PML-SLAM: a solution for localization in large-scale urban environments

Zayed Alsayed, Guillaume Bresson, Fawzi Nashashibi, Anne Verroust-Blondet

PML-SLAM: a solution for localization in large-scale urban environments

Zayed Alsayed*†, Guillaume Bresson*, Fawzi Nashashibi† and Anne Verroust-Blondet†

*Institut VEDECOM
Versailles, France
firstname.name@vedecom.fr

†Inria Paris-Rocquencourt
Le Chesnay, France
firstname.name@inria.fr

***Abstract*—Localization is considered a key factor for autonomous cars. In this paper, we present a complete Simultaneous Localization And Mapping (SLAM) solution. This algorithm is based on probabilistic maximum likelihood framework using grid maps (the map is simply presented as a grid of occupancy probabilities). The solution mainly solve three renowned localization problems (1. localization in unknown environment, 2. localization in a pre-mapped environment and 3. recovering the localization of the vehicle). Memory issues caused by the open size of outdoor environment are solved using an optimized management strategy that we propose. This strategy allows us to navigate smoothly while saving and loading probabilities-grid submaps into/from a hard-disc in a transparent way. We present the results of our solution using our own experimental dataset as well as the KITTI dataset.**

I. Introduction

One of the key aspects induced by a self-driving car is the ability to localize itself in its environment. This perception task, essential to every navigation system, is crucial for autonomous driving as it is the basis on which trajectory planning and command laws will rely. This topic has therefore received a great amount of attention from the scientific community. However, it is a difficult goal to achieve as such a solution must be able to operate in large environments, provide accurate results in real time and be able to find the vehicle's location in a previously explored environment when necessary.

Simultaneous Localization And Mapping (SLAM) techniques play a central role in making cars truly autonomous. The idea behind SLAM is, for a vehicle, to be able to incrementally build a map of its surroundings while estimating its pose (position and orientation) inside this map. An interesting, but often neglected aspect is that, once a map has been built, it can then be enriched or/and used for re-localization purposes if a vehicle re-enters a previously mapped area.

In order to work in a fully autonomous car (embedding other detection and safety algorithms), the requirements of each algorithm should be optimized. This includes localization system, as well as the architecture surrounding it, they must be designed with this constraint in mind and should not limit the size of the environment in which the vehicle is moving.

We propose here a complete SLAM solution, called PML-SLAM for Probabilistic Maximum Likelihood - SLAM, based on horizontal laser sensors. PML-SLAM is well-suited to real-life applications, it's designed to perform in low resource requirements, both in terms of computational time and memory consumption management. PML-SLAM has the particularity to be designed around a solid map management strategy in order to allow navigation to be smooth while save/load/update/enrich map of any size environment. Based on this map management strategy three of the most popular problems related to localization are solved. 1. localization in unknown environment, where our solution offers the possibility to build a consistent map. 2. localization in known environment, using a pre-built map which offers the possibility to define vehicle's trajectory. 3. the kidnapped robot problem, in this case the vehicle can retrieve precisely its location and orientation after kidnapping without any priori knowledge about its new position. In these three cases the environment map can be updated in real-time even when the vehicle route from a previously explored environment or after kidnapping.

The paper is organized as follows: Section II presents the state of the art regarding SLAM algorithms. Then, Section III introduces the PML-SLAM algorithm with it's different modules that have been developed. The architecture as well as the underlying mathematical framework are presented. Finally, Section IV deals with the experiments and the results obtained.

II. State of the art

Early work on the SLAM problem led to the foundation of a probabilistic framework [1][2][3]. Even though the SLAM problem can now be considered as being theoretically solved [1], many issues have emerged in practical implementations.

