

A Lightweight SLAM Algorithm for Indoor Autonomous Navigation

Paolo Tripicchio, Matteo Unetti, Nicola Giordani, Carlo A. Avizzano and Massimo Satler*

Abstract

Simultaneous Localization and Mapping (SLAM) algorithms require huge computational power. Most of the state-of-the-art implementations employ dedicated computational machines which in most cases are off-board the robotic platform. In addition, as soon as the environment become large, the update rate of such algorithms is no more suitable for real-time control. The latest implementations rely on visual SLAM, adopting a reduced number of features. However, these methods are not employable in environments with low visibility or that are completely dark. We present here a SLAM algorithm designed for mobile robots requiring reliable solutions even in harsh working conditions where the presence of dust and darkness could compromise the visibility conditions. The algorithm has been optimized for embedded CPUs commonly employed in light-weight robotic platforms. In this paper the proposed algorithm is introduced and its feasibility as SLAM solution for embedded systems is proved both by a simulated and a real testing scenario.

1 Introduction

The most important problem in the mobile robotics field is to determine the robot pose and localize the robotic platform in the environment, thus answering the question: "Where am I?".

From the early research in this field many different algorithms have been proposed to solve the localization issue and the mapping of the working environment. The simplest solution to the localization problem is to use an

odometric algorithm to estimate the current position of the robot. Starting from this simple approach, several works have been proposed in the literature that adopt state observers and Kalman Filtering to correct the drifting error introduced by the odometry integration. These methods introduce a measurement phase which generally exploits more accurate measurement than the odometry even if the update rate of this phase is slower. A common solution used in the literature adopts the laser range finder performing a scan match [1] and, in order to reduce the computational cost, it is possible to choose a proper set of features to be matched instead of the entire raw laser scan [2].

However, to plan a trajectory or the sequence of actions to accomplish a given mission, the knowledge of the robot motion is not sufficient. For high level behavior algorithm it is required to know the position of the robot within a map or in cases it is not known a priori, it is very useful to build the map while the robot is exploring the environment. For this reason during the years a large number of researchers have studied the problem of Simultaneous Localization and Mapping (SLAM). Most of the works have been developed over the contribution of Smith et al. [3] that uses an Extended Kalman Filter (EKF) to solve the problem. EKF covariance matrices, however, are quadratic with respect to the map size and in turn, the update time is quadratic with respect to the features of the map. This raises an issue when the SLAM algorithm has to be run in real-time on the embedded robot's electronics. Actually, in many cases, it is not possible to use an external computing device to perform computational expensive algorithms. This is the case of teleoperated or autonomous robots exploring unknown in-door environments. These kind of robots have the need to embed the localization algorithms directly on-board to be able to cope with communication black-outs and in the same time maintaining a stable and secure behavior in any circumstances.

Latest State of the Art SLAM methods targeting to low computational power platforms are based on fusion

*Gustavo Stefanini Advanced Robotics Research Center, PERCeptual RObotics laboratory, TECIP Institute, Sant'Anna Superior School of Advanced Studies. E-mails: p.tripicchio@sssup.it, m.satler@sssup.it

algorithms which exploit monocular vision and IMU devices [4], [5]. Although these methods proved to be suitable to localize mobile robots both in outdoor and in indoor environments like an office facility, they cannot be employed in dark environments. Recently there is a growing interest in the development of robotic solutions to carry out high-risk service tasks, such as the inspection of power plants and/or power lines. Many times such tasks have to be performed in challenging conditions due to the presence of dust and darkness that strongly limit the visibility of the camera sensors. In addition, industrial settings may not allow the use of GPS information and the magnetic compass generally contained in standard IMUs are not always reliable because of the presence of metallic parts which generate unusual and unpredictable electromagnetic disturbances.

This work presents a custom SLAM algorithm optimized for embedded computing that employs a laser range scanner and an IMU within the sensor fusion algorithm. The proposed SLAM algorithm is able to work both in outdoor and indoor environment even in the presence of sudden light changes and/or darkness. This provides a reliable solution for robots that must operate in unstructured and challenging environments like for instance gas boilers and industrial plants.

Within the paper, the algorithm is described and its capability is shown both in a simulation environment and in a real test scenario. Our method starts from the second version of the FastSLAM algorithm with the aim to improve computational time in order to be exploited in embedded processor with reduced computational power (compared to standard workstation). The second version of the FastSLAM algorithm has been proposed by Montemerlo et al. as an evolution of the FastSLAM algorithm [6], based on particle filtering [7], in order to improve the computation time of classical EKF implementations.

