AMCL Alternatives for ROS 2's Navigation Stack

Navigation2 Working Group

June 2020

1 Introduction

1.1 Purpose

This file contains the result of a survey of research in localization (June 2020) which aims to find compelling alternatives to the current AMCL ROS 2 package [5]. The major requirements to satisfy are its ability to work with 2D and 3D LiDAR data and its velocity.

1.2 AMCL Overview

Adaptive Monte-Carlo Localization (AMCL) is a probabilistic localization system that differs itself from the original Monte-Carlo Localization (MCL) by the usage of Kullback-Leibler Distance (KLD) sampling [4], which uses the Kullback-Leibler Distance (or relative entropy) to measure the approximation error in order to generate less samples when the uncertainty (reflected in the particle dispersion) is smaller and more when it is bigger. That dynamic (or *adaptive*) number of particles is what gives this MCL enhancement its name.

This localization technique is based in a Particle Filter (PF) that, as explained by roboticsknowledgebase.com [7], executes the following steps in a loop:

- **Re-sampling**: Replaces a random sample from the sample set according to the importance weights of the samples. If the belief on the hypothesis represented by a particle is high, the chance of it being replaced will be small (however, it could still happen).
- **Sampling**: Use the previous belief and the control information to sample the solution space. i.e. generate new hypothesis of the current robot pose.
- **Importance sampling**: Compute the *importance weights* of each particle, which will be based on the likelihood of the new measurement and what should be observed assuming the particle's hypothesis on the robot pose.

Further details on tricks and innovations in PFs can be found in the Sebastian THRUN's paper *Particle Filters in Robotics* [11].

Regarding the **strengths and weaknesses** of AMCL, Unlike Kalman Filter (KF)-based approaches, since a particle filter is used, it is a multi-modal state estimator. However, the solution space must be *sampleable*. i.e. small enough to sample it properly with an amount of discrete particles that can be managed by the available hardware.

2 Potential Alternatives

2.1 Normal Distributions Transform MCL (November 2013, 36 impact)

Oriented to industrial Automatically Guided Vehicles (AGVs), it prioritizes precision and repeatability. The Normal Distributions Transform MCL (NDT-MCL) [8] uses the Normal Distributions Transform (NDT) to represent both the map and the sensor data in the MCL algorithm, relaxing the discretization of the traditional algorithm. The authors evaluated it using offline data sets including real-world environments and compare the results to the AMCL ones.

Before this work, traditional MCL was used with KLD-Sampling (known as AMCL), achieving repeatability errors of around 6 cm, that was more reduced with scan matching, which only worked for holonomic vehicles.

Instead of considering the **measurement** of each instant a set of N_t points (as is done in MCL and AMCL), NDT-MCL [8] considers a set of N_zt normal distribution parameters (mean and covariance). Since the sensor will have some noise and thus, the distribution associated with each measure will cover a region of the grid map, N_zt is very likely to be way smaller than N_t. Moreover, NDT would already be a likelihood model of how likely is for an object to be in a certain location. In the case of the **map**, it will be also a set of normal distribution parameters.

With this representation, the **likelihood of an observation** given a map and an state (robot pose), can be computed to obtain a new distribution. However, in practice, the map cell corresponding to the rotated and translated mean is found and, then, the algorithm will search the local neighborhood to find the closest normal distribution to the computed mean. The paper also specifies how to compute the **weights** of the particles and specifies the **resampling** approach they use. They assume a given state initialization (mean and variance).

Regarding the **tests**, some of them were performed using a real robot in a 25x25 space and others, using data sets. Experiments with different amounts of dynamic objects were performed. AMCL achieved 5.4 cm at its best, with errors increasing with the grid size of the map. NDT-MCL [8] achieved 1.4 cm at its best and 2.8 cm at its worst (being this last case with a cell size of 1.8 m).