The map representation has a great impact on which environments could be tackled and the performance of the algorithm. Three main map representations can be found in the literature: landmark-based maps, grid maps and raw-measurement maps. The first one relies on the environment's specific characteristics to extract significant landmarks (points, edges, etc.) [4]. One strong constraint is to be able to have a sufficiently high number of landmarks in the scene to compute a proper localization. Grid maps divide the environment into cells. Each cell is associated to an occupancy probability [5]. Consequently, these algorithms depend on the geometric structure of the environment. Last, approaches based on raw-measurement maps directly integrate raw sensor data in order to have the most accurate map possible. The direct consequence is that these maps can be difficult to store and can

require a high computational cost for processing [6].

The sensors used (information type), the environment's characteristics (information extraction) as well as the estimation process (information processing) should guide the choice of a map representation. Regarding estimation processes, the initial solutions to address the SLAM problem were based on Extented Kalman Filters [1] and Particle Filters [3][7]. Optimization methods, such as Bundle Adjustment [8], are now computationally viable and provide interesting results. These approaches are better suited to a landmark-based representation. Conversely, other solutions based on Likelihood Maximization [9][6] are suited to grid maps. Some implementations also consider a raw-measurement map in addition. Landmark maps are usually built with vision and laser sensors whereas grid maps rely mostly on lasers.

The computational performance of SLAM algorithms is also a major criterion when real-life applications are targeted. This aspect has been considered in many recent SLAM algorithms such as [10] and [11][12]. Still in the objective of making vehicles autonomous, being able to solve the kidnapped robot problem (localize the vehicle in a map given a set of observations) is important [13]. Indeed, the vehicle should be able to use a reference map when the environment has already been mapped, as it increases the relative localization accuracy and allows the map to be enriched [8].

In this paper, we propose a solution to deal with large-scale environments using a standard computer. Our solution is based on laser sensors data only which require less processing for data extraction than cameras. This choice led us to a grid-based representation due to its lightness. Around these choices, we built an estimation process based on likelihood maximization.

III. PML-SLAM

A. Probabilistic SLAM

The general formulation of the SLAM problem is the estimation of the joint probability posterior of the robot pose and the environment map simultaneously over all previous sensor observations and command inputs:

$$P(\mathbf{x}_t, \mathbf{M}_t | \mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0) \tag{1}$$

where $\mathbf{x}_t$ is the position and orientation of the robot (x, y, α) at time t, $\mathbf{M}_t$ is the environment map at time t, $\mathbf{z}_{0:t} = \{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_t\}$ is the set of all sensor observations up to time t, $\mathbf{u}_{0:t} = \{\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_t\}$ is the set of robot motion measurements up to time t and $\mathbf{x}_0$ is the initial position and orientation of the vehicle.

SLAM algorithms mainly consist of a two-step recursive process:

- *time update (prediction):* which is a prediction of the robot state and the joint map, based on previous observations as well as command inputs.

$$P(\mathbf{x}_t, \mathbf{M}_t | \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t}, \mathbf{x}_0) = \int P(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t) \times P(\mathbf{x}_{t-1}, \mathbf{M}_{t-1} | \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t-1}, \mathbf{x}_0) dx_{t-1} \tag{2}$$

- *measurement update (correction):* which is a correction of the state (vehicle and map) based on the current observation.

$$P(\mathbf{x}_t, \mathbf{M}_t | \mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0) = \frac{P(\mathbf{z}_t | \mathbf{x}_t, \mathbf{M}_t) \times P(\mathbf{x}_t, \mathbf{M}_t | \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t}, \mathbf{x}_0)}{P(\mathbf{z}_t | \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t})} \tag{3}$$

B. General Overview

PML-SLAM for Probabilistic Maximum Likelihood - SLAM, is a complete SLAM framework for autonomous cars based on information provided by laser sensors. PML-SLAM offers two different operating modes (localization with or without an a prior map of the environment). It is based on a complete framework to manage large-scale maps (there is no limitation on the environment size and the map can be gradually expanded and/or updated in real-time).

The general flowchart of the PML-SLAM approach is given in Figure 1. It shows the architecture of the implementation which is basically built around three main blocks (dark green boxes).

