Localization and Mapping in Local Occupancy Grid Maps: Simulation in Ackerman model mobile robot

Ronald A. Cardenas[1], Jasper W. Huanay[2] and Ivan Calle[2]

***Abstract*— To be considered autonomous, a robot must be capable of localizing itself in an unknown environment, while mapping it at the same time. This problem, known as Simultaneous Localization and Mapping is a thriving research area in robotics nowadays. We present localization and mapping algorithms for a mobile robot with Ackerman steering geometry (present in modern automobiles): a HPI Buggy fuel engine mini car. The maps area modeled as grid occupancy maps and depict three scenarios of the Mechanical Department building at Universidad Nacional de Ingenieria. The Monte Carlo Localization algorithm and the Occupancy Grid Mapping are implemented for the exploration tasks, tuning the noise parameters from the robot with experiments in outdoor environments.**

I. INTRODUCTION

The Simultaneous Localization and Mapping problem has received substantial improvement the last decade due to the advance in computing power and the incorporation of a probabilistic approach into the area. In this approach, the movement of the robot is no longer a deterministic process but a stochastic one, dealing with probabilistic models for the kinematics and sensor measurement. The amazing results of this approach were shown in the DARPA Grand Challenge in 2005, where Sebastian Thrun and the Stanford team won with Stanley, an AI Robot [4]. How Stanley works and what algorithms were involved are explained in detail in [5] and [6].

The foundations for the probabilistic modeling of the robot and sensors were presented in several books in the past decade ([1], [2], and [7]). Later on, Snider ([3]) would condense several techniques of path planning for automobiles.

The present work explores the application of localization and mapping algorithms for vehicles with Ackerman steering geometries. This report is organized as follows. Section II describes the maps used for the experiments. Section III defines the models used to approximate real Ackerman kinematics, as well as sensor modeling with range finder lasers. Section IV and V explain the algorithms used for localization and mapping, respectively. Section VI presents the details of the experiments, the tuned noise parameters, and each algorithm setup. Finally, section VII discuss the results and future work.

*This is a report describing research conducted at GISCIA Lab (http://giscia.github.io).

[1], [2] are sophomore students of Mechatronics Engineering at the Department of Mechanical Engineering, Universidad Nacional de Ingenieria, Peru
[1] roncardenasacosta@gmail.com
[1] hwilliamnq@gmail.com
[3] Ivan Calle is Assistant Professor at the Department of Mechanical Engineering, Universidad Nacional de Ingenieria, Peru ivan.calle.flores@gmail.com

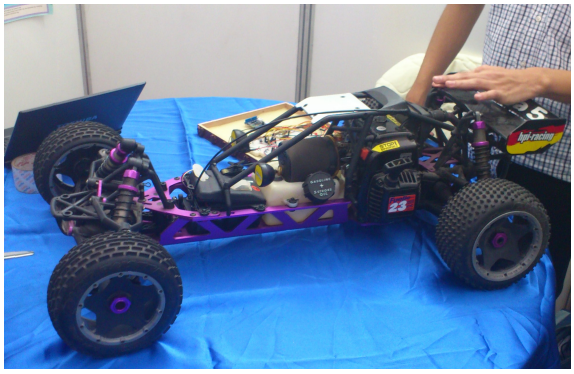


Fig. 2: HPI Buggy modeled for the experiments

II. DATASET

The experiments were performed on three hand-crafted binary 2D grid maps, presented in Figure 1. They depict real scenarios of the Mechanical Engineering Department (MED) at Universidad Nacional de Ingenieria. The first one shows the east wing of the third floor (1a), presenting some classroom doors open. The second one (1b) shows the main hall of the building, located at the first floor, showing some doors open and furniture. The third map (1c) depicts a part of a much longer corridor in the library. Here, individual cubicles can be seen distinguished.