The proposed SLAM algorithm has been implemented and tested in a Micro Aerial Vehicle (MAV) employed for semi-autonomous in-door exploration. The application of the proposed method in a real-life scenario for confined space inspection is presented in [8]. In particular the designed system is composed by a flying vehicle and a control ground station consisting of an architecture that allows an operator to remotely supervise an aerial service robot while performing a visual inspection of confined spaces in a power plant of an industrial gas burner. Basically, the human operator does not need to be a skilled pilot to operate the aerial robot, but an expert in the required service task.

The remaining part of this work is organized as follow. Section 2 introduces the SLAM problem and defines the formalism used in the literature and in the paper as well. In Section 3 the FastSLAM 2.0 algorithm is briefly pre-

sented as the starting point of our method. Section 4 introduces the proposed algorithm whereas Section 5 presents two different test sessions to compare the result of the proposed method with the original implementation of FastSLAM 2.0. The paper end with Section 6 where conclusions are drawn.

2 Simultaneous Localization and Mapping

The problem solved by SLAM algorithms is to determine a good estimation of the vehicle pose and, at the same time, to represent the operating environment, where the vehicle lays, as a map. The map is usually represented as a set of n landmarks or features that can be denoted as $F = f_1, \dots, f_n$. The trajectory of the vehicle in the space can be denoted as $S_t = s_1, \dots, s_t$ where t is an index representing the time and s_t is the vehicle pose at the discrete time interval t .

Given this assumption the SLAM algorithms available in the literature compete to obtain a posterior distribution that can be expressed as follow:

$$p(F, S_t | Z_t, U_t, D_t) \quad (1)$$

where $Z_t = z_1, \dots, z_t$ is a sequence of measures (e.g. ranges from laser, radar), and $U_t = u_1, \dots, u_t$ is the control sequence of the vehicle (e.g. imposed wheel/motor velocities). Usually these methods assume that a single feature is observed at each time t so that $D_t = d_1, \dots, d_t$ represents the data association variables and d_t specifies the feature or landmark that has been observed at time t . In the general case this association can be known or unknown.

To calculate the probability in (1), the algorithms have to compute two probability distributions, one for the odometry or motion model and the other for the measurement model. The motion probability distribution describes how the control command u_t affects the vehicle pose, the measurement model instead describes how the measurements evolve from a given state. Hence the need to solve the system of equations

$$p(s_t | u_t, s_{t-1}) \quad (2)$$

$$p(z_t | s_t, F, d_t) \quad (3)$$

These equations are usually chosen in literature as nonlinear functions with independent Gaussian noises.

In the following section the FastSLAM 2.0 algorithm will be introduced as it has been used as the starting point from which the proposed algorithm develops.

3 FastSLAM 2.0

FastSLAM 2.0 [9] starts from the observation [10] that the posterior (1) can be factored being the features or

landmarks estimable independently. So basically the probability in (1) can be split in one term that estimates the probability of a vehicle trajectory and N terms estimating the positions of each features conditioned to the trajectory :

$$p(S_t | Z_t, U_t, D_t) \prod_n p(f_n | S_t, Z_t, U_t, D_t) \quad (4)$$

The trajectory is sampled by means of a particle filter. For each particle a map is created, composed by N extended Kalman filters (EKF). The particle i is then described by the trajectory S^i and N estimates of the features expressed as mean and covariance pairs $(\mu_{n,t}^i, \Sigma_{n,t}^i)$.

The fastSLAM 2.0 algorithm starts sampling the vehicle pose based on both the motion command u_t and the measurement z_t , in other words, for each particle, it computes the probability

$$p(s_t | S_{t-1}^i, U_t, Z_t, N_t) \quad (5)$$

The next step is to update the estimate of the observed features computing the posterior

$$p(f_n | S_t^i, N_t, Z_t) \quad (6)$$

This update results equivalent to the measurement update equation of the EKF [11].

