iCERi 2010

International Conference of Education, Research and Innovation

CONFERENCE PROCEEDINGS

Madrid (Spain) - 15th-17th of November, 2010

iCERi 2018

International Conference of Education, Research and Innovation

CONFERENCE PROCEEDINGS

Published by
International Association of Technology, Education and Development (IATED)
www.iated.org

ICERI2010 Proceedings CD
Edited by
L. Gómez Chova, D. Martí Belenguer, I. Candel Torres
International Association of Technology, Education and Development
IATED, Valencia, Spain

ISBN: 978-84-614-2439-9
Depósito Legal: V-3998-2010

Book cover designed by
J.L. Bernat

SOFTWARE FOR THE LEARNING OF MONTE CARLO LOCALIZATION STRATEGIES USING OMNIDIRECTIONAL IMAGES

L. Fernandez, O. Reinoso, L. Paya, A. Gil, M. Ballesta

Universidad Miguel Hernández de Elche. Departamento de Ingeniería de Sistemas Industriales, Alicante (SPAIN)
l.fernandez@umh.es, o.reinoso@umh.es, lpaya@umh.es, arturo.gil@umh.es, m.ballesta@umh.es

Abstract

Computing the localization of a robot in an environment is an essential task when the robot has to carry out an autonomous task in that environment. In the scientific literature we can find many methods, and one of the most extended currently is Monte Carlo Localization algorithm. On the other hand in this kind of applications, the appearance-based approach has attracted the interest of researchers recently due to its simplicity and robustness.

In this work, we present a tool we developed to be used in a robotics subject. It provides the students an intuitive platform to test the Monte Carlo Localization algorithms when working with some set of omnidirectional images captured in real environments and with an appearance-based descriptor. This tool is very useful to provide real data and to facilitate some mechanisms to understand this probabilistic localization algorithm.

Keywords: Mobile robotics, omnidirectional images, Monte-Carlo localization.

1 INTRODUCTION

Finding the location of the robot is a key issue in the field of mobile robotics. The knowledge of its position in the space is crucial for an autonomous agent, since the pose is needed for a precise navigation. During the last decades the Monte Carlo algorithm has been extensively used in localization tasks in the field of mobile robotics, demonstrating both a large degree of robustness and efficiency ([1], [2], [3]). The different approaches of Monte Carlo localization methods presented to date differ depending on the sensor installed on the robot. For example in [3] a laser range sensor is used to localize the robot in the environment. The differences between the expected and the current laser measurements are used to weight the particles and discard the most unlikely ones. In [1], a camera pointing to the ceiling of the environment was used. The Monte Carlo algorithm compared the brightness in the center of the images with an image of the ceiling as seen from below. The difference between the expected and actual illumination was used to weight the particles and estimate the position of the robot. In [2] a stereo camera system is used to obtain observations from a set of point features in the environment. The observed distance computed from the stereo images is used to refine the robot location in the map.

In this paper we study the case of robot localization using omnidirectional images. A Monte-Carlo method is deployed to find the pose of the robot in a map. In particular, we have developed a software tool that is intended for teaching purposes. This tool allows the students to test different weight methods in the Monte-Carlo algorithm, along with different strategies in relation with the data association method. The main motivation for using the omnidirectional images that a camera installed on a robotic platform is its low cost, weight and moderated power consumption and due to the fact that these cameras are able to provide a high quantity of information. In this paper, we consider that the camera is installed at a fixed orientation with respect to the robot and pointing upwards in direction to an omnidirectional mirror. We also assume that the movement of the robot is restricted to a plane. In this case, a rotation of the robot corresponds approximately to a shift in the columns of the panoramic image.

To date, researchers have proposed different solutions for the localization problem using omnidirectional images. These solutions can be divided into two main groups: feature-based solutions, in which a number of significant points from each omnidirectional image are extracted, and next, each point in the image is described using an invariant descriptor (for example [4] uses omnidirectional images to find the location of the robot in a given map, using SURF features [5]), and appearance-

ISBN: 978-84-614-2439-9

based solutions, in which the whole appearance of the image is represented by a single descriptor ([6], [7]).

In a prior work [8], the results showed that appearance-based solutions are a good choice in combination with Monte Carlo methods. However, the weight methods used can lead to non-optimal results, since there exist several methods to obtain the localization of the robot based on an observation function. With this motivation, in this paper we present an educational tool that is intended to be used in the teaching of mobile robotics and, in particular, Monte Carlo methods. The tool presented here allows the student to assess different methods that allow to compute a weight for each particle and carry out a comparison in terms of the error in localization. Some of the methods included in the tool have been used in the past for localization tasks using particle filters (e.g. [9]), but, in addition, we present some own techniques [8] that in some cases are able to provide also good results in terms of accuracy. Thanks to this easy-to-use platform, students can understand the fundamentals of this method and make a comparison between different approaches.