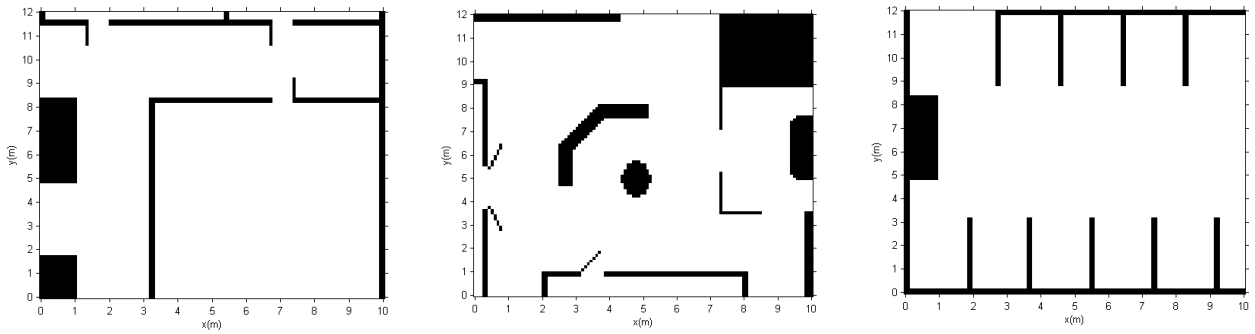
III. MOBILE ROBOT MODELING

A. Ackerman steering geometry

The Ackerman steering geometry is present in most ground vehicles nowadays, such as cars. This architecture was designed so that tires dont slip sideways when following curved paths. The design gives traction to the back wheels and leaves the steering to the front wheels. For modeling purposes, this steering geometry is often simplified and reduced to the well-known Bicycle model (Figure 3). The following assumptions and considerations are taken into account for the simplification:

- The set of four wheels is replaced by two, one for steering (front wheel) and the other for traction supply.
- The robot moves only over a plane.
- The speed and steering angles are not high nor wide, respectively, but moderate. This assures that the bicycle model's approximations are acceptable.
- Tires don't slip sideways.

The mobile robot modeled for the experiments is a HPI Baja 5B SS Buggy, an 1/10 scale sandpit vehicle with a 29cc



(a) Map 1: East wing of the third floor (b) Map 2: Main hall (c) Map 3: Entrance to Library

Fig. 1: Binary occupancy grid maps used in simulations. All of them depic scenarios of the Mechanical Engineering Department (MED).

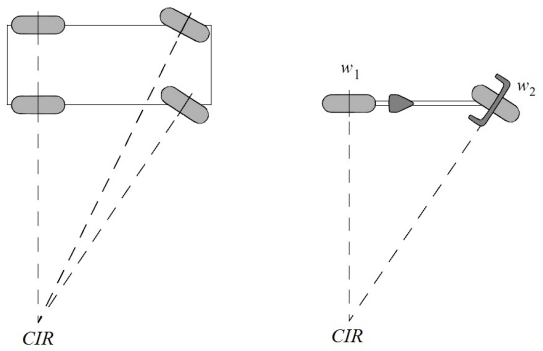


Fig. 3: Ackerman steering geometry simplified to Bicycle model for modeling purposes.

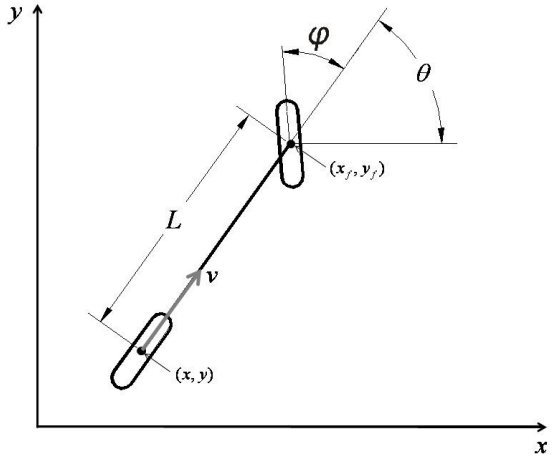


Fig. 4: Variables used in Kinematic model for the Bicycle model simplification.

fuel engine and Ackerman steering geometry. Figure 2 shows the Buggy, with width of 30cm and length of 60cm.