As a more in-depth **analysis**, AMCL is very dependent on the map resolution and the error fluctuates more. When introducing odometry perturbations or dynamic objects, the error also augments (in a similar fashion as with static maps) with the cell sizes. Regarding NDT-MCL [8], it has an almost constant performance for static maps regardless the cell sizes. However, when introducing dynamics objects, the error augments up to the ones obtained with AMCL and big cells, even it does not fluctuate much when varying the map resolution. For odometry perturbations, the performance is quite similar to the original test.

In **conclusion**, NDT-MCL [8] seems a very feasible alternative to AMCL that may optimize the memory requirements of the algorithm while improving its performance, specially with bigger maps that require bigger cell sizes. However, the presence of dynamic objects will be a problem regardless the cell size (see fig. 6 of the paper for reference).

Given the special interest of this proposal, **NDT-MCL-based methods**]has been researched. For this purpose, lists from Google Scholar and ResearchGate were used. The works on them that were considered most relevant will be explained on the following sub-subsections.

2.1.1 Dual-Timescale NDT-MCL (May 2014, 20.2 impact, ICRA2014)

Compared with NDT-MCL [8], Dual-Timescale NDT-MCL (DT-NDT-MCL) [12] uses two maps, a static map and a stort-term map. NDT occupancy grid was used for the short-term map and it was utilized only when and where the static map failed to give sufficient explanations. This method makes algo-

rithm provides equal performance as the original NDT-MCL [8] in stationary environments and superior performance in dynamic environments with semi-static objects.

2.1.2 Comparison and Discussion of Observation Models (November 2018, 12 impact)

This paper [1] compares four observation models (LFM, NDT-based model, COM and PMoEP-based model) developed for localization in highly dynamic environments.

- The **LFM** may correspond to dynamic obstacles and work in many cases, but it cannot correspond to landmark removal and/or movement.
- The **NDT** cannot correspond to landmark removal, but can increase the estimation accuracy when a high resolution ND map is used. However, the memory usage increases as the resolution increases.
- The **COM** can correspond to both dynamic obstacles and landmark removal. Additionally, its computational complexity is identical to that of the LFM. We concluded that the COM is suitable for localization in highly dynamic environments in terms of computational complexity and estimation accuracy.
- The **PMoEP** can accurately estimate a pose of an ego vehicle, even when the environment changes significantly. However, the PMoEP has a drawback in terms of computation time, and an acceleration method is required for use in practical situations.

Therefore, if there is enough time, it may be possible to use COM instead of the NDT method.

2.1.3 IRON (September 2015, 7 impact, IROS 2015)

This paper [9] introduced the IRON interest point detector and the IRON descriptor and illustrated how they can be used to align NDT-maps with high accuracy and speed.

The C++ source code of the IRON keypoint detector and the IRON descriptor, together with the complete registration algorithm can be downloaded at the website of the Technical University of Ilmenau.

2.2 Multi-Resolution Gaussian Mixture Maps (April 2017, 29 impact)

Multi-Resolution Gaussian Mixture Maps [14].

Too focused on self-driving cars? (road surface reflectivity). It seems it proposes a way to make maps more compact (and how to use them for localization) rather than proposing an actual localization method.

2.3 L3-Net: Learned LiDAR Localization (September 2019, 14 impact)

L3-Net [6] is a learning-based LiDAR localization system that achieves centimeter-level localization accuracy comparable current (September 2019) hand-crafted state-of-the-art techniques. It uses several Deep Neural Networks (DNNs). The authors collected multiple sequences for the testing dataset over the same road and areas for testing purposes. It contains sequences with a year's time interval, 360° LiDAR and more than 380 km of real-traffic driving distributed in 6 different routes.

This approach substitutes the traditional modules of a localization algorithm in the following way:

1. Extract **keypoints** evaluated by their linearity and scattering (defined by the eigenvalues of the neighbors of each point).

2. A single mini-PointNet extracts feature **descriptors** to encode statistical properties of the points, trained prioritizing robustness.