The tool presented here assumes that a grid represents the map, where several omnidirectional images are taken at certain positions of the environment. This approach differs form the proposed by other authors, who represent the environment by taking images during a fixed trajectory of the robot [9]. The grid-approach, however, is considered as a more complete representation of the environment. Second, the educational tool described here permits testing different weight methods that allow to localize the robot using a particle filter, as well as different ways to perform the data association in the suggested map.

In this map, each omnidirectional image is described by a single Fourier descriptor that represents the appearance with invariance to the rotation. This descriptor has been chosen based on a prior work [10], in which the Fourier descriptor allowed a fast comparison between the current image and the map by means of a vector distance measurement. In addition, we found [10] that the processing time needed to compute the Fourier transform is comparable to common feature extraction and description methods.

The rest of the paper is organized as follows: section 2 describes the Fourier transform used in the context of omnidirectional images. Next, section 3 describes the Monte-Carlo algorithm and its application to the problem of localization in mobile robotics. Following, section 4 presents the weight methods that have been included in the educational application and can be tested to study different ways to include the likelihood of observations during Monte Carlo localization. Section 5 describes the appearance and utilisation of the educational tool and presents some initial results obtained. Finally, section 6 exposes the main conclusions and proposes future work.

2 FOURIER SIGNATURE WITH OMNIDIRECTIONAL IMAGES

To date, different description methods have been used in the context of omnidirectional robot vision. In this work, we make use of Fourier-based techniques. Given an omnidirectional image $f(x,y)$ with N_x rows and $N_y$columns, we can obtain the most relevant information from the image by means of the Discrete Fourier Transform. In order to do so, there are several possibilities, such as to implement the 2D Discrete Fourier Transform [10], the Spherical Fourier Transform of omnidirectional images [11] or the Fourier Signature of the panoramic image [6]. The Fourier signature exploits better the invariance to ground-plane rotations in panoramic images [10]. The most important information is concentrated in the low frequency components of each row, so we can work only with the information from the k first columns in the Signature. Also, this feature presents rotational invariance. It is possible to prove that if each row of the original image is represented by the sequence a_n and each row of the rotated image by a_{n-q} (being q the amount of shift), when the Fourier Transform of the shifted sequence is computed, we obtain the same amplitudes A_k than in the non-shifted sequence, and there is only a phase change, proportional to the amount of shift q, as described in the following equation:

$$F[\{a_{n-q}\}] = A_k \exp - j\frac{2\pi qk}{N_y}; \quad k = 0,\text{K}\ ,N_y - 1 \tag{1}$$

3 MONTE CARLO LOCALIZATION

In robot localization we are interested in the estimation of the pose of the vehicle (typically, the state $x_t = (x, y, \theta)$) at time t using a set of measurements $z_{1:t} = \{z_1, z_2, \ldots, z_t\}$ from the environment (in our case, the Fourier Signatures of the omnidirectional images) and the movements $u_{1:t} = \{u_1, u_2, \ldots, u_t\}$ of the robot. In *Monte Carlo Localization (MCL)*, the probability density function $p(x_t \mid z_{1:t}, u_{1:t})$ is represented by a set of M random samples $\chi_t = \{x_t^i, i = 1, \ldots M\}$ extracted from it, named particles. Each particle can be understood as a hypothesis of the true state of the robot $x_t^i = (x^i, y^i, \theta^i)$. The weight of each sample (particle) determines the importance of the particle. The set of samples defines a discrete probability function that approximates the continuous belief.

The initial set of particles represents the initial knowledge $p(x_0)$ about the state of the mobile robot on the map. When we use a particle filter algorithm, in global localization, the initial belief is a set of poses drawn according to a uniform distribution over the robot's map. If the initial pose is partially known up to some small margin of error (local localization), the initial belief is represented by a set of samples drawn from a narrow Gaussian centered at the known starting pose of the mobile robot. The Monte Carlo Localization algorithm is described briefly in the next lines, and consists of two phases:

- **Prediction Phase**: At time t a set of particles $\overline{\chi}_t$ is generated based on the set of particles $\overline{\chi}_{t-1}$ and a control signal u_t. This step uses the motion model $p(x_t \mid x_{t-1}, u_t)$. In order to represent this probability function, the movement u_t is applied to each particle while adding a pre-defined quantity of noise. As a result, the new set of particles $\overline{\chi}_t$ represents the density $p(x_t \mid z_{1:t-1}, u_{1:t})$.
- **Update Phase**: In this second phase, the observation z_t obtained by the robot is used to compute a weight ω_t^i for each particle in the set $\overline{\chi}_t$. This weight represents the observation model $p(z_t \mid x_t)$ and is computed as $\omega_t^i = p(z_t \mid x_t^i)$. In this paper we propose different methods for the computation of this weight that will be described in Section 4. The weights are normalized so that $\sum \omega_t^i = 1$. As a result, we obtain a set of particles accompanied by a weight $\overline{\chi}_t = \{x_t^i, \omega_t^i\}$.

The resulting set χ_t is calculated by resampling with replacement from the set $\overline{\chi}_t$, where the probability of resampling each particle is proportional to its importance weight ω_t^i. Finally, the distribution $p(x_t \mid z_{1:t}, u_{1:t})$ is represented by the set χ_t.

4 WEIGHT METHODS

As described in the previous section, the Monte Carlo Algorithm introduces the current observation z_t of the robot by means of computing a weight ω_i for each particle and performing a resampling process. In this section, we describe different strategies for the computation of the weights. These weights have been included in the educational tool. The student may repeat every localization experiment while using different weighting methods and compare the results in terms of mean localization error or convergence speed.

We consider that our map is formed by a set of N bi-dimensional landmarks $L = \{l_1, l_2, \ldots, l_N\}$, forming a grid in the environment with a particular resolution. Each landmark l_j has an omnidirectional image I_j associated and a Fourier descriptor d_j that describes the global appearance of the image, thus $l_j = \{(l_{j,x}, l_{j,y}), d_j, I_j\}$. Next, we describe the localization method proposed. We consider that at time t the robot captured an image and computed the Fourier descriptor d_t. Using this Fourier descriptor we compare the descriptor d_t with the rest of descriptors $d_j, j = 1 \ldots N$ and find the B landmarks in the map that are closest in appearance with the current image I_t. In this sense, we allow the correspondence of the current observation to several landmarks in the map. We consider that this is a special case of the data association problem. In addition, this correspondence benefits the localization algorithm, since it may restrict the computation of the observation model to a reduced set of landmarks, thus reducing the computational effort. The students can obtain results when varying this

parameter in order to assess its influence. In addition the selection of B landmarks in terms of appearance will allow us to evaluate the importance of the description method used.

We base the localization of the robot on the Monte Carlo algorithm explained in the previous section. Next, we propose several methods that allow to compute the weight of each particle $\omega_t^i = p(z_t \mid x_t^i)$, thus providing different observation models:

Weight Method 1 (W1): product of gaussians centered on each image landmark considering the distance to the descriptor.

$$\omega_t^i = \prod_{j=1}^{B} \exp\{-v_j \sum_l^{-1} v_j^T\} \qquad (4)$$

where, $v_j = (l_{j,x}, l_{j,y}) - (x^i, y^i)$ is the difference between the position of the landmark l_j and the position (x^i, y^i) of the particle i. The matrix $\sum_l$ is a diagonal matrix $\sum_l = diag(\sigma_l^2, \sigma_l^2)$. The variance σ_l^2 is chosen experimentally in order to minimize the error in the localization and the students cannot change its value in the tool. We recall that the product of gaussian distributions is also a gaussian. The results demonstrate that this method tends to center the particles rapidly near the true robot pose, however, it suffers from some problems when the data association phase fails (e.g. the selected landmark l_j lies far away from the actual robot pose).

Weight Method 2 (W2): Sum of gaussians centered on each image landmark.

$$\omega_t^i = \sum_{j=1}^{B} \exp\{-v_j \sum_l^{-1} v_j^T\} \qquad (5)$$

where v_j and matrix $\sum_l$ is analogous to the matrix defined in W1 and its values were also selected experimentally. In this case, the observation model $p(z_t \mid x_t)$ is not gaussian, since it is formed by a sum of gaussians, being thus multi-modal. As we will show in the experimental results, this method is less sensitive to errors in data association whereas it is able to achieve nice localization results.

