Extended Kalman Filter (EKF)-based Local SLAM in Dynamic Environments: A Framework

Horatiu George Todoran[1] and Markus Bader[2]

[1]Automation and Control Institute, Vienna University of Technology, Austria
george.todoran@tuwien.ac.at
[2] Institute of Computer Aided Automation, Vienna University of Technology, Austria
markus.bader@tuwien.ac.at

Abstract. In the domain of mobile robots local maps of environments are used as knowledge base for decisions to allow reactive control in order to prevent collisions by following a global trajectory. These maps are normally discrete and updated with a relatively high frequency, but with no dynamic information. The proposed framework uses a sparse description of clustered scan points from a laser range scanner. These features and the system odometry are used to predict the agent ego motion as well as feature motion using an Extended Kalman Filter. This approach is similar to a Simultaneous Localization and Mapping (SLAM) algorithm but with low-constraint features. The presented local Simultaneous Localization and Mapping (LSLAM) approach creates a decision base, holding a dynamic description which relaxes the requirement of high update rates. Simulated results demonstrate environment classification and tracking as well as self-pose correction in static and in dynamic environments.

Keywords: EKF, SLAM, adaptive filtering, dynamic descriptors, grouped data

1 Introduction

Despite the fact that a lot of research has been conducted on environment mapping, most of the approaches still assume it to be static. As a consequence, tasks assigned to mobile agents have been generally solved constraining the underlying algorithms and approaches to such an assumption. Thus, classical approaches to Simultaneous Localization and Mapping (SLAM) such as FastSLAM [5] or ICP-based [6] encounter difficulties and often-times lead to divergence in dynamic unstructured environments. Regarding map based localization-tasks, considerable modification of the layout imposes offline re-mapping. Even in local path-planning, common approaches treat sensor data at each iteration as static obstacles, relying on relatively high control frequencies to deal with dynamic environments.

Local maps typically consist of close-vicinity representations of the environment. As they represent the closest layer of perception in relation with the agent dynamic tasks (path-following, grasping etc.) [7], they are required to be accurate, online and

adfa, p. 1, 2011.

descriptor-rich. However, traditional approaches generate local-maps without any dynamic descriptors of the local environment.

The presented framework proposes an alternative to local map-creation and environment description for mobile agents in an online, fast-computing manner. Thus, the environment is modelled and grouped as rigid dynamic objects, treating object discovery, time-out, merging, splitting and symmetries. Using the acquired information regarding the objects dynamic state, agent self-pose correction is performed, enchanting local map-building to local SLAM (*LSLAM*). In addition, the framework outputs the classified objects and their dynamic descriptors for further usage in self-localization, mapping, path-planning, sensor-fusion and other robotics tasks.

Grouping of data in higher level features--objects has been widely studied in computer vision and robotics communities and recently proposed in SLAM approaches [1]. However, this work aims to include high-level features in a more complex SLAM problem, where dynamic entities are present. Dynamic object tracking has been addressed by Montessano [2] in his PhD. thesis, analyzing various filtering techniques. Bar-Shalom et al [4] analyses dynamic object tracking as well and presents a process noise for velocity-bound and acceleration-bound models assuming a Wiener-process.

This paper is organized as follows: Section 2 presents in detail the approach of the LSLAM EKF estimator, followed by simulated results for pose-correction and data association, presented in Section 3. Conclusions are drawn in Section 4.

2 Approach

Fig. 1 presents the overview of the framework, including its modules and their input-output data. The laser sensor data undergoes a three-layered abstraction, from points (p) to segments ($\mathcal{S}$) and finally objects ($\mathcal{O}$).