3. Usage of 3D convolutions (as already done in learning-based stereo vision) to **improve the localization accuracy**.

4. The matching probabilities of all the dimensions are computed and, therefore, the **optimal estimation** is obtained.

As an additional note, the **motion dynamics** (usually modeled with filtering method such as PFs), are implicitly encapsulated by deep Recursive Neural Networks (RNNs).

The **inputs** are a pre-built 3D point cloud map and the online LiDAR point cloud. The LiDAR-obtained point cloud is down-sampled with voxel grid. The dynamic objects are mostly removed from the map using semantic segmentation provided by PointNet++. The framework will also take as an input a predicted pose, usually generated by an Inertial Measurement Unit (IMU) or by the motion model of the vehicle. The **output** will be the optimal offset between the final and the predicted pose.

The **result** was a system able to achieve errors between 3 and 5 cm in Real-Time (RT), processing each time in 121.3 ms (with a GTX 1080 Ti GPU, an i7 9700K CPU and 16 GB of RAM).

In **conclusion**, this approach is quite focused on autonomous driving. It does not generalize to mobile platforms, so some adaptations may be needed. It also seems to be opening the way to learning-based approaches, so their trustability is yet to be proved (and its performance, improved). Overall, its behavior is both accurate and fast, but its dependence in GPU makes it privative for its usage in robots that do not have this hardware available.

2.4 PoseMap: 3D LiDAR Localization (October 2018, 4 impact)

PoseMap [3] is a long-term localization approach conceived for 3D LiDAR sensors. The method consists in extracting feautres from the LiDAR measurements and matching them using a sliding window fashion. It also allows replacing and adding local maps when necessary. They use a LiDAR and an IMU and achieve robust localization results at 8 Hz in dynamic environments (industrial and off-road environments during 18 months and 100 km without failure).

This localization method is based on C-SLAM [2], which will be explained next. For each LiDAR scan, features (*surfels*) are extracted from the new point cloud and projected into the map following the data obtained using an IMU and/or odometry. Next, the projected surfels are matched with those already in the map with a K-Nearest Neighbours (K-NN) search, using the obtained correspondences to build constraints for the optimization. Deviations from the IMU measurements are penalized to ensure smoothness. Continuity with the previous trajectory is enforced providing initial condition constraints. The optimization is solved with an iterative re-weighted least squares in an M-estimator. For the loop closures, whenever a significant translation or rotation is performed, a set of surfels is stored.

For the proposed localization method, a combination of local constraints (matching local features) and matching between the local and the map features was used. Without using the constraints, the method runs at more than 20 Hz, but the robustness to changes in the environment drops (e.g. vegetation changes). Using a sliding window for the matching, the frequencies are between 4 and 10 Hz.

The method includes **online map extensions and updates**. If the map is left, the new discovered features are added. The map nodes are continuously evaluated for major changes, being updated when a substantial (and static) change occured.

As a **conclusion**, the paper seems to focus more in the map format and how to use it for localization rather than innovating in localization techniques. This work feels like more oriented to extend/adapt

a SLAM method to solve the localization problem (providing a higher update rate). The results seem accurate, but only qualitative proof is shown.

2.5 RT 3D LiDAR Localization (December 2019, 9 impact)

This approach [15] consists in dividing the map in smaller point clouds to speed the registration and improve the robustness and fuse several odometers in the *PointLocalization* method, optimizing the accuracy and frequency of the localization. The method provides state estimations at 20 Hz with accuracies of around 10 cm. It has been tested for more than six months in a crowded factory and operated successfully after more than 2000 km.

PointLocalization requires two **sensor inputs**, one being the current surrounding environment and the other being any kind of odometry. In the paper, a Hall effect sensor is used as an encoder measuring from a 36T gear, which allows to detect complete rotations and, thus, velocity. The velocity is provided at 10 Hz, minimum rate required for the odometry input of the method.