Weight method 3 (W3): sum of gaussians centered on each image landmark and considering the difference in the descriptors.

$$\omega_t^i = \sum_{j=1}^{B} \exp\{-v_j \sum_l^{-1} v_j^T\} \exp\{-h_j \sum_d^{-1} h_j^T\} \qquad (6)$$

where v_j and $\sum_l$ have been defined in the previous methods and $h_j = \mid d_j - d_t \mid$ defines the difference between the module of the Fourier descriptor associated to the current image observed and the module of the descriptor associated to the landmark l_j. The descriptors are normalized so that the summation of the euclidean distance of the current descriptor d_t to the rest of the B associations equals one, $\sum_{j=1}^{B} h_j = 1$. The matrix $\sum_d = diag(\sigma_d^2)$ is an $k \times k$ matrix, being k the length of the Fourier descriptor. The main difference of this method with respect to W2 is the consideration of the difference in the descriptor in the observation model $p(z_t \mid x_t)$. This fact generally gives higher weights to particles situated near a landmark that is close in appearance to the current observation.

Weight method 4 (W4): product of gaussians centered on each image landmark and considering the difference in the descriptors.

$$\omega_t^i = \prod_{j=1}^{B} \exp\{-v_j \sum_l^{-1} v_j^T\} \exp\{-h_j \sum_d^{-1} h_j^T\} \qquad (7)$$

where v_j, h_j, $\sum_l$ and $\sum_d$ have been defined in the previous methods. This method is similar to method W1 but considering the effect of the similarity in the description when computing the weight.

Weight method 5 (W5): sum of gaussians centered on each landmark position and considering the difference in the descriptors as well as the orientation of the landmarks (images).

$$\omega_t^i = \sum_{j=1}^{B} \exp\{-v_j \Sigma_l^{-1} v_j^T\} \exp\{-h_j \Sigma_d^{-1} h_j^T\} \exp\{-g_j \Sigma_\theta^{-1} g_j^T\} \qquad (8)$$

where v_j, h_j, Σ_l and Σ_d have been defined in the previous methods. The variable $g_j = (\theta_j - \theta_i)$ computes the difference between the expected orientation θ_j and θ_i the orientation of the particle. Given the current descriptor d_t and the descriptor d_j the orientation θ_j can be computed from equation 1. In this case, and since the map is known, the orientation of all the landmarks (images) in the map is known in advance. The matrix Σ_θ is selected experimentally and the students cannot change its value.

Weight method 6 (W6): product of gaussians centered on each landmark position and considering the difference in the descriptors as well as the orientation of the landmarks (images).

$$\omega_t^i = \prod_{j=1}^{B} \exp\{-v_j \Sigma_l^{-1} v_j^T\} \exp\{-h_j \Sigma_d^{-1} h_j^T\} \exp\{-g_j \Sigma_\theta^{-1} g_j^T\} \qquad (9)$$

where v_j, h_j, g_j, Σ_l, Σ_d and Σ_θ have been defined in the previous methods. This method is similar to the W5, but considering the product of the gaussian distributions.

Weight method 7 (W7): gaussian distribution at the center of mass of a discrete particle system. This method is inspired in a system of particles, each one having a mass related to the similarity with the current descriptor d_t observed by the robot. The weight for each particle is computed as:

$$\omega_t^i = \exp\{-f_j \Sigma_f^{-1} f_j^T\} \qquad (10)$$

where $f_j((x^i, y^i) - \hat{c})$ computes the difference between the position of the particle i and the center of mass computed as:

$$\hat{c} = \sum_{j=1}^{B} l_j \cdot m_j \qquad (11)$$

where the virtual mass m_j is computed as $m_j = \exp\{-h_j \Sigma_d^{-1} h_j^T\}$. The masses m_j are normalized so that $\sum_{j=1}^{B} m_j = 1$. The covariance matrix Σ_f is computed as the covariance associated to $\hat{c}$.

Weight method 8 (W8): gaussian distribution at the center of a spring-mass system. This method is inspired by a spring-mass system [6]. The constant of each spring is related to the similarity in the description, thus, landmarks more similar to the current observation try to atract the mass more tightly. To simplify the calculations, m_j is equal to 1 for all the mass of the system. The weight for each particle is computed as:

$$\omega_t^i = \exp\{-f_j \Sigma_s^{-1} f_j^T\} \qquad (12)$$

where $f_j((x^i, y^i) - \hat{c})$ computes the difference between the position of the particle i and the center $\bar{c}$ of a spring-mass system. In this case, the matrix Σ_s is computed as the covariance associated to $\bar{c}$.