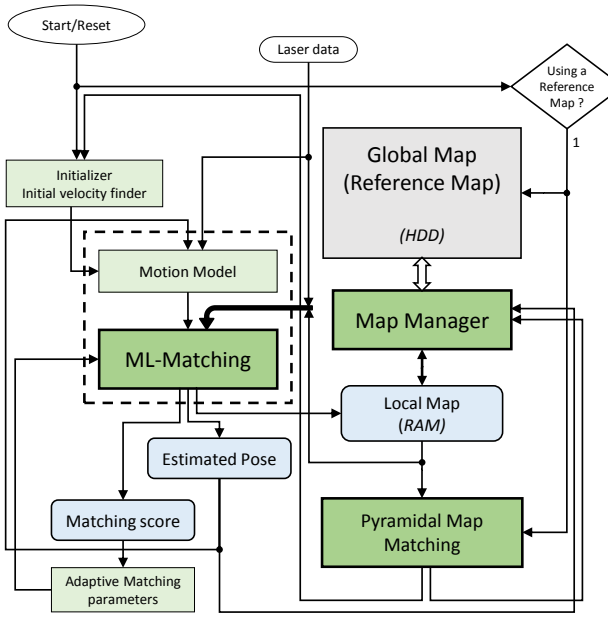


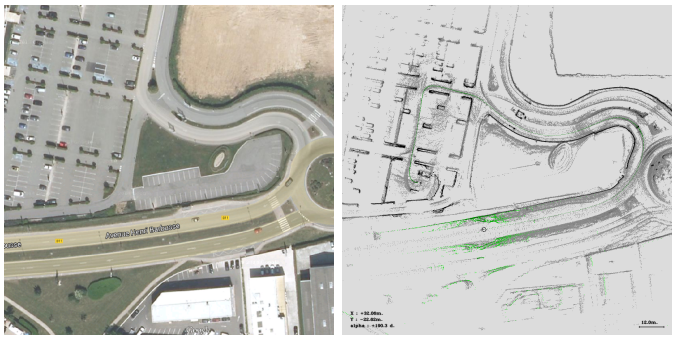
Fig. 1. General flowchart of the PML-SLAM algorithm

Different module illustrated on this flowchart will be detailed in the rest of this section.

C. Overview on Data Representation

The map is expressed through a grid of probability cells with a parameterizable resolution, which makes it possible to control the discretization level of the environment. We also take advantage of this discretization to maintain maps at different resolutions to perform localization recovery faster (cf.

III-H). The cell size, in the highest resolution, is set according to the laser sensor characteristics to obtain the best localization accuracy possible. An example of such a map is shown in Figure 2.



(a) Satellite map (b) Map built by PML-SLAM

Fig. 2. Map from Plaisir-France

The map, at resolution level r and at time t, is denoted as $\mathbf{M}_{t,L=r}$. It is the fusion of all submaps $\{\mathbf{M}^1, \mathbf{M}^2, \cdots \mathbf{M}^t\}_{L=r}$, where the submap $\mathbf{M}^t_{L=r}$ is the environment seen at time t. The different resolution level are used to solve kidnapped robot problems (more details in Section III-H).

$\mathbf{M}^t_{L=r}$ is the collection of cells $\{m^1, m^2, \cdots m^N\}$ with N being the maximum number of cells in the submap. Each cell corresponds to an occupancy probability $[0, 1]$ where a cell occupancy probability close to 0 means that it is likely to be free. Conversely, a probability close to 1 means that a cell is likely to be occupied.

Figure 2(b) shows the probability grid map of the real environment presented in Figure 2(a). The dark cells are those with a high occupancy probability and the light cells represent the ones that are likely to be free. The green points represent the current observation data.

D. Motion Model

The Motion Model computes the pose prediction based on a vehicle model. It takes into account the vehicle's previous pose $\mathbf{x}_{t-1}$ and the vehicle's speed $\mathbf{v}_{t-1}$ (linear and angular) to provide a list of position and orientation candidates $\mathbf{X}_t = \{\mathbf{x}_{1,t}, \mathbf{x}_{2,t}, \cdots \mathbf{x}_{n,t}\}$ where the vehicle can be located.

