3DX is steered by giving different speed commands to the left and right wheel. Therefore, the motion input vector U is defined as in (6).

$$U = \begin{pmatrix} \omega_{Right} & \omega_{Left} \end{pmatrix}^T \quad (6)$$

$$\beta = \frac{\omega_{Right} - \omega_{Left}}{d} \quad (7) \qquad R = \frac{\omega_{Left}}{\beta} \quad (8)$$

With the help of d, the distance between the robot's driven wheels, and the defined parameters R and β in (7) and (8)

respectively, it is possible to formulate the motion model function in equation (9).

$$f = \begin{pmatrix} \hat{x}_s(k) + (R + \frac{d}{2})(sin(\hat{\theta}_s + \beta) - sin(\hat{\theta}_s)) \\ \hat{y}_s(k) + (R + \frac{d}{2})(-cos(\hat{\theta}_s + \beta) + cos(\hat{\theta}_s)) \\ \hat{\theta}_s(k) + \beta \end{pmatrix} \quad (9)$$

For calculation the predicted state covariance $\Sigma_X(k+1|k)$ with equation (5) the Jacobian F of the motion model function is mandatory. It can be calculated by derivating f with respect to the state X. [4]

$$F = \frac{\partial f}{\partial X} = \begin{pmatrix} 1 & 0 & (R + \frac{d}{2})(cos(\hat{\theta}_s + \beta) - cos(\hat{\theta}_s)) \\ 0 & 1 & (R + \frac{d}{2})(sin(\hat{\theta}_s + \beta) - sin(\hat{\theta}_s)) \\ 0 & 0 & 1 \end{pmatrix} \quad (10)$$

The Pioneer 3DX robot is able to determine its odometry off the shelf. By using backward differences the next state can also be predicted. With this solution approach the equations for the motion model function f and its Jacobian F can be simplified to equation (11) and (12).

$$f = \begin{pmatrix} \hat{x}_s(k) + \Delta x_{odom}(k) \\ \hat{y}_s(k) + \Delta y_{odom}(k) \\ \hat{\theta}_s(k) + \Delta \theta_{odom}(k) \end{pmatrix} \quad (11)$$

$$F = \begin{pmatrix} 1 & 0 & -\Delta y_{odom}(k) \\ 0 & 1 & \Delta x_{odom}(k) \\ 0 & 0 & 1 \end{pmatrix} \quad (12)$$

The motion model noise Q represents the trust in the state that has been predicted by the motion model. The noise, defined in equation (13), is implemented with a static and a dynamic component. The static component represents that the robot may even slip while the odometry does not notice any movement. The dynamic component mirrors that the odometry errors depend on the amount of movement within a time step.

$$Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} q_{x,stat} + q_{x,dyn}|\Delta x_{odom}(k)| \\ q_{y,stat} + q_{y,dyn}|\Delta y_{odom}(k)| \\ q_{\theta,stat} + q_{\theta,dyn}|\Delta \theta_{odom}(k)| \end{pmatrix} \quad (13)$$

2) Observation Prediction: The general task of the observation prediction is to use the predicted state from the described state prediction to determine a predicted observation. The implemented observation model is composed by two parts. The ray casting algorithm and the Iterative Closest Point (ICP) algorithm. The ray casting simulates a laser scan in the predicted position and the ICP tries to match this simulated laser scan with the real laser scan.

The ray casting algorithm enters the pre-acquired map, which is stored as an occupancy grid, at the predicted state $\hat{X}(k+1|k)$ and then simulates a laser scan, with the same maximum range, starting angle, ending angle and angle increment as the used laser rangefinder. The algorithm starts at the starting angle and searches for the first occupied grid cell in this direction. It calculates the distance between the predicted state position and the occupied cell and stores it. This is repeated for every laser beam until the ending angle of the real laser is reached. The total of all calculated values $\hat{m}(k+1|k)$ is then handed to the ICP algorithm in the same format as the real laser scan $m_{LRF}(k+1|k)$. The idea of the algorithm is shown in figure 2. The maximum accuracy is limited by the resolution of this map, which equals 0,05 $\frac{m}{pixel}$. [5]