Weight method 9 (W9): triangular distribution. This method is inspired in the weight function introduced by [9]. The weight for each particle is computed as:

$$\omega_t^i = \frac{1}{B} \sum_{j=1}^{B} S_j (D_{\max}^i - \sqrt{v_j v_j^T}) \qquad (13)$$

where $S_j = (1 - \sqrt{h_j h_j^T})$ and $v_j = (l_{j,x}, l_{j,y}) - (x^i, y^i)$ computes the difference between the position of the particle i and the landmark j. $D^i_{\max}$ is the metric distance between the farthest landmark and the position of the particle i. This weight method represents a triangular distribution centered on each landmark as to the appearance of each acquired image.

As we show in section 6, using the developed tool, students can compare the performance of these weight methods in a quite intuitive way an all the results and evolution of the process are shown graphically.

5 DESCRIPTION OF THE EXPERIMENTS

5.1 Database creation

To build the databases we have included in the tool, we have used a Pionner P3-AT robot (fig.1a) that is equipped with a catadioptric system (fig. 1b). The catadioptric system consists on a forward-looking camera and a parabolic mirror that provides omnidirectional images of the environment. The omnidirectional images are transformed to panoramic images with a size of 64x256 pixels to work with this information in an efficient way (fig. 1c).

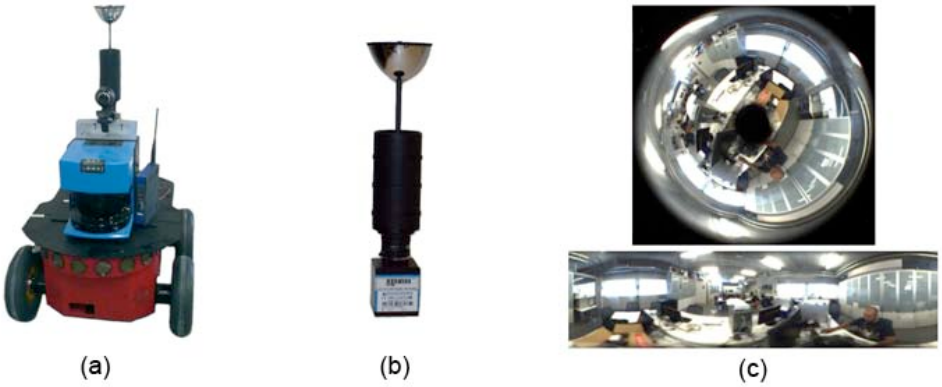

Fig. 1. (a) Pioneer P3-AT robot, (b) catadioptric system and (c) omnidirectional and panoramic image

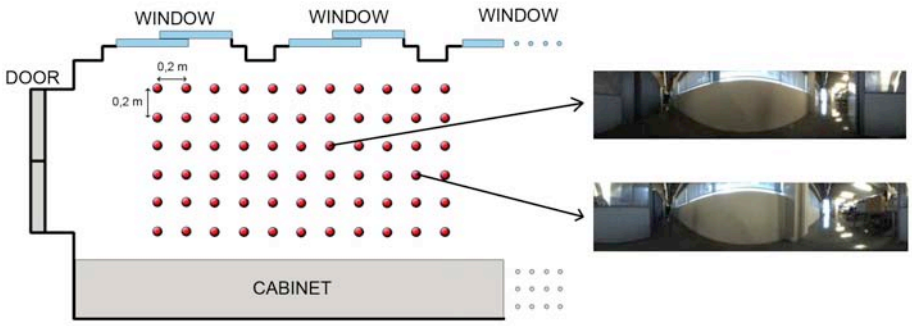

Fig. 2. Bird eye's view of the grid used to build the map, with two examples of panoramic images

We have captured a set of omnidirectional images on a pre-defined *10x10 cm* grid in a corridor (an indoor environment). We have taken a total number of *131* images (*21* images in *x-axis* and *11* images in *y-axis*). To build the map, we only used a pre-defined 20x20 cm grid of these images, with a total number of *66* images (*11* images in *x-axis* and *6* images in *y-axis*), one in two images (the rest images will be used in a later phase to test the localization algorithms). We can see a bird's eye view of the grid used to build the map and two examples of panoramic images on fig. 2.