At this point a sampling phase is needed to weight each particle in order to match the desired posterior, this step is usually known as importance sampling. The probability for the i -th particle to be sampled is given by

$$p(z_t | S_{t-1}^i, U_t, Z_{t-1}, D_t) \quad (7)$$

FastSLAM 2.0 can be used also in the case of unknown data association, in this case each particle makes its own local association (\hat{d}_t) maximizing the data association likelihood:

$$\hat{d}_t^i = \operatorname{argmax}_{n_t} p(z_t | d_t, \hat{D}_{t-1}^i, S_t^i, Z_{t-1}, U_t) \quad (8)$$

In the case of unknown data association there is the need to create the features of the map dynamically. The usual approach to create new features is to add them when the data association probability \hat{d}_t^i is below a certain threshold. However, to remove spurious data occurring in real-world acquisitions, it is possible to employ a Bayes filter similar to the one used on occupancy grid map literature [12].

FastSLAM 2.0 results more robust to noise with respect to EKF algorithms [13] for the fact that each particle makes a local data association opposed to having a single data association for the whole filter.

The presented algorithm (named SlamCGS) starts from the results of FastSLAM 2.0 that proved to be efficient and robust trying to decrease the computing power

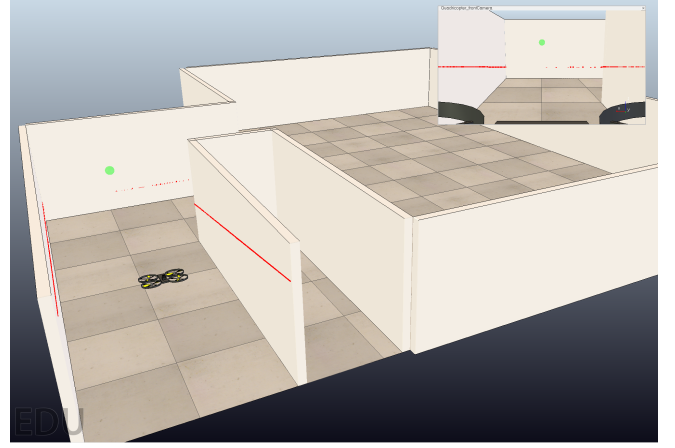


Figure 1: V-REP simulation to test the proposed algorithm against the FastSLAM 2.0 implementation

needed for and the precision of the estimation. This is requested in many applications where the SLAM algorithm is used in the control loop to maintain vehicle stability and the computing power is reduced because of the embedded processors that are used in the robotic platforms. Such computing units usually are also responsible for streaming video data and the overall communication with a remote supervising unit thus reducing the available computing power to run the control loop itself.

The main idea used to extend the FastSLAM 2.0 algorithm comes from the literature on EKF localization with unknown association. In section 2 we anticipated one of the main assumption used by many SLAM algorithms that is to assume that in each loop only one measurement is obtained and the algorithm is cycled many times with a single measurement as input. In the proposed algorithm instead in each loop all the measures are taken into consideration to obtain a posterior probability. In this way the proposed algorithm results faster and more precise and this result will be shown in the following sections.

4 SlamCGS

In this section the proposed algorithm will be detailed to obtain a clearer view of its functioning.

To obtain a closed form from the sampling of the distributions there is the need to approximate the probability $p(z_t | s_t, F, d_t)$ with a linear function plus Gaussian noises.

In the following we will approximate in the algorithm this measurement probability with a linear function $h(s_t, f_{nt})$ plus a Gaussian noise with zero mean and covariance Q_t and the probabilistic motion model $p(s_t | u_t, s_{t-1})$ as a nonlinear function $g(u_t, s_{t-1})$ plus a Gaussian noise with zero mean and covariance R_t . Hence the overall algorithm can be expressed as in Algorithm 1.

```

for each particle do
    Predict the pose;
    Assign a Covariance to the pose;
    Calculate the Data Associations;
    Update the pose;
    Sample the pose;
    Update the Features;
    Discard the Dubious features;
end
Resample the particles;
    
```

Algorithm 1: One step of the SlamCGS

We consider the i -th particle composed by the spatial trajectory (s_t^i) and the full set of N_t features of the map that are described by a mean ($\mu_{n,t}^i$), a covariance ($\Sigma_{n,t}^i$) and a corresponding visibility counter ($v_{n,t}^i$). The visibility counter is used to discard outliers and dubious features. Each loop of the algorithm starts taking a particle described by:

$$s_{t-1}^i, N_{t-1}^i, \{\mu_{1,t-1}^i, \Sigma_{1,t-1}^i, v_{1,t-1}^i; \dots; \mu_{N,t-1}^i, \Sigma_{N,t-1}^i, v_{N,t-1}^i\} \quad (9)$$

Then we predict the pose of the vehicle calculating $\hat{\mu}_t^i = g(s_{t-1}^i, u_t)$ and the covariance of the pose $\hat{\Sigma}_t^i$ is set equal to the covariance R_t of the Gaussian noise. u_t represents the control command given to the vehicle.