After receiving the LiDAR and odometry messages, the current odometry is applied to the last LiDAR localization, providing a prediction of the robot's pose. Another way of getting a prediction (if odometry information is not available) is using the LiDAR momentum, which is done by computing the transformation between the last two LiDAR measurements and assuming that it has been repeated. One of those estimations will be used to determine the Region of Interest (ROI) on the map that will be used to match the current LiDAR input, which is created by expanding the (odometry or LiDAR momentum based) estimated current pointcloud one meter. Finally, Iterative Closest Point (ICP) is used to register the source point cloud (current LiDAR scan) in the target point cloud (ROI sub-map).

Finally, the odometry and LiDAR results are fused using the Steady-State Approximation of Extended Kalman Filter (SSKF) [13]. SSKF [13] is used instead of the traditional Extended Kalman Filter (EKF) approach because the EKF can only take measurements from the current state, while with this method, delayed measurements may be used.

After the **experimentation** with the KITTI dataset, the authors show how the error is particularly low at speeds between 20 and 30 km/h. They also proved that the method does not accumulate error due to the odometry. The results when using real data of an industrial area show errors between 1 and 10 cm. The method provided localization results at 10 Hz, needing times between 0.01 and 0.1 seconds to complete each cycle, with a clear peak between 0.04 and 0.06 s. As an additional note, they generate the map using a graph-based SLAM method.

In **conclusion**, the method provides more than 10 poses each second with a high accuracy. A key part of this method that could be extrapolated to other ones is how they use odometry or LiDAR momentum generate the ROIs of the map to accelerate the matching.

3 Comparison and Conclusion

In Table 1, a quantitative short comparison between all the potential alternatives is shown, choosing as the key elements the method's accuracy and the rate at which they are capable of providing localization results.

As can be appreciated, NDT-MCL-based methods (Subsection 2.1) provides the lowest error and it makes it possible to work with big grid sizes, being very versatile. However, the performance of the vanilla NDT-MCL decreases considerably when navigating in dynamic environments. The DT-NDT-MCL [12] approaches improves the performance in environments with middle-sized dynamic objects, as explained in their paper and visualized in this video. Moreover, Interest point descriptor for RObust NDT-map matching (IRON) detector and descriptor [9] can increase the speed and accuracy in matching NDT maps.

Method	Error (cm)	Rate (Hz)	Additional Notes
Vanilla AMCL (2.1)	6	?	Vulnerable to size of grid cells
NDT-MCL (2.1)	1.5 - 3	25 +	Vulnerable to dynamic environments
DT-NDT-MCL (2.1.1)	1.5 - 2.2	30 - 40	More robust for dynamic environments
IRON (2.1.3)	2	75*	More speed and accuracy in matching
Gauss. Maps (2.2)	10	5 - 10	-
L3-Net (2.3)	5	10	Learning-based, GPU required
PoseMap (2.4)	-	8	Local AND local-map sliding window matching
RT (2.5)	10	20	Best at 20 Km/h, ROI map matching

Table 1: Quantitative comparison between potential AMCL alternatives. The first rows correspond to the baseline, the second group are the NDT-MCL-based methods and the third show other alternatives. * show that this rate is only for point cloud alignment

Regarding the non-NDT-MCL-based methods, the RT localization one, exposed in Subsection 2.5, is capable of providing localization data at the highest frequency amongst the evaluated approaches. Its cost is scarifying accuracy, raising the error up to 10 cm.

In **conclusion**, we consider that, given the already theoretically good results of NDT-MCL, we should implement its vanilla version and, once it is done, consider adding further improvements such as:

- **Improve performance in dynamic environments** using beam skipping or the dual timescale proposed in DT-NDT-MCL [12].
- **Improve the matching** using the IRON detector and descriptor [9].
- **Improve the (re)sampling** using the following considerations, extracted from the SLAM course of the University of Freiburg [10]:
  - **Stochastic Universal Re-Sampling** instead of the classical *Roulette Wheel*. Lecture 11, min 00:21:00.
  - **Selectrive Re-Sampling**, to avoid killing good hypothesis for small errors. Lecture 13, min 00:50:00.
  - **Multi-Modal/Two-Step Sampling**, to enhance the multi-modal capabilities of PFs. It consist in, instead of sampling over the closes t mode to the odometry mean, sampling a Gaussian around the odometry mean and then moving each particle to its closest mode. Lecture 13, min 01:11:40.