Here, we are using a *Constant Velocity* model to describe the behavior of the vehicle's motion. Indeed, as the time interval δt between two consecutive laser scans is small ($\approx$ 60 milliseconds), we assume that the variation in the vehicle's speed $\delta \mathbf{v}_t$ is also small. Our prediction model is expressed as:

$$\mathbf{\Delta x}_t = \mathbf{\Delta x}_{t-1} + \delta t \times \delta \mathbf{v}_t \tag{4}$$

$$\mathbf{x}_{t|t-1} = \mathbf{x}_{t-1|t-1} + \mathbf{\Delta x}_t \tag{5}$$

E. Initial velocity finder

The role of the Initial velocity finder module is to deal with scenarios which begin when the vehicle is already moving. With the Constant Velocity model presented previously, these cases can lead to wrong estimates at the beginning of such a trajectory.

The module computes a rough estimation of the movement by minimizing the distance (translation and rotation) between the two first laser scans $\mathbf{z}_0 = \{\mathbf{z}^0_0, \mathbf{z}^1_0 \cdots \mathbf{z}^{n-1}_0\}$ and $\mathbf{z}_1 = \{\mathbf{z}^0_1, \mathbf{z}^1_1 \cdots \mathbf{z}^{n-1}_1\}$ where $\mathbf{z}^i_j$ is the i^{th} 2D point in the j^{th} laser scan composed of n points.

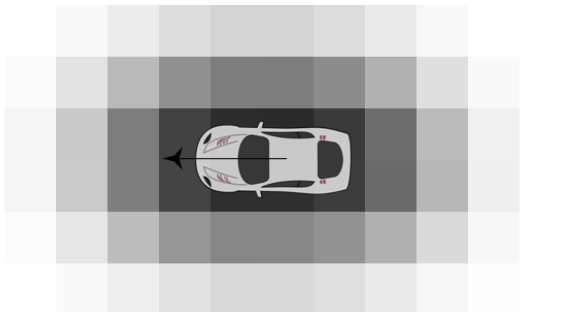


Fig. 3. Search area around the vehicle to compute its movement.

This first estimate is estimated as follows ($dist$ is the function computing the Euclidean distance between two points):

$$\mathbf{\Delta x}_1 = \frac{1}{n} \sum_{i=0}^{n-1} dist(\mathbf{z}^i_0, \mathbf{z}^i_1) \tag{6}$$

$$\mathbf{v}_1 = \mathbf{\Delta x}_1 / \delta t_{(0,1)} \tag{7}$$

The values computed $(\mathbf{\Delta x}_1, \mathbf{v}_1)$ can then be transmitted to the Motion Module presented above.

The search space is limited by the maximum distance the vehicle can travel during time between the first two observations $(\mathbf{z}_0, \mathbf{z}_1)$. Figure 3 shows the distribution of candidates all around to cover the possible area. Each square represents one candidate on a low resolution map to accelerate the search process.

F. Maximum Likelihood Matching

The Maximum Likelihood Matching module corresponds to the correction process in SLAM algorithms (see Equation 3)). It takes into account the new measurement of the environment $\mathbf{z}_t$ in order to refine the state estimation $\mathbf{x}_{t|t-1}$ proposed in the output of the Motion Model.

This module performs a maximization of similarity between the current laser scan $\mathbf{z}_t$ and the map $\mathbf{M}_{t-1}$ from the previous time using the list of candidate poses $\mathbf{X}_t$ provided by the motion model:

$$\begin{aligned} \mathbf{x}_{t|t} = arg\ max\ \{&P(\mathbf{z}_t|\mathbf{x}_{t|t-1}, \mathbf{M}_{t-1}) \\ &\times P(\mathbf{x}_{t|t-1}|\mathbf{x}_{t-1|t-1}, \mathbf{u}_t)\} \end{aligned} \tag{8}$$

$$P(\mathbf{z}_t|\mathbf{x}_{t|t-1}, \mathbf{M}_{t-1}) \propto \sum_{cell}^{N} \sum_{i=1} P(m^i_{t-1}) \tag{9}$$

where m^i_{t-1} is occupied.