B. Kinematic model

The robot state is defined as the triplet (x, y, θ) containing the robot global position and orientation. The kinematic model allows to predict the following position state of the robot, taking into account noise sources inherent to its movement. Table I defines the variables used in the upcoming equations. Figure 4 allows identify the state variables in the Bicycle simplification.

Robot state:	
(x, y)	coordinates of the robot's gravity center
θ	robot's global orientation
Kinematic model:	
L	length of the robot
v	robot's linear velocity
T	Time period for state change
φ	steering angle
ε_α	Probabilistic noise with parameter α

TABLE I: Notation for kinematic model equations

Ideally, the position changing in each state would be given by the the following deterministic equation.

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{pmatrix} = \begin{pmatrix} x_t + Tv\cos(\theta) \\ y_t + Tv\sin(\theta) \\ \theta_t + \frac{Tv\tan(\varphi)}{L} \end{pmatrix} \quad (1)$$

However, the real movement of robot is noisy. This noise is modeled as Gaussian distributions and introduced into the model through linear (v) and rotational ($varphi$) velocity and then spread into position changing. The velocities are modeled as follow: constants plus Gaussian noise with media 0 and standard deviation dependent on a weighted sum of the square constant values, as stated in the equation:

$$\begin{pmatrix} \hat{v} \\ \hat{\varphi} \end{pmatrix} = \begin{pmatrix} v \\ \varphi \end{pmatrix} + \begin{pmatrix} \varepsilon_{\alpha_1 v^2 + \alpha_2 \varphi^2} \\ \varepsilon_{\alpha_3 v^2 + \alpha_4 \varphi^2} \end{pmatrix} \quad (2)$$

Replacing this velocities (eq. 2) in 1 the real movement

model is obtained, as follows:

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x + T\hat{v}\cos(\theta) \\ y + T\hat{v}\sin(\theta) \\ \theta_t + \frac{T\hat{v}\tan(\hat{\varphi})}{L} \end{pmatrix} \quad (3)$$

The noise parameters α_1, α_2, α_3, and α_4 are tuned through experiments, depending on the sensors and the navigation environment.

C. Sensor modeling

The sensor modeled for the experiments is a Hoyuko URG-04LX-UG01 Scanning Laser Rangefinder with the following features:

- Detectable range from 20 to 5600 mm, 1mm resolution.
- Scanning arc of 240°arc, 0.36°angular resolution.
- Scanning time: 100 msec/scan.
- Noise: 25dB or less.
- 5V operating voltage.

The measurement process is modeled as a conditional probability density depending on current position state, as follows:

$$p(z_t|x_t) = \prod_{k=1}^{K} p(z_t^k|x_{1:t}) \quad (4)$$

where K is the number of laser-beam measurements generated by the sensor, and k is each one of this measurements.

Given the high resolution of the sensor modeled, the whole measurement range is sub-sampled in order to get a smaller number of measurements and allow the system perform faster and at real time in a future real implementation.

The measurement model used was *likelihood field model*, which overcomes the lack of smoothness present in the beam-based sensor model. We assume three types of sources of noise and uncertainty:

1) Measurement noise: Modeled using Gaussians, involving find the nearest obstacle in the map. Let dist be the distance from the measurement coordinates $(x_{z_t^k}, y_{z_t^k})$ and the nearest object in the map m. Then the probability of a sensor measurement is given by a zero-centered Gaussian modeling the sensor noise:

$$p_{hit}(z_t^k|x_t, m) = \varepsilon_{\sigma_{hit}^2} dist^2 \quad (5)$$

2) Failures: Max-range readings (Z_{max}) are assumed to distinct large likelihood and modeled by a point-mass distribution p_{max} centered at Z_{max}.