5.2 Localization of the robot with MCL

With the complete set of images described in the section 5.1, we have build 6 different paths to simulate the robot movement on map. The objective of this section is to describe how to compute the localization of the robot when it moves on the different paths. It must be carried out just comparing the visual information with the information in the map and with a Monte Carlo Localization algorithm. This process is carried out in the following way:

5.2.1 Fourier signature

To compute the localization of the robot for each image of the path, we compute the difference between the appearance of this image with respect to the appearance of the map images. With this aim, we use the euclidean distance between the Fourier signatures. If $d_i(u,v)$ is the Fourier signature of the image $I_i(x,y)$ and $d_j(u,v)$ is the Fourier signature of the image $I_j(x,y)$, then the distance between scenes i and j is:

$$D_{ij} = \sqrt{\sum_{u=0}^{N_x}\sum_{v=0}^{N_y}(d_i(u,v) - d_j(u,v))^2}$$

5.2.2 Monte Carlo algorithm

Once we have obtained the euclidean distance between the images, we use these distances to compute the weight of the particles of the particle filter. The process employed is the process explained in the section 3.

6 DESCRIPTION OF THE TOOL

In this section we present a detailed description of the operation of our application. The simplicity of use is the general philosophy while designing this tool, trying to guide the user during the whole process and taking into account the fact that the user may not be an expert in the robotics field. The main objective was to provide the students with a set of weight methods to understand and to test the validity of appearance-based Monte Carlo localization and the necessary tools to put them into practice and to make a critical comparison of the performance of them in a real environment.

6.1 User's manual

We have developed the application using MATLAB [12]. The application must be launched from this program so the student has to run first MATLAB in his computer. When the user runs the tool a graphical interface appears where the students can carry out all the necessary operations. We can see the appearance of this graphical interface during an experiment in fig. 3. In general, this figure shows all the necessary options to perform the Monte Carlo Localization process (in an interactive way), a bird's eye view of the map, the particles set, the particles center, the odometry travelled path, the real travelled path and the actual real pose of the robot. All these data are updated step by step so that student can understand the process.

Hereafter we describe the steps to be completed during a typical sequence of use of the tool and the different options it offers to the students. The first step consists in configuring the parameters of the movement of the robot. This step, on the one hand, allow us to simulate the errors that occur in the robot's odometry, and on the other hand, permits choosing between different possible routes and the distance travelled in each movement (fig. 4). When we want to configure the odometry error, we have to configure the parameters alpha. These parameters correspond to the variances of the Gaussian errors that are involved in the odometry simulation. When we modify these parameters *alpha*, we can see how the movement of the particles change (fig. 5).

In the second step of the experiment, the student has to choose the parameters related to the Monte Carlo Filter. First, the user has to choose the number of the particles used to do the experiment. With this parameter the student will be able to understand the importance of the number of particles (hypotheses) in a Monte Carlo Localization process, and the necessity to find an appropriate number of samples for the localization to work correctly. Secondly, the user has to choose among the eight different types of weight that we have proposed in section 4. The user can see the performance of the localization under the different methods. After that, the student can choose the number of associations

the robot must realize to compute the weight of the particles of the system. This parameter (B) will influence the localization process; if this parameter takes a relatively low value, the error in localization remains small, but as the number of associations increases, the error in the location is small in the sum-of-gaussian methods, but increases rapidly in the product of gaussian methods (W2, W4 and W6). Finally, the user can select if the localization will be local o global.

Once all the parameters have been selected, the student will press the 'Start' button to begin the localization process. Along the process, the students can see the Monte Carlo algorithm performance. The user can observe the simulated odometry (yellow line in fig. 3), the real path (red line in fig. 3), the particle movement (blue points in fig. 3), associated the landmarks (green circles in fig. 3) and the center of the particle population (pink circle in fig. 3). In addition the student will see the error in the location and dispersion of particles for each step or movement (fig. 6). Finally, the user can pause the process by pressing the 'Stop' button and later continue with it by pressing the 'Continue' button, or reset the process by clicking on the 'Reset' button.