In other words, this step consists in finding the best position and orientation $\mathbf{x}_{t|t}$ of the vehicle by fitting the observation $\mathbf{z}_t$ on the map $\mathbf{M}_{t-1}$.

After performing the correction step, we update the system joint state $(\mathbf{x}_t, \mathbf{M}_t)$ by merging the new observation in the map:

$$\mathbf{M}_t = \mathbf{M}_{t-1} \cup \mathbf{M}^t \tag{10}$$

This module is completed by a feedback process to adapt the matching parameters depending on the matching score. The threshold for cell selection (which is based on their probability) is adjusted to maximize the number of matched cells in the next SLAM update. If the matching score is too low, the feedback triggers a reset of the system in order to re-localize the vehicle.

G. Map Manager

This map manager is the key feature of our work. The map is represented by a high resolution occupancy grid and we aim at performing SLAM in large-scale environments. This is not compatible as maps grow in size and we have limited memory resources. The Map Manager module provides a robust management of our limited resources. The goal is to be able to have high resolution maps for accurate matching without putting any limitation on the environment size.

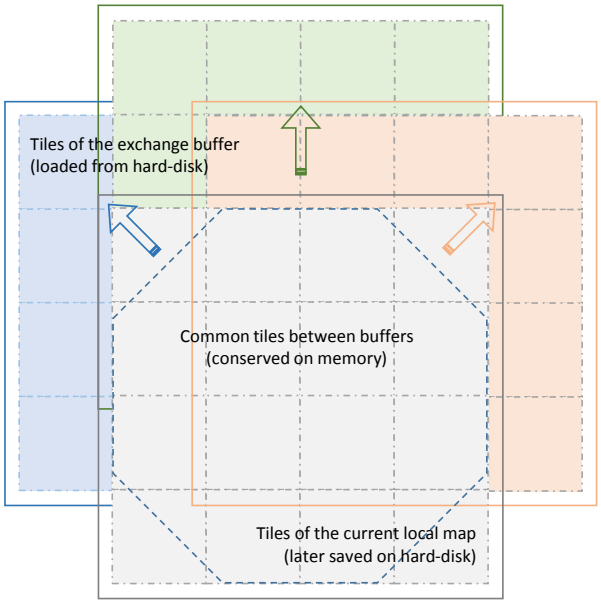


Fig. 4. Map buffers to deal with large-scale environments

Our technique consists in managing the map between the limited Random Access Memory (RAM) and the hard-disk-drive (HDD) where space is not an issue. First, the map is divided into small areas which we call tiles. Then, we only keep in the memory a current local part of the global map which is still big enough to navigate seamlessly from one local map to another. The other tiles are saved on the hard-disk. Tiles are then loaded and unloaded depending on the pose of the vehicle so as always being able to match current observations. An illustration of this map management strategy is given in Figure 4. The grey square is the current local map buffer, containing a grid of 16 tiles, which is actually loaded in the RAM. The other squares (blue, green and red) are candidate configurations to which the local buffer may next move, depending on the vehicle's direction. The arrows of same color shows the next configuration to be adopted.

Furthermore, the key ingredient behind exchanging buffers seamlessly between RAM and HDD in real-time is by compressing probability grid submaps. The buffer (grid of probability) is seen as an image buffer. This allows us to use a image compression techniques. Here we are using a compression technique in PNG which accelerates files saving and loading operations as well as reducing the space required on the hard-drive.

H. Pyramidal Map Matching

The Pyramidal Map Matching module is able to solve the kidnapped robot and relocalization in pre-mapped environments (with what is called a reference map) problems. The idea is to perform an extensive search of the current observation over the reference map to find the robot's pose with relation to this map. As this process can be unreliable and time consuming, the reference map is expressed at different resolution levels. At low resolutions, finding a correspondence is fast but not accurate. By going through the different levels, we are able to refine the pose quickly, by testing only a few pose candidates.