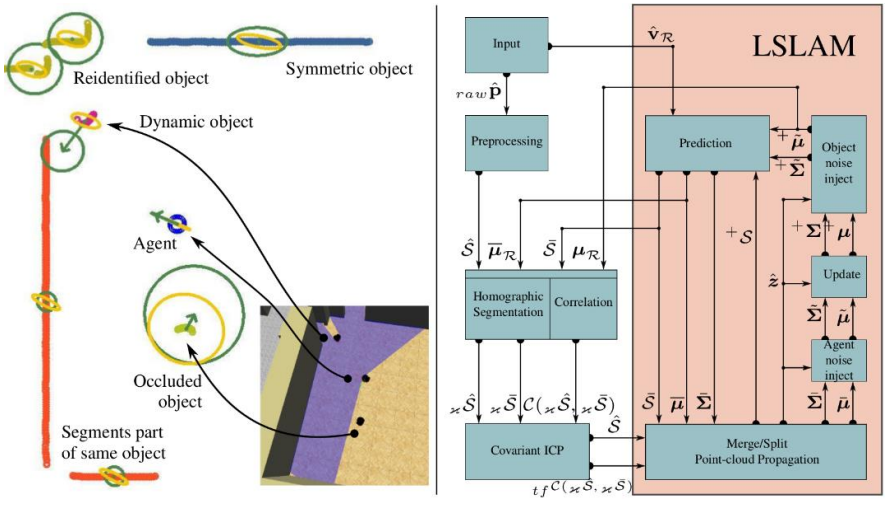

Fig. 1. Generated local map showcasing various non-trivial scenarios (left), Framework overview (right)

The input point-cloud ${}_{raw}\hat{\mathbf{p}}$ is segmented under homography constraints and its constituting segments are associated. Pose displacements corresponding to each segment pair along with uncertainty is computed by the *Covariant ICP* module. The loop is being closed an Extended Kalman Filter (EKF) type estimator implemented in the *LSLAM* module. Based on the agent predicted state $\overline{\boldsymbol{\mu}}_{\mathcal{R}}$, correlated segments information and objects observations $\hat{\boldsymbol{z}}$, the estimator updates the state of the world. In the following, the underlying approach of the *LSLAM* module is presented.

Prediction

Agent prediction. Given the mapped state of the agent including its velocities and accelerations, assuming a constant acceleration model, its state is being predicted. The noise covariance matrix is being modelled as parameterized *dt*-proportional, using 2 parameters (β). The Jacobian of the agent's motion model with respect to the noise parameters *V* (1) maps the noise covariance matrix in the agent's state space (2).

$$\overline{\boldsymbol{\mu}}_{\mathcal{R}} = \mathbf{g}_{\mathcal{R}}(\boldsymbol{\mu}_{\mathcal{R}}, dt) + \overline{\boldsymbol{\alpha}}_{\mathcal{R}}(dt), \qquad \overline{\boldsymbol{\alpha}}_{\mathcal{R}}(dt) = \mathbf{V}\mathbf{M}(dt)\mathbf{V}^{T} \quad (1)$$

$$\mathbf{V} = \frac{\partial \mathbf{g}_{\mathcal{R}}}{\partial \boldsymbol{\mu}_{\mathcal{R}_a}}, \qquad \boldsymbol{\mu}_{\mathcal{R}_a} = \begin{bmatrix} a_{lin} \\ a_{ang} \end{bmatrix}, \qquad \mathbf{M}(dt) = dt \begin{bmatrix} \beta_{\overline{\mathcal{R}},0} & 0 \\ 0 & \beta_{\overline{\mathcal{R}},1} \end{bmatrix} \quad (2)$$

Agent pseudo-observation .In many situations, odometry information is subject to inconsistent noise (time delay, drifts, encoder errors, wheel slips, deviation from linearization point) and furthermore impose a refinement of the agent prediction method. Thus, at the end of the prediction step, the mapped system is being pseudo-updated with the agent's control input. The noise of the observation is modelled as having a base value and a state-velocity-proportional value (3).

$$\hat{\boldsymbol{\alpha}}_{\mathcal{R}}(dt) = dt^2 \begin{bmatrix} (\beta_{\mathcal{R},0} + \beta_{\mathcal{R},1}|\overline{v}_{lin}|)^2 & 0 \\ 0 & (\beta_{\mathcal{R},2} + \beta_{\mathcal{R},3}|\overline{v}_{ang}|)^2 \end{bmatrix} \quad (3)$$

Object prediction. Assuming a constant-acceleration model, the objects are being predicted. The object's pose prediction has no real use but pose covariance is being predicted and propagated over time. However, point--clouds that are belonging to an object are being predicted based on its dynamic state and propagated accordingly to ensure robust segment-correspondences. The object process noise $\overline{\boldsymbol{\alpha}}_{\mathcal{O}}\ dt$ is being modelled as described by Bar-Shalom [4] under the assumption of continuous-time white noise (Wiener process).

$$\overline{\boldsymbol{\mu}}_{\mathcal{O}_i} = \mathbf{g}_{\mathcal{O}_i}\left(\boldsymbol{\mu}_{\mathcal{O}_i}, dt\right) + \overline{\boldsymbol{\alpha}}_{\mathcal{O}}(dt) \quad (4)$$