$$p_{max}(z_t^k|x_t, m) = I(z = Z_{max}) \quad (6)$$

3) Random measurements: Range finders occasionally produce entirely unexplained measurements. This measurements will be modeled using a uniform distribution spread over the entire sensor measurement range $[0, Z_{max}]$:

$$p_{rand}(z_t^k|x_t, m) = \frac{1}{Z_{max}}, if 0 < z_t^k < Z_{max} \quad (7)$$

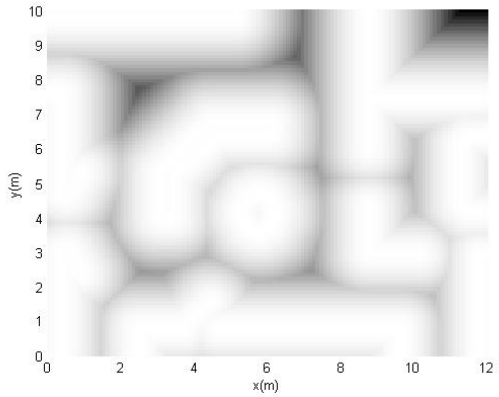


Fig. 5: Likelihood field for main halls map (Figure 1b).

These three distributions are now mixed by a weighted average, defined by the parameters z_{hit}, z_{max}, and z_{rand} with $z_{hit} + z_{max} + z_{rand} = 1$, as follows:

$$p(z_t^k|x_t, m) = z_{hit} * p_{hit} + z_{max} * p_{max} + z_{rand} * p_{rand} \quad (8)$$

Figure 5 depicts the likelihood of an obstacle detection as a function of global $x - y$ coordinates, called the likelihood field. In this example the main hall map (map2, 1b) is used.

IV. MONTE CARLO LOCALIZATION

For localization in the grid map, the *Monte Carlo Localization* (MCL) algorithm was used. Between its advantages worth mentioning we find that:

- It is capable of solving the global localization and, with the correct modification, kidnapped robot problems.
- It can process raw sensor measurement, no need to extract features from sensor values.
- It is non-parametric, e.g. it is not bounded to an unimodal distribution, as is the case for the Extended Kalman Filter localizer.

The MCL algorithm approximates the posterior of the model and measurement probabilistic models using the particles filter. It represents the belief $bel(x_t)$ by a set of M particles $\chi_t = \{x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]}$. The initial belief $bel(x_0)$ is obtained by randomly generating M such particles from the prior distribution $p(x_0)$ (uniform for our experiments), and assigning the uniform importance factor $1/M$ to each particle. Figure 6 shows the basic version of the algorithm.

The accuracy of the approximation is easily determined by the size of the particle set. Increasing the number of particles increases the accuracy, but trades off computational resources.

V. OCCUPANCY GRID MAPPING

Occupancy grid maps address the problem of generating consistent maps from noisy and uncertain measurement data, under the assumption that the robot pose is known. The basic idea is to represent the map as a field of random variables, arranged in an evenly spaced grid. Each random variable is

```
Algorithm MCL(X_{t-1}, u_t, z_t, m):
    X̄_t = X_t = ∅
    for m = 1 to M do
        x_t^[m] = sample_motion_model(u_t, x_{t-1}^[m])
        w_t^[m] = measurement_model(z_t, x_t^[m], m)
        X̄_t = X̄_t + ⟨x_t^[m], w_t^[m]⟩
    endfor
    for m = 1 to M do
        draw i with probability ∝ w_t^[i]
        add x_t^[i] to X_t
    endfor
    return X_t
```

Fig. 6: Monte Carlo Localization, a localization algorithm based on particle filters.

binary and corresponds to the occupancy of the location is covers.

Occupancy grid mapping algorithms implement approximate posterior estimation for those random variables, modeled as follows:

$$p(m|z_{1:t}, x_{1:t}) \tag{9}$$

where m is the map, $z_{1:t}$ the set of all measurements up to time t, and $x_{1:t}$ is the path of the robot, that is, the sequence of all its poses. The controls $u_{1:t}$ play no role in occupancy grid maps, since the path is already known. Hence, they are omitted.