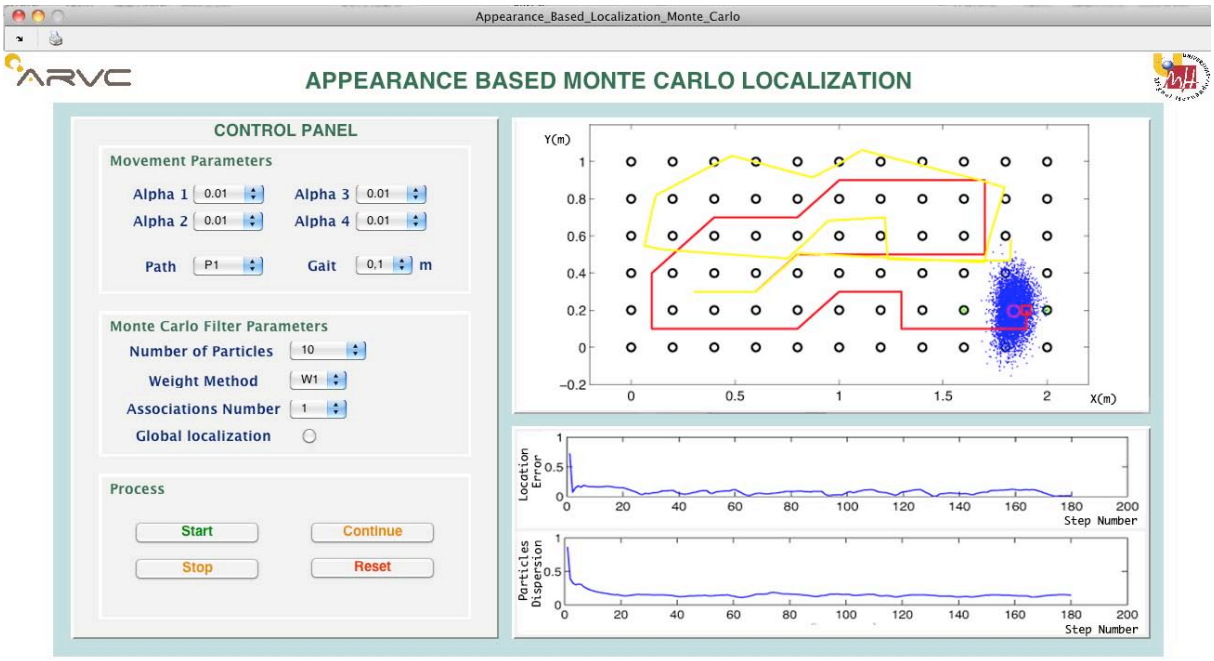

Fig. 3. Appearance of the graphical interface of the application

Fig. 4. (a) Path the user can choose and (b) distance for each movement the user can choose to do the experiment

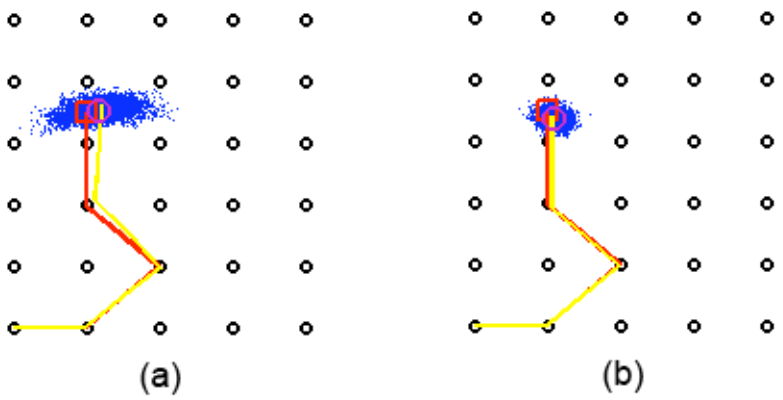

Fig. 5. Two examples of an odometry simulation for (a) non-homogeneous alpha parameters and (b) homogeneous alpha parameters

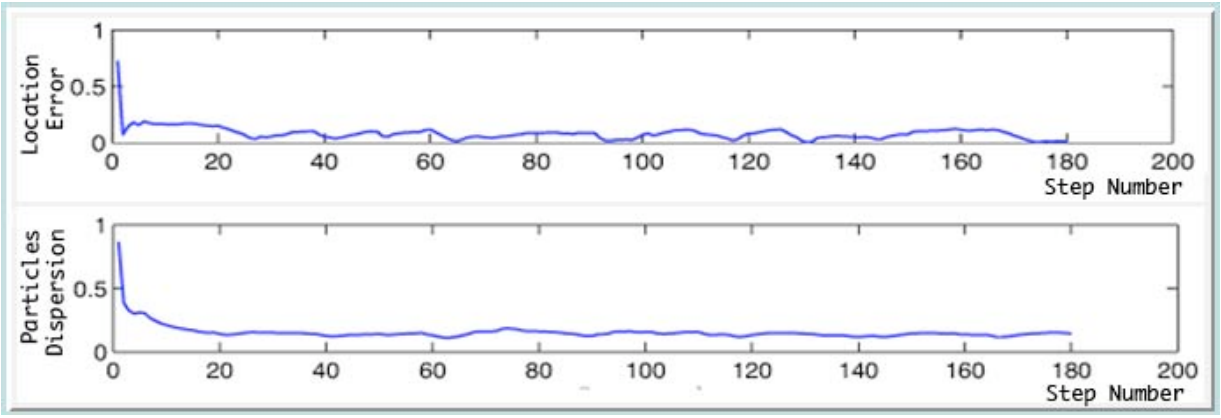

Fig. 6. Example of location error and particles dispersion graph

7 CONCLUSION

This work presents a software tool we have implemented to be used in a robotics and computer vision subject in the doctorate studies. With this tool, the students can fully understand the appearance-based approach in robotics Monte Carlo localization, with the next features.