This module uses a similar strategy than proposed by Xie et al. [9]. The main difference comes from the map representation. Instead of a raw-measurement map, we take advantage of our Map Manager module to directly use a grid representation. It has the advantage to allow a faster and constant loading time compared to [9] where the time will vary according to the map size.

As the search can be long without prior knowledge (GPS position for instance), the vehicle continues to perform SLAM while the Pyramid Map Matching module works as a background task. Once the pose on the reference map is found, the current state is brought back into the frame of the reference map.

IV. Experiments

The first experiments were carried out using the KITTI odometry database [14] which provides datasets (with Velodyne, cameras and a ground truth) in various environments for performance evaluation. In order to adapt the data to our implementation inputs, we cut the original Velodyne data to simulate an ordinary single-layer LiDAR scanner configuration (360° field of view with a 0.25° resolution).

In order to test the relocalization performance of PML-SLAM, we also built our own datasets thanks to a vehicle equipped with 5 laser scanners covering a 360° field of view with a 0.25° resolution. Only one layer was used to obtain the results presented in this article. An IMU was fused with a RTK-GPS to provide a ground truth.

All the experiments took place in urban and peri-urban environments in real conditions (moving obstacles were present). All tests were performed on a computer equipped with a Core-i7 running at 2.9 GHz. For all the results presented below, PML-SLAM took on average 2 ms to process a scan layer.

A. SLAM in unknown environments

In this experiment, we used the sequence "05" from the KITTI odometry dataset, collected in a residential environment. We performed SLAM without any prior knowledge on the environment.

At the beginning of the trajectory, a map of 16 tiles (resolution $10\ pixels/meter$) were initialized. During the whole trajectory, a total of 32 tiles were created and saved on the hard disk, covering a $720,000\ m^2$ surface. The map produced takes up 981 KB on the hard disk. The computed trajectory is shown in Figure 5.

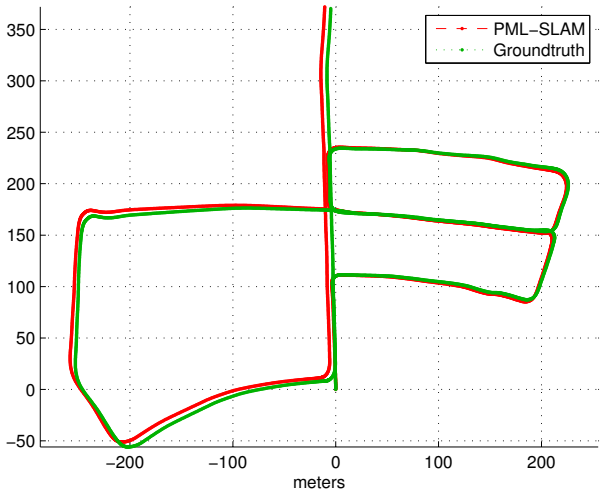


Fig. 5. Vehicle paths from the Kitti odometry dataset, the path estimated by PML-SLAM and the ground truth path.

The errors in displacement and heading at each iteration are illustrated in the Figure 6. As we can see, the errors in displacement are bounded by $\pm\ 5\ cm$, while the errors in heading are limited to $\pm\ 0.3°$.

Path estimated by SLAM is close to the ground truth. The deviation throughout the trajectory in distance and in heading are shown in Figure 7. We can see that after $2,204$ meters the deviation in distance is about 6 meters with 0.1°of deviation in heading.

B. Recovering from kidnapping

In this experiment we used a dataset which we collected in a city using our platforms. The aim of this test is to illustrate

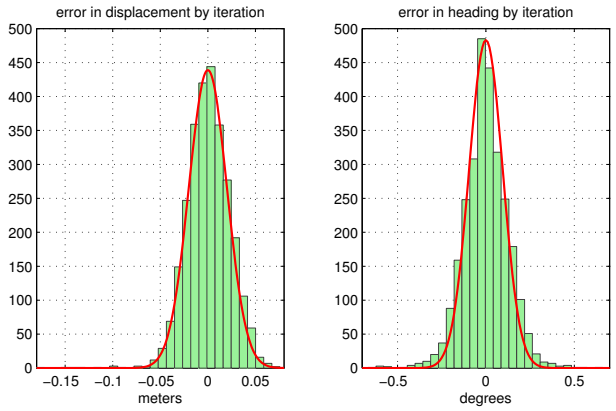


Fig. 6. Errors in heading and displacement by iteration during the test.