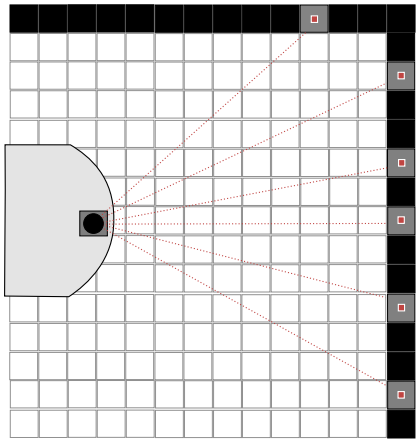
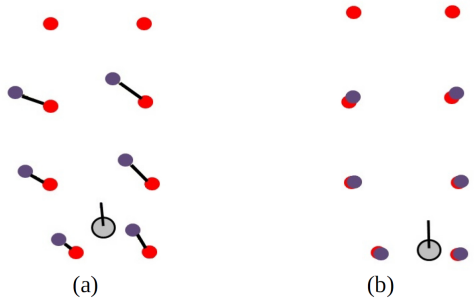


Fig. 2: Raycasting

The Iterative Closes Point algorithm (ICP) minimizes the distance between two clouds of points. One of the point clouds, red dots in figure 3, is the reference or target point cloud, which is kept fixed. The other one, purple dots in figure 3, the source point cloud, is transformed to best match the reference. [4]



(a) (b)

Fig. 3: Example ICP matching

The used ICP algorithm is part of the PCL C++ library. If the algorithm converged, it returns a four by four transformation matrix between the two point clouds. The transformation matrix includes a three dimensional rotation matrix and a translation vector. The observation difference vector V, defined in equation (3), between the real observation and the predicted one can be extracted from the ICP transformation matrix. For the case of two dimensional point clouds, the transformation matrix M has the below shown format.

$$M = \begin{pmatrix} cos(\theta_v) & sin(\theta_v) & 0 & \Delta x_v \\ -sin(\theta_v) & cos(\theta_v) & 0 & \Delta y_v \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

▭ : rotation matrix
▭ : translation vector

B. Matching

The matching step in the extended Kalman filter generally prevents updates with outliers. In the implemented EKF the matching is also responsible for checking the ICP algorithm quality and for determining the trust into the ICP's result.

To do so two mismatch conditions have been implemented.

- ICP did not converge
- ICP did not match more points than the a set threshold

If one of these conditions is fulfilled the next state $\hat{X}(k+1|k+1)$ and and the next covariance $\Sigma_X(k+1|k+1)$ will be set equal to the odometry predicted state $\hat{X}(k+1|k$ calculated with equation (4), respectively the odometry predicted covariance $\Sigma_X(k+1|k)$ of equation (5).

C. Update

If none of the mismatching conditions is fulfilled the update step will use the results from the ICP to combine them with the odometry predicted state to make a more precise prediction of the current state and covariance, based on odometry and LRF.

Because the observation model is based on the ICP, which will give results that are linear dependent to the predicted state $X(k+1|k)$, the Jacobian of the observation model H results in a 3 by 3 identity matrix.

$$H = \frac{\partial h}{\partial X} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{14}$$

With H being an identity matrix the formulas for the extended Kalman filter update step can be significantly simplified. The Kalman gain $K(k+1)$ can be calculated with only the predicted covariance $\Sigma_X(k+1|k)$ and the observation noise matrix R. The noise and inaccuracies form the laser rangefinder itself, the ray casting, the mapping and the ICP calculations are summed up in the observation noise matrix R. Therefore, the values tend to be much higher than for a laser rangerfinder alone.

$$K(k+1) = \Sigma_X(k+1|k) \cdot (\Sigma_X(k+1|k) + R(k+1))^{-1} \tag{15}$$

$$R = \begin{pmatrix} r_x & 0 & 0 \\ 0 & r_y & 0 \\ 0 & 0 & r_\theta \end{pmatrix} \tag{16}$$

The state and covariance update equations are given in (17) and (18). For the state update the observation difference vector V, calculated by the ICP algorithm, as well as the Kalman gain $K(k+1)$, are used. The next covariance only depends on the predicted covariance and the Kalman gain.

$$\hat{X}(k+1|k+1) = \hat{X}(k+1|k) + K(k+1)V(k+1) \tag{17}$$

$$\Sigma_X(k+1|k+1) = (I - K(k+1)) \cdot \Sigma_X(k+1|k) \tag{18}$$

IV. Results

To evaluate the implemented EKF a testing environment with a pre-defined trajectory was created. The robot would drive one rectangular shaped lap in one part of the pre-acquired map. In figure 4 the map is shown. Both, the odometry states and the EKF estimated stated plus their covariance, were saved in order to compare them afterwards.