Now for each of the M measures we have to calculate the data association, this is done taking the measure z_t^k with $k = 1 : M$ and looping through each possible features in the map in the range $j \in [1, N_{t-1}]$.

We predict the measurement $\bar{z}_j = h(\mu_{j,t-1}^i, \hat{\mu}_t^i)$ and obtain the measurement covariance matrix Q_j calculating the Jacobian with respect to the map feature $H_{F,j}$.

$$H_{F,j} = \nabla_{F_j} h(\mu_{j,t-1}^i, \hat{\mu}_t^i) \quad (10)$$

$$Q_j = Q_t + H_{F,j} \Sigma_{j,t-1}^i H_{F,j}^T \quad (11)$$

Once obtained the measurement information it is possible to calculate the correspondence likelihood as

$$\pi_j^k = |2\pi Q_j|^{-1/2} \exp(-\frac{1}{2}(z_t^k - \bar{z}_j)^T Q_j^{-1} (z_t^k - \bar{z}_j)) \quad (12)$$

Once obtained the correspondences of the measure z_t^k with all the features in F^i , we compute the maximum likelihood correspondence $c^k = \argmax \pi_j^k$. If the result is lower than a certain threshold the z_t^k measure corresponds to a new feature that should be added to the map.

To update the pose for each measure z_t^k we predict the measurement with the formula $\bar{z} = h(\mu_{c,t-1}^i, \hat{\mu}_t^i)$. We

then calculate the Jacobians of h with respect to the pose x and the map feature m :

$$H_s = \nabla_s h(\mu_{c,t-1}^i, \hat{\mu}_t^i) \quad (13)$$

$$H_F = \nabla_F h(\mu_{c,t-1}^i, \hat{\mu}_t^i) \quad (14)$$

Under this EKF-style approximation, the proposal distribution is Gaussian with the following parameters:

$$\hat{\Sigma}_t^i = [H_s^T Q^{-1} H_s + (\hat{\Sigma}_t^i)^{-1}]^{-1} \quad (15)$$

$$\hat{\mu}_t^i = \hat{\Sigma}_t^i H_s^T Q^{-1} (z_t^k - \bar{z}) + \hat{\mu}_t^i \quad (16)$$

Where the measurement covariance matrix Q is defined as:

$$Q = Q_t + H_F \Sigma_{c,t-1}^i H_F^T \quad (17)$$

The pose s_t^i is then sampled from the normal distribution $N(\hat{\mu}_t^i, \hat{\Sigma}_t^i)$.

For the measurement update, for each measure z_t^k , we first produce a measurement prediction $\hat{z} = h(\mu_{c,t}^i, s_t^i)$, then we calculate the Jacobians with respect to the map features (H_F) and the pose (H_s) and obtain the measurement information Q_c . The update equation then becomes similar to the standard EKF measurement update [11]:

$$K = \Sigma_{c,t-1}^i H_F^T Q_c^{-1} \quad (18)$$

$$\mu_{c,t}^i = \mu_{c,t-1}^i + K(z_t^k - \hat{z}) \quad (19)$$

$$\Sigma_{c,t}^i = (I - K H_F) \Sigma_{c,t-1}^i \quad (20)$$

The visibility counter $v_{c,t}^i$ is incremented by a fixed quantity. The importance weight is computed with the following two steps:

$$L = H_s \hat{\Sigma}_t^i H_s^T + H_F \Sigma_{c,t-1}^i H_F^T + Q_t \quad (21)$$

$$w^i = w^i * (|2\pi L|^{-1/2} \exp(-\frac{1}{2}(z_t^k - \hat{z})^T L^{-1} (z_t^k - \hat{z}))) \quad (22)$$

At this point we obtained the weights for the re-sampling phase. However, before performing the re-sampling, we try to filter out measurement outliers and dubious features. This is done taking into account the presence and absence of features in the measurement. Observing a feature provides positive evidence for its existence, whereas not observing it when s_t falls within the vehicle's perceptual range provides negative evidence. A feature that does not corresponds to any map feature is considered at first as a new feature, in the following cycles becomes a dubious feature and it is eliminated.

For such a reason at each loop the visibility counters of the visible particles are decremented. When this counter drops below a predefined threshold, the corresponding feature is removed from the map. With this mechanism the particles free themselves of spurious features.