The standard occupancy grid approach breaks down the map estimation problem to one of estimating the map cell by cell, as follows:

$$p(m|z_{1:t}, x_{1:t}) = p(m_i|z_{1:t}, x_{1:t}) \tag{10}$$

for all grid cell m_i. The occupancy grid mapping algorithm uses the log-odds representation of occupancy:

$$l_{t,i} = \log \frac{p(m_i = 1|z_{1:t}, x_{1:t})}{1 - p(m_i = 1|z_{1:t}, x_{1:t})} \tag{11}$$

with l_0 as the prior probability for each cell.

$$l_0 = \log \frac{p(m_i = 1)}{p(m_i = 0)} = \log \frac{p(m_i)}{1 - p(m_i)} \tag{12}$$

Figure 7 defines the algorithm used. The *inverse sensor model* implements the inverse measurement model $p(m_i|z_t, x_t)$ in its log-odds form. This model assigns to all cells within the sensor range frontier an occupancy value of $l_{occ} > l_0$ if the range of the cell is within $\alpha/2$ the detected range, where α controls the width of the frontier. It returns $l_{free} < l_0$ if the range to the cell is shorter than the measured range by more than $\alpha/2$.

Figure 8 shows an implementation of this inverse sensor model, where β is the sensor range coverage in radians.

```
Algorithm occupancy_grid_mapping({l_{t-1,i}}, x_t, z_t):
    for all cells m_i do
        if m_i in perceptual field of z_t then
            l_{t,i} = l_{t-1,i} + inverse_sensor_model(m_i, x_t, z_t) − l_0
        else
            l_{t,i} = l_{t-1,i}
        endif
    endfor
    return {l_{t,i}}
```

Fig. 7: The occupancy grid algorithm, a modified version of the binary Bayes filter.

```
Algorithm inverse_range_sensor_model(i, x_t, z_t):
    Let x_i, y_i be the center-of-mass of m_i
    r = √((x_i − x)^2 + (y_i − y)^2)
    φ = atan2(y_i − y, x_i − x) − θ
    k = argmin_j |φ − θ_{j,sens}|
    if r > min(z_max, z_t^k + α/2) or |φ − θ_{k,sens}| > β/2 then
        return l_0
    if z_t^k < z_max and |r − z_max| < α/2
        return l_occ
    if r ≤ z_t^k
        return l_free
    endif
```

Fig. 8: A simple inverse measurement model for robots equipped with range finders.

VI. EXPERIMENTS

A. Model parameters tuning

1) Grid dimensions: Each map represents an area of 12x10m^2, with grid size of 0.10m.

2) Robot dimensions: The following dimensions and limitations were considered while defining the kinematic motion model.

- Width: 30cm.
- Large: 60cm.
- Maximum steering angle: 34°.
- Distance from wheels to longitudinal robot axis: 15cm.

3) Motion model: The noise parameters for the kinematic motion model were tuned performing outdoor controlled driving experiments with the HPI Buggy car. These parameters, defined as α in section III-B, are given the following values:

$$\alpha = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} = \begin{pmatrix} 10^{-3} \\ 10^{-4} \\ 10^{-3} \\ 10^{-4} \end{pmatrix} \tag{13}$$

4) Measurement model: The mixing weights for each distribution considered in $p(z_t^k|x_t, m)$ were tuned while calibrating the sensor.

$$\begin{pmatrix} z_{hit} \\ z_{rand} \\ z_{max} \end{pmatrix} = \begin{pmatrix} 0.333 \\ 0.111 \\ 0.556 \end{pmatrix} \tag{14}$$

The intrinsic noise parameter (σ_{hit}) was derived from the data sheet of the sensor and set to 0.0075m or 7.5mm. The whole coverage range was sub-sampled, taking only 21 ray samples from a coverage of 200°.

B. Localization

The localization algorithm was configured with the following parameters and initial setup:

- Number of particles (M): 500.
- Initial particles' state (x, y, θ) drawn from uniform distribution. Particles falling into an occupied grid cell were resampled until they fall into an unoccupied one.
- Initial weight of particles set to $1/M$.