1. We have included a database with panoramic images (grey-scale) of an environment. This database is used to build the map of the environment and to make the different types of possible paths.
2. The student can select the value of the errors to simulate the robot's odometry (alpha1, alpha2, alpha3, alpha4).
3. The user is allowed to choose among 6 different paths and he can choose the distance travelled for each movement (gait).
4. It is possible to select the number of the particles to do the experiment and the weight method employed to resample the particles. In addition the student can choose the number of associated landmark.
5. Moreover, the user can decide to carry out a global localization (unknown initial pose) or a local localization (known initial pose).
6. The tool is fully interactive. Once the localization process is finished, several graphical representations of the data can be showed to know the degree of accuracy of the method used. This allows the students to perform a comparative study of the weight methods.

This tool has demonstrated to be very useful for the students to fully understand the MCL appearance-based method and other basic computer vision concepts. The students distinguish the different parameters to be configured so that the method works correctly. They learn the odometry problems when the robot moves around the environment and how it is possible to solve these problem by means of a Monte Carlo algorithm. Once the practical sessions have been completed, the student is capable to develop more complex algorithms to control the movements of a robot using this approach.

Acknowledgements

This work has been supported by the Spanish government through the project DPI2010-15308. 'Exploración integrada de entornos mediante robots cooperativos para la creacioón de mapas 3D visuales y topológicos que puedan ser usados en navegación con 6 grados de libertad'.

REFERENCES

[1] Dellaert, F., Burgard, W., Fox, D., Trun, S. Using the conden- sation algorithm for robust, vision-based mobile robot localization. IEEE Computer Society Conference on Computer VIsion and Pattern Recognition (CVPR99), (1999).

[2] Gil, A., Reinoso, O., Vicente, M. A., Fernández, C., Payá. L. Monte carlo localization using SIFT features. Lecture Notes in Computer Science, I(3523), (2005) 623–630.

[3] Trun, S., Fox, D., Burgard, W., Dellaert, F. Robust monte carlo localization for mobile robots. Artificial Intelligence, vol 128(1-2), (2000) 99–141.

[4] Murillo, A. C., Guerrero, J. J., Sagüés, C. Surf features for efficient robot localization with omnidirectional images. In Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), San Diego, CA, USA, (2007).

[5] Bay, H., Tuytelaars, T., Van Gool, L. Surf: Speeded up robust features. In Proceedings of the ninth European Conference on Computer Vision, (2006).

[6] Menegatti, E., Maeda, T., Ishiguro, H. Image-based memory for robot navigation using properties of omnidirectional images. Robotics and Autonomous Systems, 47(4), (2004) 251–276.

[7] Jogan, M., Leonardis, A. Robust localization using eigenspace of spinning-images. In Proc. of the IEEE Workshop on Omnidirectional Vision, Hilton Head Island, USA, (2000) 37–44.

[8] Fernández, L., Gil, A., Payá, L., Reinoso, O. An Evaluation of Weighting Methods for Appearance-based Monte Carlo Localization using Omnidirectional Images. In IEEE International Conference on Robotics and Automation (ICRA, 2010 Workshop on Omnidirectional Robot Vision), Alaska, (2010).

[9] Menegatti, E., Zocaratto, M., Pagello, E., Ishiguro, H. Image-based monte carlo localisation with omnidirectional images. Robotics and Autonomous Systems, 48(1), (2004) 17–30.

[10] Payá, L., Fernández, L., Reinoso, O., Gil, A., Úbeda, D. Appearance- based dense maps creation. comparison of compression techniques with panoramic images. In Proc. of the Int. Conf. on Informatics in Control, Automation and Robotics, Milan, Italy, (2009) 238–246.

[11] Rossi, F., Ranganathan, A., Dellaert, F., Menegatti, E. Toward topological localization with spherical fourier transform and uncali- brated camera. In Proc. of the Int. Conf. on Simulation, Modeling and Programming for Autonomous Robots, Venice, Italy, (2008) 319–330.

[12] http://www.mathworks.com/products/matlab