5 Methods Comparison

In this section the proposed method is compared to the original fastSLAM 2.0 implementation in a specific scenario. The scenario is the one of a UAV flying in an indoor environment for inspection purposes. Hence, the UAV embedded computing unit should stream the video captured by the on-board camera and in the same time should be responsible for the navigation control system as well as the SLAM computation. Given the reduced computing power, a testing session has been organized to test the computational efforts required by both methods as well as to verify the quality of the mapping. We present here a testing session composed of two different setups. The first setup (sub-section 5.1) has been implemented as a simulation inside the robot simulator V-REP [15], whereas the second setup (sub-section 5.2) is represented by a real scenario. In both setups, the chosen indoor location is represented by an office environment which is mainly characterized by straight walls. In particular, during the experimental trials, the platform flew inside the meeting room in our research facility.

Given the chosen setting, a good choice as map features for the estimation of the walls locations is represented by the Hessian coordinates. This requires that the sensors of the robot is able to measure distances and angular displacements of walls with respect to the robot frame. We have chosen to use walls only as natural landmark, because (a) their extraction from sensor data is proved to be very reliable with respect to other features, (b) they are less prone to occlusion than other features and moreover (c) they are commonly present in all indoor environment. It has to be point out that, this choice can be a limitation if the environments has no straight walls. However, other features, like corners and edges could be added without great modification to the algorithm itself.

Let then F be a set of n line features F_p ,

$$F = \{F_p, p = 0, \dots, n-1\} \quad (23)$$

where each F_p is represented in Hessian form with respect to the map coordinate frame O_{xy} .

The Hessian representation [16] of a wall with respect to a reference frame O_{xy} (see Fig. 2) is given by the triplet:

$$(r, \theta, v) \quad (24)$$

where:

- $r = |OP^*|$ is the distance from O to the closest point P of the wall;
- $\theta = \theta_{cc}(i, n)$ is the angle, defined over the interval $[0, 2\pi)$, by which the unit vector i of the x-axis has to rotate counter-clockwise to overlap on the outward normal n to the wall;

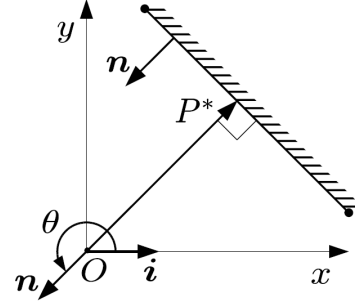


Figure 2: diagram of the hessian coordinates used to describe a wall feature.

- $v = -\frac{OP^* \cdot n}{|OP^*|}$ is the position parameter: its value is 1 when O is in front of the wall and -1 when O is behind the wall.

The Hessian form contains no information about length and position of end points of the wall in the map. However, these information can seldom be used because the detection of end points in cluttered environments is difficult.

A line extraction algorithm based on a least square error technique [17] is applied to the set of range readings from a laser range finder mounted on the robot and the Hessian coordinates of each data features are computed. Let F be a set of m data features F_i ,

$$F = \{F_i, i = 0, \dots, m-1\} \quad (25)$$

where each F_i is represented in Hessian coordinates with respect to the robot reference frame O_{RxRyR}

$$F_i = (r_{F_i}, \theta_{F_i}, 1) \quad (26)$$

5.1 V-REP Simulation Test

The V-REP simulation makes use of a rigid body dynamic simulation of a quadrotor according with the formulation in [18]. The virtual scenario is composed by a short corridor and a square room inside which the quadrotor performs a trajectory defined by seven target positions. These target points are given as goal to the navigation control algorithm of the quadrotor and they have been selected in order to fully explore the virtual environment. In Fig. 1 a snapshot of the simulation is shown depicting the quadrotor moving along the corridor together with laser scan over the walls and a forward camera image. The green spot in the figure shows the target point currently taken as reference for the navigation control of the quadrotor.