Merging/Splitting analysis

In situations when object merging or splitting is proposed (object merge/split is flagged when observing multiple segments corresponding to one or vice-versa) the strength of computing transforms covariance is exploited. The means and covariances of the proposed-to-merge segments/objects are being pair-wise combined forming a

new mean $\boldsymbol{\mu}_{ij}$[1]. The cost function of the two transforms "neighboring level" is modelled as the product of the two initial distributions, evaluated in $\boldsymbol{\mu}_{ij}$. The pair-wise merging happens if the cost function is bigger than a threshold. In case a flagged merge or split is evaluated as valid, the state vector is resized and segments are associated to the new object configurations.

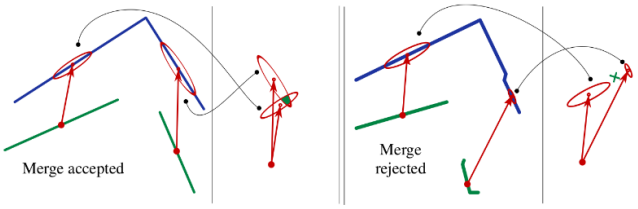

Fig. 2. Object merging/splitting scenarios

Update

The general object observation function uses the computed translational and rotational deviation of corresponding segments along with its uncertainty, as computed by the *Covariant ICP* module. As the center of mass of the matched point-clouds changes over time, the predicted pose state of each object has to be initialized before update (5), according to its new center of mass (computed by the *Covariant ICP* module).

$$\overline{\boldsymbol{\mu}}_{\mathcal{O}_i:x,y,\theta} = \mathbf{h}^{-1}\left(\overline{\mathbf{z}}_{\mathcal{O}_i}\right), \qquad \overline{\mathbf{z}}_{\mathcal{O}_i} = \begin{bmatrix} \overline{x}_x \\ \overline{x}_y \\ \overline{x}_\theta \end{bmatrix}_{\mathcal{O}_i}^{\overline{\mathcal{R}}}, \qquad \hat{\mathbf{z}}_{\mathcal{O}_i} = \begin{bmatrix} \hat{x}_x \\ \hat{x}_y \\ \hat{x}_\theta \end{bmatrix}_{\mathcal{O}_i}^{\overline{\mathcal{R}}} + \hat{\boldsymbol{\alpha}}_{\mathcal{O}_i} \quad (5)$$

Adaptive filtering

As the observed environment assumes a general low-constraint model, with no correlation between objects, agent pose-correction is desired to happen only when sufficient observed objects are in steady-state. Thus, additional uncertainty of the tracked objects is being injected as a scaled entry of their base prediction noise (6). The scale factor $s_{\mathcal{O}_i}$ is proportional to the Mahalanobis distance of the object residual $\mathbf{q}$ in the velocity probability distribution[2] $\boldsymbol{\Sigma}_{\mathcal{O}_{i_v}\mathcal{O}_{i_v}}$. The object adaptive noise provides better object state estimation in situations of under-scaled base prediction noise in highly-dynamic tracked motions (e.g. circular trajectory with small radius at high velocity).

$$\tilde{\boldsymbol{\Sigma}}_{\mathcal{O}_i\mathcal{O}_i} = \overline{\boldsymbol{\Sigma}}_{\mathcal{O}_i\mathcal{O}_i} + s_{\tilde{\mathcal{O}}_i}\,\overline{\boldsymbol{\alpha}}_{\mathcal{O}}(dt) \quad (6)$$

$$s_{\tilde{\mathcal{O}}_i} = {}^{t}\mathbf{q}^{T}\,\boldsymbol{\Sigma}^{-1}_{\mathcal{O}_{i_v}\mathcal{O}_{i_v}}\,{}^{t}\mathbf{q}\,, \qquad \mathbf{q} = \hat{\mathbf{z}}_{\mathcal{O}_i} - \overline{\mathbf{z}}_{\mathcal{O}_i} \quad (7)$$

---

[1] Merging uncertainty computed through Gaussian distributions multiplication

[2] Evaluating the Mahalanobis distance ensures scalability and behaves equally for various pose uncertainties of the system