Figures 9, 10, and 11 show the simulations in all three maps. For each map, three stages of the localization are shown. The first one (figures 9a, 10a, and 11a) is the initial state of the particles. The second one (figures 9b, 10b, and 11b) is after a few iterations, when the robots has already localized itself. The third one (figures 9c, 10c, and 11c) shows how it keeps localizing itself and converging to an even more accurate real position over time.

C. Mapping

For mapping experiments, all the route of the robot (position states $x_{1:t}$ is considered as known. The probability of an occupied cell, used to define l_0 in equation 12, is defined as the sum of all occupied cells divided by the number of cells in the map, as follows.

$$p(m_i = 1 | z_{1:t}, x_{1:t}) = \frac{\sum_{i \in nxm} I(m_i == 1)}{n * m} \tag{15}$$

where n and m are the dimensions of the binary grid map. The relationship between the log form probabilities l_0, l_{occ}, and l_{free} given by

$$l_{free} < l_0 < l_{occ} \tag{16}$$

is achieved by scaling l_0 by certain factor (keeping in mind that it is negative) as follows:

$$\begin{aligned} l_{occ} &= 0.5 * l_0 \\ l_{free} &= 1.5 * l_0 \end{aligned} \tag{17}$$

Figure 12, 13 and 14 show the experiments in all three maps. The inferred occupancy maps show clearly the difference between obstacles (dark grey cells), free space (white cells), and unknown areas (grey cells). The red dots show each position state of the robot.

VII. RESULTS DISCUSSION AND FUTURE WORK

The results obtained in all three maps reassure the robustness of the algorithms analyzed. Even when high measurement noises delay the algorithms convergence a couple of iterations more, these always were capable of localize or infer a valid map.

The Monte Carlo Localization algorithm benefits highly from the richness wall-surface forms (such as seeing in map 1c), making the algorithm converge in no more than 5 iterations with good features. The first map (1a) was particularly challenging, since both hallways look alike. After further navigation, the robot was able to accurately localize itself, despite having two of three main high probability regions for the position state in the first iterations.

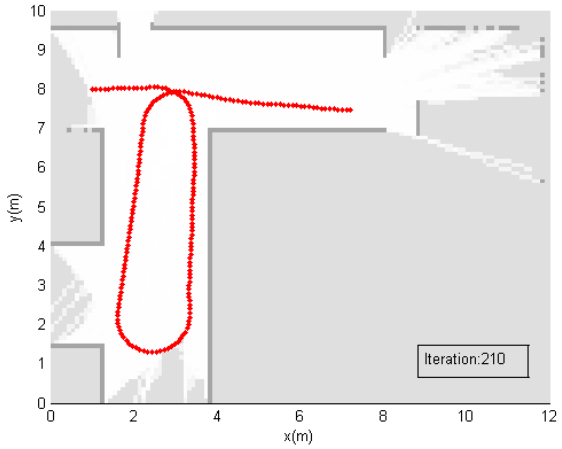


Fig. 12: Inferred occupancy grid map for map 1.

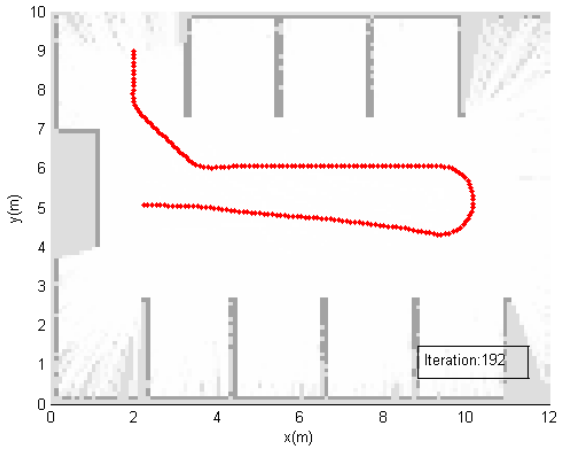


Fig. 13: Inferred occupancy grid map for map 2.