The aim of this test session is mainly focus on the comparison of the proposed algorithm with the FastSLAM 2.0 algorithm, in terms of required algorithm execution time and robot pose estimation. The particle filter used

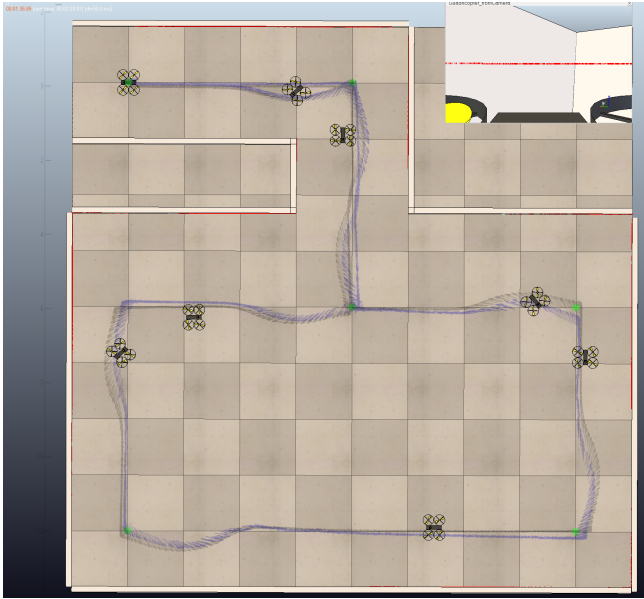


Figure 3: Estimated trajectory by the SLAMCGS (blue arrows) compared with the simulated one (black arrows). The target points are represented as green spots.

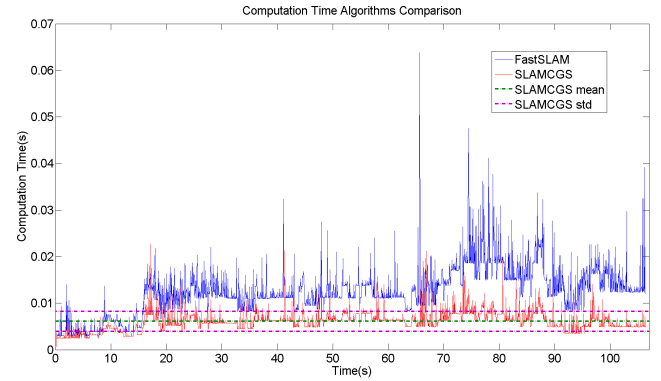


Figure 5: Execution time comparison between the SLAMCGS and the FastSLAM 2.0 during the simulation experiment.

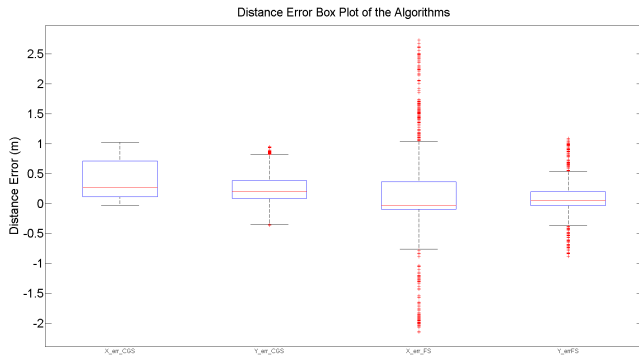


Figure 4: Position error shown as a box plot for x and y axis of both the algorithms (SLAMCGS and the FastSLAM 2.0)

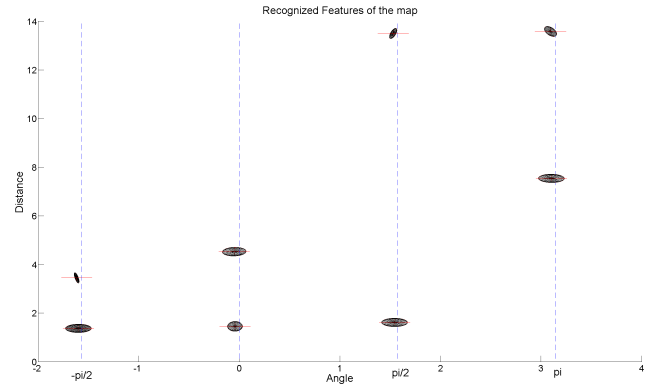


Figure 6: Maps features measured by the algorithm represented as covariance ellipsoids in model space. All the walls were correctly identified. (The ellipsoids in the figure has been scaled to make them readable).