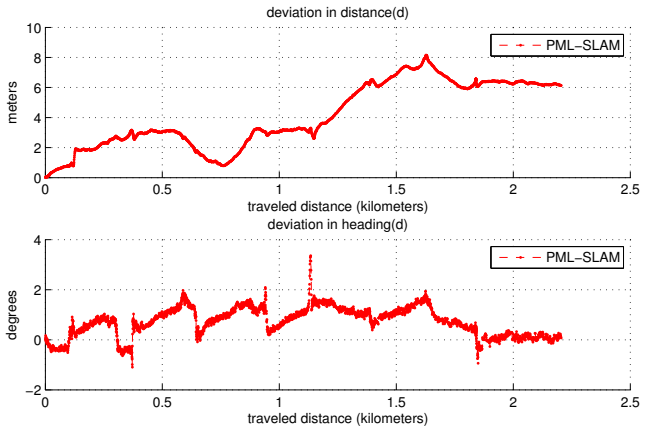


Fig. 7. Deviation from real path in distance and heading along journey.

the kidnapped robot problem and how we deal with it. The trajectory performed and the recovery from kidnapping are shown in Figure 8.

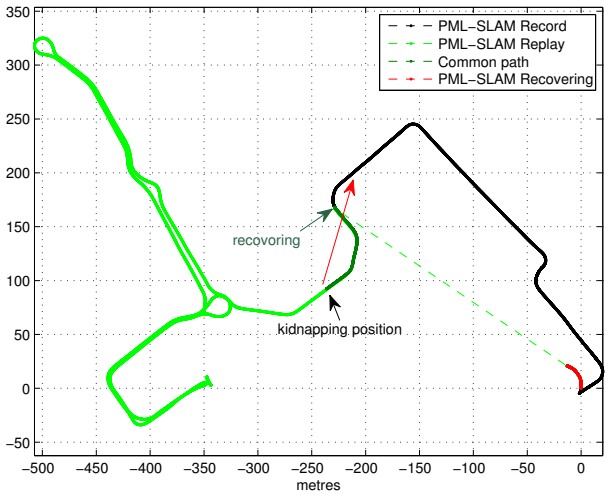


Fig. 8. Vehicle path estimated by PML-SLAM during experiment in mapping with kidnapping and recovering from kidnapping.

The vehicle starts to build a map (black path), then the

vehicle is kidnapped and put in another location (diamond on the trajectory). As a consequence, the vehicle starts a new local SLAM process (red path) whose computed trajectory takes a similar shape to the previous route. During this time, the position is provided in a local reference. The matching process then finds the vehicle new position on the pre-built map (recover from kidnapping). The recovery moment is represented by the dashed green lines. The vehicle's positions provided from now on are given on the global map. The map is further enriched during the common path (in dark green). Once the vehicle enters areas not previously mapped (light green), the vehicle continues to map and localize itself.

The map built at the beginning of the experiment served as a recovering-from-kidnapping map, and covered $360,000\ m^2$. The localization precision after recovering was $\pm\ 5\ cm$.

The operation of recovering took about 27 seconds in this example. However, this time depends on the random selection of matching candidates and is thus variable. Nevertheless, while searching for its position, the vehicle is still localizing itself with the SLAM process. The final map built takes up 508 KB on the hard disk and it should cover $517,500\ m^2$ of surface.

C. Memory consumption

The limitation in resources is an important issue treated in this paper. Figure 9 shows a comparison in terms of memory consumption regarding the maximal size of the area covered by the SLAM system with and without the Map Manager module presented in this article (see Section III-G).