The test results are shown in figure 5 with the covariance ellipses being calculated as 95 % confidence ellipses. With only the odometry measuring the current robot state, the calculated position after one lap, is about 15 cm away from the starting point. Due to poor odometry calibration, sensor errors or wheel slippage, the odometry accumulates a lot of errors during this short distance.

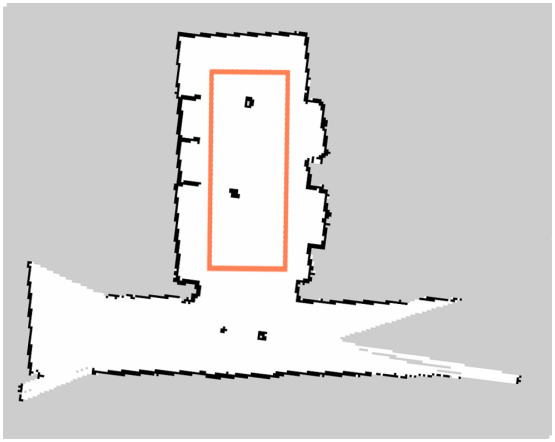


Fig. 4: Used map and trajectory for testing

The EKF predicted states stay much closer to the real trajectory. Every time the ICP is able to match the predicted and the real point cloud the accumulated errors are corrected by a laser based update. Some of these corrections can be recognised as a small jump of the EKF state in figure 5. Due to the modelled motion model with its noise matrix Q, the covariance grows during the times, where there are no matches. It then decreases heavily with a successful match. Therefore, the shown covariance ellipses are smaller after a jump in the state. The left part of the trajectory in figure 5 shows this process. Especially as the shape of odometry only and EKF are identical. The EKF only relies on odometry information at this part, before it is able to recover with a laser based update. In the end the EKF predicted state is only about 2.5 cm away from the actual starting position.

For obtaining the map displayed in figure 4 and evaluating the EKF's performance, some extra obstacles were placed in the environment. Although the implemented EKF was able to converge without these obstacles, the provided extra information by these reference objects improved its performance. Especially in the lower part of the map, with the two sideways being out of range of the laser rangefinder, the obstacles helped the EKF to match successfully.

During the testing of the system, the environment was not static. The environment consisted in an elevator room where people were walking around and doors were opened and closed. Although this was not the ideal environment, it was noticed that it did not affect the behaviour of the EKF algorithm at all. The EKF algorithm was able to remain stable in terms of position estimate and did not diverge, even though the pre-acquired map was not perfectly matching with the real world. Obviously this robustness to a dynamic environment

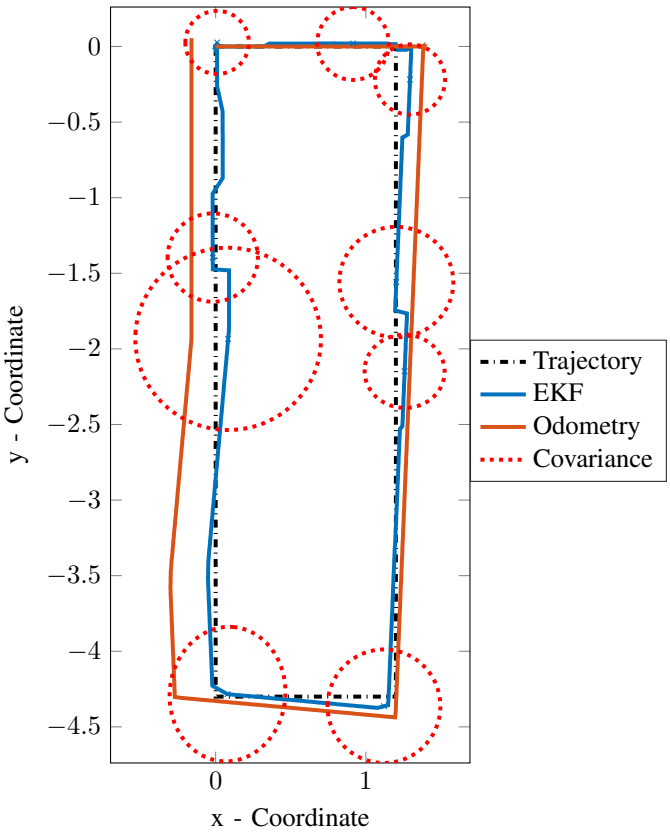


Fig. 5: Comparison between odometry only and EKF implementation

will diminish with an increasing amount of dynamic changes to the environment.