in both the algorithms has 100 particles. The entire trajectory performed by the quadrotor is shown in Fig. 3 where all the target points defining the path of motion are visible in green. The simulated quadrotor moves forward in the small corridor, then turn right reaching the interior of the room and so it starts navigating the room in counterclockwise reaching each desired target point. Finally the robot turn right to exit the room and then left on the corridor corner to reach the starting point again. In the picture both the simulated trajectory (black arrows) and the one estimated by the SLAMCGS algorithm (blue arrows) are shown. The localization error resulting from the estimation algorithm is shown in Fig. 4 by means of a box-plot. The error has been computed for each axis (i.e. x and y axis) comparing the real position value coming from the V-REP simulation with the one resulting from the estimation process. The result obtained by the proposed algorithm is comparable with the one obtained by the FastSLAM 2.0 implementation. In particular, the FastSLAM 2.0 generates several outliers given the small number of features of the map, whereas the SLAMCGS performs better in presence of few map features. The computational time spent by both methods is presented in Fig. 5. The results prove that the SLAMCGS method improves on time performances and it is better suited for embedded processors. Finally, the Fig. 6 shows the measured features of the map during the whole simulation. The features have been drawn according to the model space diagram which shows the Hessian representation of the features extracted with the methods previously described in Sec. 5. The result shows that the algorithm correctly identifies the 8 features corresponding to the 8 walls in the map.

5.2 Hardware test

The hardware setup for the test is composed by:

- **Cyberquad platform:** The testing hardware is a custom version of the commercial CyberQuad MAXI developed by Cyber Technology (Perth, Australia). It is a MAV equipped with four brushless motors and it is generally used in the field of surveillance and detection. The device has an on-board controller, which provides a good flight stability and it allows to easy mount different payloads which provide the flexibility to carry a variety of different payload modules. The MAV dimension are 69x56x20 cm. For the details concerning the custom installed payload and the description of the control architecture, the reader can refer to [8].
- **Hokuyo laser range finder:** The quadrotor is equipped with a laser that scan in a line an area in a 270 degrees cone centered in the forward moving direction of the vehicle.

- **PANDA board:** All the computations of the platform (video streaming, communication with the operator control unit and SLAM computation) are run on this electronic board. The PandaBoard is a low-power, low-cost single-board development platform based on the Texas Instruments OMAP4430 which features a dual-core 1 GHz ARM Cortex-A9 MP-Core CPU, a 304 MHz PowerVR SGX540 GPU, IVA3 multimedia hardware accelerator with a programmable DSP, and 1 GB of DDR2 SDRAM. The connectivity is provided by wired 10/100 Ethernet as well as wireless Ethernet and Bluetooth. It also has two USB host ports and one USB On-The-Go port, supporting USB 2.0. In our current work, the device runs an Ubuntu Linux distribution.

In the hardware test the pilot has driven the vehicle keeping the altitude still on a virtual plane and performing line segments of 3 meters rotating of 90 degrees at each segment end, according to the following turning sequences: right, right, left, right, right, right, left, right returning in this way at the starting point with the same orientation.

The particle filter used again 100 particles and the maximum number of feature map was set to 10 given that the testing environment presented a reduced number of walls.

Figure 7 shows the resulting trajectory as estimated by the winning particle. It is possible to see that at each segment end the MAV performed a 90 degrees rotation and this result in an area dense of points in the plot.

Figure 8 reports the computational time spent by the proposed algorithms compared with the one spent by the FastSLAM 2.0. The results prove that the SLAMCGS method result faster, the average computing time is about 40ms in the employed PandaBoard.

6 Conclusion

The work presents an improved version of the FastSLAM 2.0, which has been optimized in order to take into the account the limited amount of computing power resulting from the embedded electronics into MAV platforms.

The implementation of the proposed SLAM algorithm has been reported and its validity has been proven both in a simulation environment and in a real test scenario. In the former case, the proposed algorithm has been compared with the FastSLAM 2.0 algorithm, in terms of required algorithm execution time and robot pose estimation accuracy. In the latter case, the algorithm has been embedded into a quadrotor in order to navigate inside our office room.

The results coming from the simulation experiment prove the effectiveness of our algorithm concerning the localization (the mean error value along the true tra-

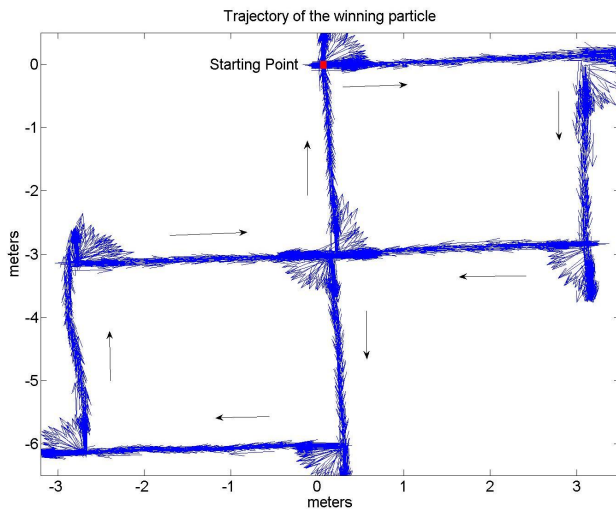


Figure 7: Estimated localization of the MAV in the hardware test.