4 Contact

This contribution was done by LI Xin and TORRES CÁMARA José M., while mentored by MACENSKI Steve.

Acronyms

AGV Automatically Guided Vehicle. 2

AMCL Adaptive Monte-Carlo Localization. 1, 2

DNN Deep Neural Network. 3

DT-NDT-MCL Dual-Timescale NDT-MCL. 2, 5, 6

EKF Extended Kalman Filter. 5

ICP Iterative Closest Point. 5

IMU Inertial Measurement Unit. 4

IRON Interest point descriptor for RObust NDT-map matching. 5, 6

K-NN K-Nearest Neighbours. 4

KF Kalman Filter. 1

KLD Kullback-Leibler Distance. 1, 2

MCL Monte-Carlo Localization. 1, 2

NDT Normal Distributions Transform. 2, 3

NDT-MCL Normal Distributions Transform MCL. 2, 3, 5, 6

PF Particle Filter. 1, 4, 6

RNN Recursive Neural Network. 4

ROI Region of Interest. 5

RT Real-Time. 4, 6

SSKF Steady-State Approximation of Extended Kalman Filter. 5

References

[1] Naoki Akai, Y. Morales, Takatsugu Hirayama, and H. Murase. Toward localization-based automated driving in highly dynamic environments: Comparison and discussion of observation models. 11 2018.

[2] Michael Bosse, Robert Zlot, and Paul Flick. Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping. *IEEE Transactions on Robotics*, 28:1104–1119, 10 2012.

[3] Philipp Egger, Paulo Borges, Gavin Catt, Andreas Pfrunder, Roland Siegwart, and Renaud Dube. Posemap: Lifelong, multi-environment 3d lidar localization. pages 3430–3437, 10 2018.

[4] Dieter Fox. Kld-sampling: Adaptive particle filters. pages 713–720, 01 2001.

[5] Navigation 2 Working Group. Amcl ros 2 package.

[6] Weixin Lu, Yao Zhou, Guowei Wan, Shenhua Hou, and Shiyu Song. L3-net: Towards learning based lidar localization for autonomous driving. 09 2019.

[7] roboticsknowledgebase.com. Adaptive monte carlo localization, Feb 2020.

[8] Jari Saarinen, Henrik Andreasson, Todor Stoyanov, and Achim Lilienthal. Normal distributions transform monte-carlo localization (ndt-mcl). pages 382–389, 11 2013.

[9] Thomas Schmiedel, Erik Einhorn, and Horst-Michael Gross. Iron: A fast interest point descriptor for robust ndt-map matching and its application to robot localization. pages 3144–3151, 09 2015.

[10] Cyrill Stachniss. Slam and robot mapping ws 2013/14 - university of freiburg, 2014.

[11] Sebastian Thrun. Particle filters in robotics. 1, 10 2002.

[12] Rafael Valencia, Jari Saarinen, Henrik Andreasson, Joan Vallvé, Juan Andrade Cetto, and Achim Lilienthal. Localization in highly dynamic environments using dual-timescale ndt-mcl. 05 2014.

[13] Miguel Valls, Hubertus Hendrikx, Victor Reijgwart, Fabio Meier, Inkyu Sa, Renaud Dube, Abel Gawel, Mathias Bürki, and Roland Siegwart. Design of an autonomous racecar: Perception, state estimation and system integration. 04 2018.

[14] Ryan Wolcott and Ryan Eustice. Robust lidar localization using multiresolution gaussian mixture maps for autonomous driving. *The International Journal of Robotics Research*, 36:027836491769656, 04 2017.

[15] Yilong Zhu, Bohuan Xue, Linwei Zheng, Huaiyang Huang, Ming Liu, and Rui Fan. Real-time, environmentally-robust 3d lidar localization. pages 1–6, 12 2019.