A. Improvements

While testing the implemented EKF in simulation and in the described real world test scenario some weak points have already been revealed.

The PCL'S ICP algorithm sometimes converged using either too few points or an observation that is not the same as the real one. Given this, the implemented EKF would sometimes match in wrong positions. To improve this, it would be necessary to modify the ICP's sensibility or use other filter algorithms before using ICP to match the point clouds, in order to avoid wrong matches. As the focus of this project was on implementing the EKF itself, improvements to the overall ICP performance were not further considered.

Algorithms like a scaling algorithm could potentially increase the ICP's performance. In situations where either the predicted or the real position is very close to an obstacle and the other one is far way from the distance, a scaling algorithm would prepare the point clouds for the ICP algorithm, by scaling them to similar magnitude. After this, points that represents the same feature in the real world should be of the same size and could be matched with the ICP algorithm.

Additional improvement could be achieved by using a map with higher resolution. The test map had a resolution of $0.05\frac{m}{pixel}$ and therefore could in the very best case only lead to an accuracy of this order.

B. Kidnapping

To create a robustness to kidnapping situations a special condition has been implemented in the EKF algorithm. Whenever 100 consecutive mismatches occure, the covariance matrix will be multiplied with the factor 10. This gives the EKF the possibility to search its position in a bigger area making it possible to encounter its real position. Despite being good for kidnapping scenarios, this method also includes an extra problem, as it can make the prediction diverge from the real position. If the EKF matches a wrongly predicted observation with the real observation only once, it will decrease its covariance and also calculate its position wrongly. Recovering from this will be a challenge just like using the normal EKF behaviour. To try and correct this problem we have a varying threshold of ICP algorithm. When we increase the covariance we also turn the sensibility of the ICP up by decreasing the threshold. This way we can avoid to some degree mismatches in the EKF algorithm.

The kidnapping robustness was tested and but only had satisfying results in situations where the kidnapped position was not too far away from the last predicted position. Otherwise the EKF most times matched with wrong positions that had a similar surrounding than the real position.

V. Conclusion

Instituto Superior Tcnico (IST) was created in 1911 from the division of the Industrial and Commercial Institute of Lisbon. Alfredo Bensade, an engineer, was IST's first dean (19111922) and promoted a wide-rang reform in the Portuguese higher technical education, including the first engineering courses at IST: mining, civil, mechanical, electrical, and chemical-industrial. IST's second dean was Duarte Pacheco (19271932), also an engineer, who was responsible for the construction of the university campus at Alameda. The architect Porfrio Pardal Monteiro designed it. Meanwhile, IST became part of the recently created Technical University of Lisbon. Throughout the following decade, the image of engineers from IST was projected into major engineering works, promoted by Duarte Pacheco, who was by the time Minister of Public Works.

References

[1] S. Thrun, W. Burgard and D. Fox, *Probalistic Robotics*, The MIT Press, 2005

[2] S.J. Julier and J.K. Uhlmann, *Unscented filtering and nonlinear estimation*, Proceedings of the IEEE: 401422., 2004

[3] L. Teslic, I. Skrjanc and G. Klancar, *EKF-Based Localization of a Wheeled Mobile Robot in Structured Environments*, J Intell Robot Syst, 2010

[4] T. Maneewarn and K. Thung–od *ICP-EKF Localization with Adaptive Covariance for a Boiler Inspection Robot*, 2015 IEEE Conference on Robotics, Automation and Mechatronics,2015

[5] E. Ivanjko, I. Petrovic *Extended Kalman filter based mobile robot pose tracking using occupancy grid maps*, Proceedings of the 12th IEEE Mediterranean, 2004