The agent state convergence is evaluated in a similar fashion to Congwei Hu et al [3]. However, instead of using only the predicted residuals, the total Mahalanobis distances of all the objects residuals at time *t* and the same metric for the last *N* time steps is being evaluated (9). This way, large additional agent noise will not be triggered when the observed system is closer to unsteady state. In practice, the additive noise is triggered if $s_{\tilde{\mathcal{R}}} > \beta_{\mathcal{R}}$. The additive noise $\tilde{\boldsymbol{\alpha}}_{\mathcal{R}}(dt)$ injected in this phase (8) can have various forms depending on the expected drifts of the agent from the model (time skew, slips, short distance kidnappings, frame-rate drop etc.).

$$\tilde{\boldsymbol{\Sigma}}_{\mathcal{O}_{\mathcal{R}}\mathcal{O}_{\mathcal{R}}} = \overline{\boldsymbol{\Sigma}}_{\mathcal{O}_{\mathcal{R}}\mathcal{O}_{\mathcal{R}}} + s_{\tilde{\mathcal{R}}}\tilde{\boldsymbol{\alpha}}_{\mathcal{R}}(dt) \quad (8)$$

$$s_{\tilde{\mathcal{R}}} = \beta_{\tilde{\mathcal{R}}} \frac{{}^{t}\mathbf{Q}^{T}\,{}^{t}\boldsymbol{\Sigma}^{-1}_{\mathcal{O}_v\mathcal{O}_v}\,{}^{t}\mathbf{Q}}{\frac{1}{N}\sum_{i=t-N}^{t-1}{}^{i}\mathbf{Q}^{T}\,{}^{i}\boldsymbol{\Sigma}^{-1}_{\mathcal{O}_v\mathcal{O}_v}\,{}^{i}\mathbf{Q}} - 1\,, \qquad {}^{i}\mathbf{Q} = {}^{i}\hat{\mathbf{z}}_{\mathcal{O}} - {}^{i}\overline{\mathbf{z}}_{\mathcal{O}} \quad (9)$$

It is worth noting that during a cycle of the filter, the agent adaptive noise injection process is evaluated first. After the agent noise injection and all object updates, the object residuals are recomputed (using the updated agent information) for the object noise injection. This way, erroneously high values of object injected noise due to pre-update agent state divergence is avoided.

Initialization and point-cloud propagation

As more objects get discovered, they are appended and initialized according to the agent's uncertainty and their initial observation. Especially for moving objects, keeping their state when they are out of view is not required, their uncertainty becoming so big that correct data association is not feasible. Thus, each object is being discarded when its velocity uncertainty ellipse surface exceeds certain thresholds (static out-of-view objects are remembered and potentially used for loop closures).

In the current stage, object reconstruction is not implemented, thus for short-term mapping the "longest" point-cloud from a match is being propagated, given that it provides enough up-to-date information about the current observed segment.

3 Simulated results

All the experiments are being conducted using the Gazebo simulation environment under Robotics Operating System (ROS). The agent is a Pioneer-P3DX mobile robot equipped with a Hoyuko laser range sensor and its angular and linear velocities are constrained to certain measured acceleration limits ($a_{max} \sim 0.4m/s^2$). As ROS publishes laser and odometry data unsynchronized at a frequency of 10Hz, a message-filter is used for message synchronization. Thus, the frequency of the framework is set to 8Hz, ensuring input data synchronization without loop-skips. The input data from the simulator is being noised, with fixed Gaussian noise for the laser sensor range readings ($\sigma_{\phi} \sim 0.01m$) and proportional Gaussian noise for the observed agent velocities ($\sigma_{v} \sim |v| \cdot 0.02m/s$). The framework provides three visualization tools, one for ICP results, transforms and error metric and other two for agent local view and world

view. Within the viewers, 3σ pose uncertainty is depicted in yellow while the velocity mean and uncertainty in green. For all experiments, the agent is tele-operated.

Object correspondences

Even though the framework provides short-term memory differential mapping, higher-level features such as object correspondences are being extracted. The following set of experiments illustrates the capabilities of the framework to successfully track identified objects in non-trivial scenarios.