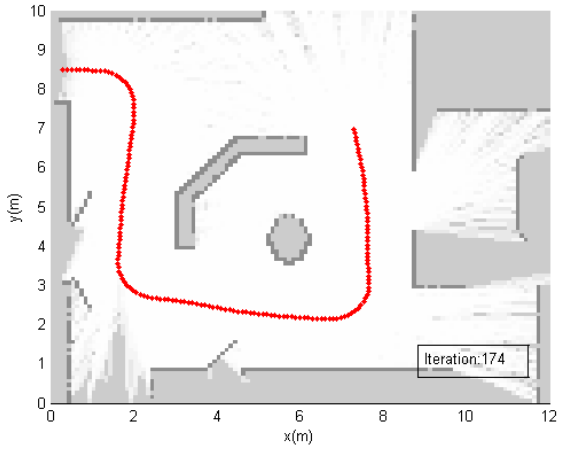
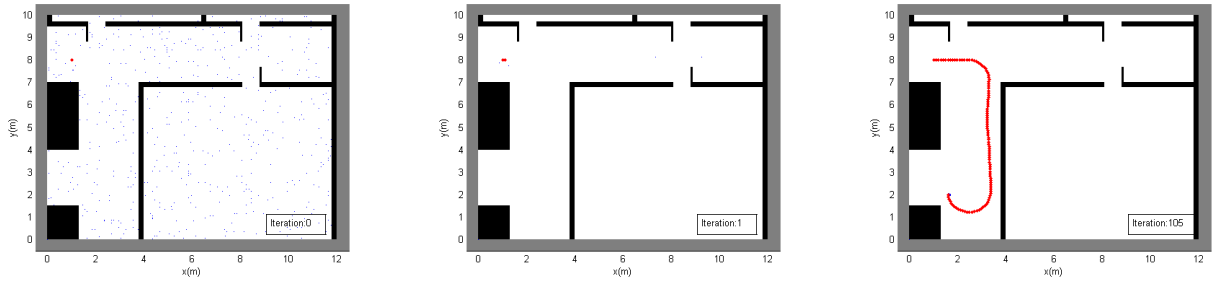
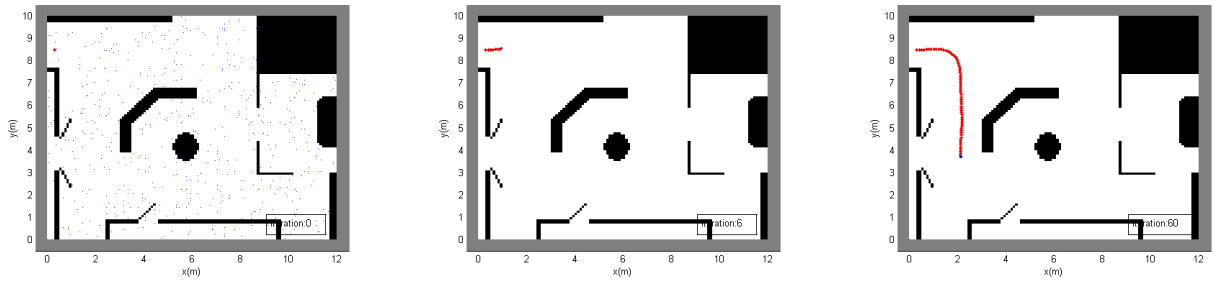


Fig. 14: Inferred occupancy grid map for map 2.