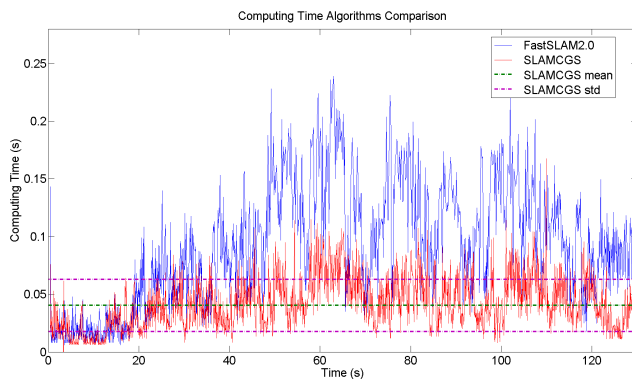


Figure 8: Execution time comparison between the SLAMCGS and the FastSLAM 2.0 on the embedded hardware.

jectory is comparable to the one obtained with the FastSLAM 2.0), whereas the comparison with the FastSLAM 2.0 highlights the goodness of the proposed algorithm in terms of execution speed and CPU load. On the other hand, the experimentation on the real platform confirms the results obtained from the simulation case and the robustness of the algorithm with respect to real test conditions (sensor noise and delays between the signals acquired by the employed sensors). The experimental test in the real scenario proves the feasibility of the algorithm running on a electronic board commonly used on platforms employed in the field.

References

- [1] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *AUTONOMOUS ROBOTS*, vol. 4, pp. 333–349, 1997.
- [2] E. B. Olson and S. Teller, "Robust and efficient robotic mapping," Tech. Rep., 2008.
- [3] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *Int. J. Rob. Res.*, vol. 5, no. 4, pp. 56–68, Dec. 1986. [Online]. Available: <http://dx.doi.org/10.1177/027836498600500404>
- [4] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, May 2013.
- [5] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-slambased navigation for autonomous micro helicopters in gps-denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011. [Online]. Available: <http://dx.doi.org/10.1002/rob.20412>
- [6] 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*. Edmonton, Canada: AAAI, 2002.
- [7] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: fifteen years later," in *Oxford handbook of nonlinear filtering*. Springer, 2011.
- [8] M. Satler, M. Unetti, N. Giordani, C. Avizzano, and P. Tripicchio, "Towards an autonomous flying robot for inspections in open and constrained spaces," in *International Conference on System, Analysis and Automatic Control*, 2014.
- [9] M. Montemerlo and S. Thrun, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *In Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2003, pp. 1151–1156.
- [10] K. Murphy, "Bayesian map learning in dynamic environments," in *In Neural Info. Proc. Systems (NIPS)*. MIT Press, pp. 1015–1021.
- [11] P. S. Maybeck, *Stochastic models, estimation, and control*, ser. Mathematics in Science and Engineering, 1979, vol. 141.
- [12] H. Moravec, "Sensor fusion in certainty grids for mobile robots," in *Sensor Devices and Systems for Robotics*, ser. NATO ASI Series, A. Casals, Ed. Springer Berlin Heidelberg, 1989, vol. 52, pp. 253–276.
- [13] M. Montemerlo and S. Thrun, "Simultaneous Localization and Mapping with Unknown Data Association Using FastSLAM," 2003, pp. 1985–1991.

- [14] J. Guivant and E. Nebot, "Optimization of the simultaneous localization and map building algorithm for real time implementation," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 242–257, 2001.
- [15] E. Rohmer, S. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 1321–1326.
- [16] J. J. Leonard, H. F. Durrant-Whyte, and O. Pj, "Directed sonar sensing for mobile robot navigation," 1992.
- [17] K. O. Arras and R. Y. Siegwart, "Feature extraction and scene interpretation for map-based navigation and map building," pp. 42–53, 1998. [Online]. Available: <http://dx.doi.org/10.1117/12.299565>
- [18] V. Martinez, "Modelling of the flight dynamics of a quadrotor helicopter," *A MSc Thesis in Cranfield University*, 2007.