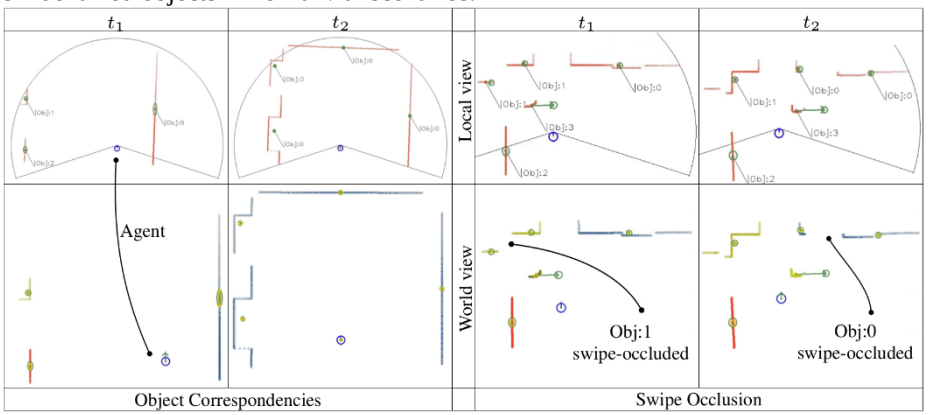

Fig. 3. Semantic memory in various scenarios

Merging/splitting. Given finite and relatively small sensor range with respect to the environment size as well as occluded areas of environments, the agent is supposed to map segment correspondences to their true state. Thus, the experiment presents object discovery and merging with segment splitting. At time t_1, the agent observes 3 segments, assumed to be different objects. As more information is acquired, the objects get merged so that at t_2, all observed segments are evaluated as the same object.

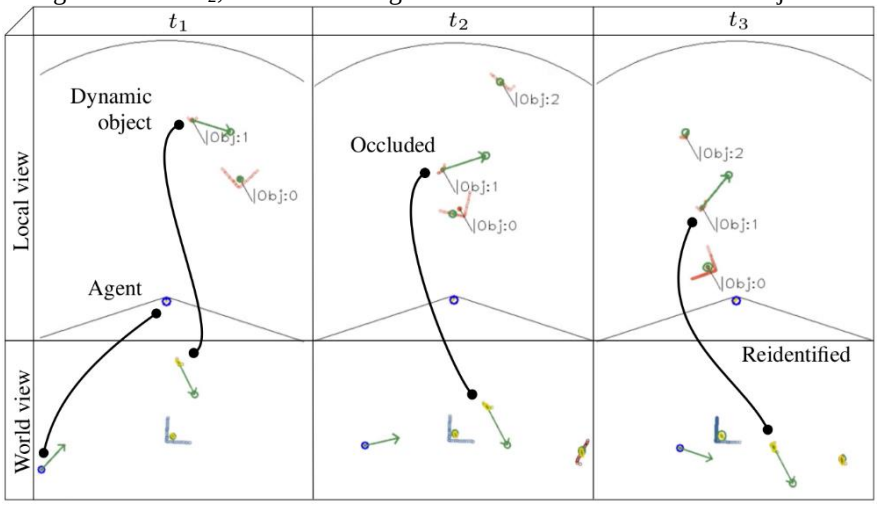

Fig. 4. Occluded tracking

Swipe-occlusion detection. Most of the scenarios that involve dynamic objects and static objects in the background will undergo "swiped-occlusion" situations, in which the dynamic object will partially occlude from one end to the other the static background. In such situations, propagating the last observed point-cloud will fail to re-identify the background object after the "swipe". However, as presented in 2.5, this approach achieves such semantic memory.

Occluded tracking. Short-term occlusion of objects are often present scenarios in dynamic environments. Especially for path-planning tasks, trajectory optimization and even collision avoidance can be achieved in case of short-term occluded memory, the trajectory of the object being remembered. In this experiment, the agent is undergoing a circular motion and observes a moving object, which becomes fully occluded. However, when the object reappears in the field-of-view, it is correctly matched.

Self-pose correction

The following experiments present agent self-pose correction assuming dynamic as well as static environment. The encoder drifts have been achieved by pausing the simulation at certain time-steps and manually modifying the agent pose.

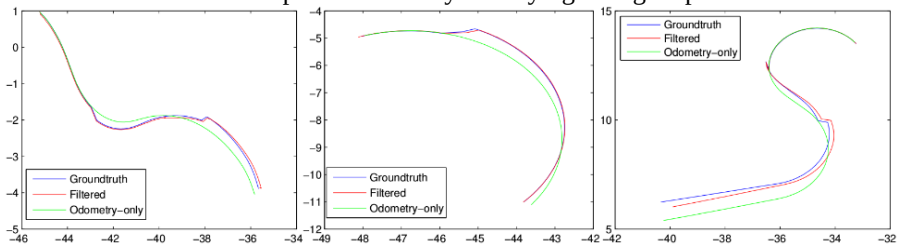

Fig. 5. Self-pose correction in dynamic environments

As described in Section 2, the pose of the agent is expected to be corrected with high degrees of accuracy as long as parts of the environment are in steady state. Fig. 5 presents the filtered trajectory of the robot when it undergoes short-term deviations from the motion model in various scenarios (*m*:*m*).