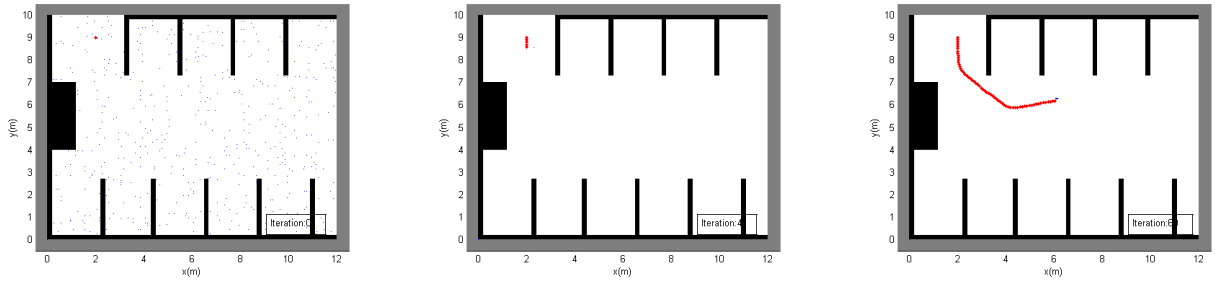
(a) Particles set initialized (b) After 1 iteration (c) After 105 iterations

Fig. 9: Monte Carlo Localization algorithm experiments in map 1.



(a) Particles set initialized (b) After 6 iteration (c) After 60 iterations

Fig. 10: Monte Carlo Localization algorithm experiments in map 2.



(a) Particles set initialized (b) After 4 iteration (c) After 64 iterations

Fig. 11: Monte Carlo Localization algorithm experiments in map 3.

The Occupancy grid mapping algorithm output is highly dependent of the kinematic model for the robot, since slight deviations could propagate and produce a distorted map, without proper noise modeling. Maps rich in wall-surface (1b and 1c) forms normally need more than one lap of navigation in order to get a high quality map.

Future lines research in robot navigation for this kind of mobiles include vision-based and graph-based SLAM techniques; 3D robot navigation with drones; and the combination of visual, sensorial and textual information about the environment ([8], [9], [10], [11]).

ACKNOWLEDGMENT

This research has been supported by the Mechanical Engineering Department. We also gratefully acknowledge the Artificial Intelligence and Control Systems Laboratory (GIS-CIA) for provide us with the necessary infrastructure and computational resources. In addition, we highly appreciate all the useful reviews and guidelines provided by Professor Alberto Coronado and Ricardo Bustinza.

REFERENCES

[1] Thrun, S., Burgard, W., and Fox, D.. 2005. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press.

[2] Siegwart, R., Nourbakhsh, I., and Scaramuzza, D.. 2011. Introduction to Autonomous Mobile Robots (2nd ed.). The MIT Press.

[3] Snider, J.. 2009. Automatic Steering Methods for Autonomous Automobile Path Tracking. Carnegie Mellon University.

[4] Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., and Mahoney, P. (2006). Stanley: The robot that won the DARPA Grand Challenge. Journal of Field Robotics, 23(9), 661692.

[5] Thrun, S., Montemerlo, M., and Aron, A., Probabilistic Terrain Analysis For High-Speed Desert Driving. Proceedings of Robotics Science and Systems, Philadelphia, PA, USA, 2006.

[6] Montemerlo, M., Thrun, S., Dahlkamp, H., Stavens, D., and Strohband, S., Winning the DARPA grand challenge with and AI robot, in American Association for Articial Intelligence, 2006.

[7] Hartley, R. and Zisserman, A. 2000. Multiple view geometry in computer vision, Cambridge University Press: Cambridge, UK.

[8] Wei, Y., Brunskill, E., Kollar, T., and Roy, N., 2009. Where to go: interpreting natural directions using global inference. In ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation, pages 37613767, Piscataway, NJ, USA. IEEE Press.

[9] Tellex, S., Kollar, T., Dickerson, S., Walter, M., Banerjee, A. G., Teller, S., and Roy., N. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In Association for the Advancement of Artificial Intelligence (AAAI).

[10] Walter, M., Hemachandra, S., Homberg, B., Tellex, S., and Teller, S., Learning semantic maps from natural language descriptions. In RSS, 2013.

[11] Kollar, T., Tellex, S., Roy, D. and Roy, N., 2010. Toward Understanding Natural Language Directions. Proceedings of the 5th ACM/IEEE International Conference on Human-robot Interaction HRI'10, Osaka, Japan, 2010, 259-266.