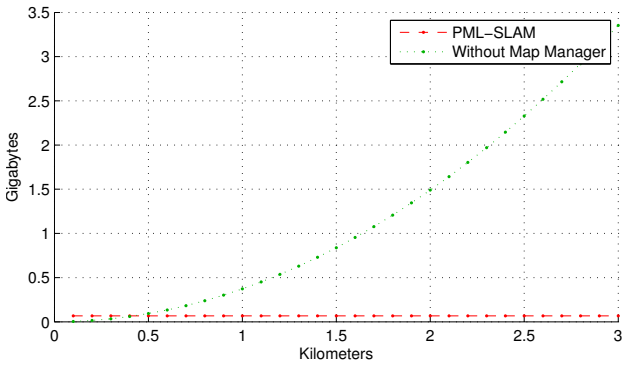


Fig. 9. Memory consumption regarding the area covered by SLAM

We can see on the graphic that memory required by a SLAM system without Map Management increases quadratically in function of the environment size. In contrast the memory required by a SLAM with a Map Manager is constant regardless the environment size. In realistic scenarios of autonomous driving we can't predefine the map size, a vehicle need to have it's freedom to navigate.

V. Conclusion

We have presented a complete SLAM algorithm, called PML-SLAM. The proposed solution is based on the probabilistic maximum likelihood framework coupled with a grid representation. Its integration inside a complete architecture allows to use any previously built map if available. A map matching algorithm, based on a pyramidal search with multi-resolution map, is able to re-localize the vehicle inside a map given an observation, thus solving the kidnapped robot case. Our approach has been designed with real-life applications in mind and is consequently light both in terms of memory requirements and processing time. The map is smartly managed to be loaded and unloaded on the hard drive when needed thus allowing to work in large-scale environments.

We have validated PML-SLAM with several experiments over different scenarios: SLAM localization, relocalization and navigation inside an existing map. The results show that our approach is viable for real-time urban localization without restrictions on the size of the environment.

In future work, we plan to fuse several localization algorithms within a supervision layer in order to increase robustness and localization accuracy.

References

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous Localization and Mapping: Part I," *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[2] T. Bailey and H. Durrant-Whyte, "Simultaneous Localization and Mapping (SLAM): Part II," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[3] S. Thrun, W. Burgard, and D. Fox, "A Real-Time Algorithm for Mobile Robot Mapping With Applications to Multi-Robot and 3D Mapping," in *IEEE International Conference on Robotics and Automation*, vol. 1, 2000, pp. 321–328.

[4] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[5] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[6] T.-D. Vu, "Vehicle Perception: Localization, Mapping with Detection, Classification and Tracking of Moving Objects," Ph.D. dissertation, Inria Grenoble Rhône-Alpes - LIG Laboratoire d'informatique de Grenoble, 2009.

[7] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization And Mapping Problem," in *AAAI/IAAI*, 2002, pp. 593–598.

[8] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Real Time Localization and 3D Reconstruction," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 363–370.

[9] J. Xie, F. Nashashibi, M. Parent, and O. G. Favrot, "A real-time robust global localization for autonomous mobile robots in large environments," in *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*. IEEE, 2010, pp. 1397–1402.

[10] O. E. Hamzaoui, "A fast scan matching for grid-based laser SLAM using streaming SIMD extensions," in *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*, 2010, pp. 1986–1990.

[11] ——, "tinySLAM: A SLAM algorithm in less than 200 lines C-language program," in *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*, 2010, pp. 1975–1979.

[12] J. Levinson, M. Montemerlo, and S. Thrun, "Map-Based Precision Vehicle Localization in Urban Environments," in *Proc. of Robot. Sci. Syst.*, vol. 4. Citeseer, 2007, p. 1.

[13] S. Se, D. Low, and J. Little, "Global Localization using distinctive visual features," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002, pp. 226–231.

[14] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets Robotics: The KITTI Dataset," *International Journal of Robotics Research (IJRR)*, 2013.