Even though the framework has been designed for agents with relatively accurate odometry values and a dynamic word assumption, it can as well be operated assuming a static environment. The following tests have been conducted feeding into the framework a fix measured angular and linear velocity set to *0* (odometry impairment).

Fig. 6 presents estimated trajectories of the agent (left, *m*:*m*) and the estimated linear and angular agent velocities(right, *m*/*s*:*s*).

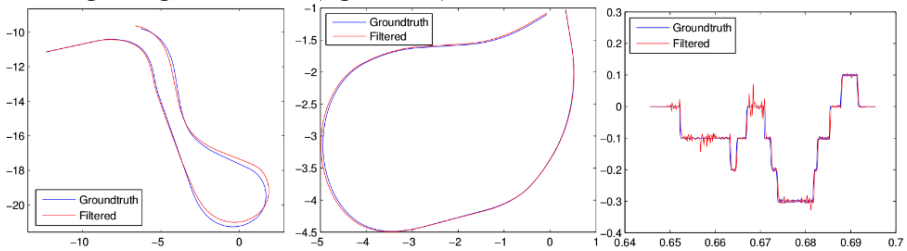

Fig. 6. Self-pose correction (left) and agent angular velocity (right) in static environments

4 Conclusions and future work

Agent self-pose correction in dynamic environments is still a weakly addressed problem within the robotics community. The presented framework has proven to extract sufficient information from partially steady-state dynamic environments, even though low constraint models of the environment and agent are assumed. By limiting environment perception to 2D, complexity of the problem is low enough for fast-online computation. However, especially in structured environments, the reduced dimensionality makes data association difficult given high probability of symmetries. Thus, image processing techniques such as *Homographic Segmentation* and *Covariant ICP* have been developed and used to increase robustness of data association.

As laser range sensors represent a norm in mobile agents nowadays, their accuracy is assumed by the framework to extract higher-level features from the environment: Objects. Such an approach proves to simplify EKF SLAM, reducing drastically the state vector size and thus computation time, even with detailed assumed models.

The focus of the presented work is on short-term memory reasoning of state point-clouds. Thus, long-term mapping and symmetry-robust loop-closures have not been addressed. However, simulated results prove that problems such as occluded tracking, gradual total occlusion and semantic memory are addressed and solved even with the above mentioned limitations. Shifting the focus towards long-term memory tasks, incorporating offline maps as constrained static objects will improve results.

Research could be focused on higher-level tasks such as local path-planning, using approaches such as Model Predictive Control (MPC) or Dynamic Window Approach (DWA), generalized for dynamic obstacle states. Such approaches should provide complex behaviors such as avoidance maneuvers due to external dynamic objects.

Last but not least, when high robustness and maturity of the framework is achieved, open-source publication of the code as a ROS node is desired, inviting the robotics community to take advantage of a package that provides pose-correction and dynamic descriptors of the environment for future research in various related fields.

References

1. S. Choudhary, A. J. B. Trevor, H. I. Christensen, F. Dellaert, SLAM with object discovery, modeling and mapping, IROS-2014
2. L. Montesano, Detection and tracking of moving objects from a mobile platform. Application to navigation and multi-robot localization, Universidad de Zaragoza, 2006
3. C. Hu, W. Chen, Y. Chen, D. Liu, Adaptive Kalman Filtering for Vehicle Navigation, Journal of Global Positioning Systems 01, 2003
4. Yaakov Bar-Shalom, X. Rong Li, Thiagalingam Kirubarajan, Estimation with Applications to Tracking and Navigation, John Wiley & Sons, 2001
5. M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem, In Proceedings of the AAAI National Conference on Artificial Intelligence, 2002
6. A. Nüchter, K. Lingemann, J. Hertzberg, H. Surmann, 6D SLAM—3D mapping outdoor environments, Journal of Field Robotics, 2007
7. R.B Rusu, I.A. Sucan, B. Gerkey, S. Chitta, M. Beetz, L.E. Kavraki, Real-time perception-guided motion planning for a personal robot, IROS-2009