Continuous Probabilistic Mapping by Autonomous Robots

Jesús Salido Tercero[1], Christiaan J. J. Paredis[2], and Pradeep K. Khosla[2]

[1] Universidad de Castilla-La Mancha, ES de Informática,
Ronda de Calatrava 5, 13071 Ciudad Real, Spain
[2] Carnegie Mellon University,
Institute for Complex Engineered Systems,
5000 Forbes Avenue, Pittsburgh, PA 15213, USA

Abstract. In this paper, we present a new approach for continuous probabilistic mapping. The objective is to build metric maps of unknown environments through cooperation between multiple autonomous mobile robots. The approach is based on a Bayesian update rule that can be used to integrate the range sensing data coming from multiple sensors on multiple robots. In addition, the algorithm is fast and computationally inexpensive so that it can be implemented on small robots with limited computation resources. The paper describes the algorithm and illustrates it with experiments in simulation and on real robots.

1 Introduction

For efficient navigation in cluttered environments, maps are an indispensable tool. Maps allow mobile robots to autonomously plan trajectories through the environment while avoiding obstacles. Maps are often pre-recorded or generated by human operators. However, in unknown or rapidly changing environments, these maps are usually unavailable or outdated. In order to maintain autonomous operation in these situations, robots should record their own experiences and build internal maps themselves.

In recent research results, two different approaches can be identified for representing the environment:

1. *Metric or grid-based representations* [5][9]. In this approach, the world is divided into evenly spaced cells in a grid. Each cell contains information related to the corresponding region in the environment.
2. *Topological representations* [4] Here the world is represented as a graph in which nodes represent states, places, or landmarks. If there is a direct path between two nodes, they are connected with an arc.

Both approaches have advantages and disadvantages. Metric maps are easy to build and maintain, but they require accurate measurements of the robot position. Furthermore, building metric maps is in general time-consuming and may require a large amount of storage space. Topological maps, on the other hand, are much more difficult to build but do not depend as much on accurate position measurements. In addition, their compactness makes them highly suitable for fast global planning. Overall, metric maps have been used most often for sensor-based map-building.

In this paper, we present a novel approach to building grid based metric maps. Its novelty resides in its generality and simplicity. It requires only limited computational resources and can be easily implemented in real-time. It allows multiple sensing modalities to be combined (sonar, laser-range-finder, camera, etc.) and it can combine maps from multiple robots into a single global map.

2 Previous related work

As we mentioned in the introduction, metric maps have been most successfully applied until now. In particular, *grid based occupancy maps* [5][9][10] have shown to be convenient and useful. In an occupancy map, the world is represented by a grid in which each cell contains the probability of occupancy; a value of zero corresponds to free space, a value of one corresponds to an obstacle.[1]

In grid based maps, Bayes' rule [11] is used to update occupancy values over time. The updated occupancy value *(a posteriori probability)* is based on the occupancy value in the current map *(a priori probability)* and the current sensor reading *(conditional probability)*. The update process requires a sensor model defining the occupancy values of the grid cells around the sensor, given the current sensor reading. Several solutions have been proposed for the sensor modeling problem:

1. **Probabilistic models**. A probability density function is used to model the sensor. Commonly, Gaussian probability density functions are used. [5][10].
2. **Histograms**. This approach is usually applied for *local planning* (i.e. obstacle avoidance), for which a fast response is critical. To avoid the time consuming evaluation of a probability density functions, a simple histogram is used. [2].
3. **Learned models**. Instead of using a Gaussian approximation, the probability density function is learned [15]. This provides greater accuracy but at a the cost of greater computational requirements at run-time as well as for training. A neural network trained in simulation has been used to determine the probability density function [15].

The type of sensor model has a significant impact on the applicability of the method. For applications in which very accurate global plans need to be constructed and computational resources are readily available, the most accurate sensor model is appropriate, namely a learned model, even if this requires a high computation cost. On the other hand, if information for local obstacle avoidance is the goal of the algorithm, a very simple sensor model can be used with adequate results. Because such a simple model can be evaluated very quickly, a high sample rate can be achieved, which allows the robot to move more quickly while still avoiding collisions.

The method that we propose in this article, makes a compromise between these two extremes. It uses a sensor model that is slightly more expensive to evaluate than in the Histogramic In-Motion Mapping approach [2]. Yet, the method provides maps that are sufficiently accurate to perform global planning, at a fraction of the cost of learned models.

The whole map-building process consists of the following three steps:

[1] For visual purposes that likelihood is usually represented by gray levels where 'black' means a completely occupied cell and 'white', free space.

1. Sensing.
2. Computation of conditional occupancy probabilities based on the sensor model.
3. Update the occupancy values in the grid using Bayes' rule.

An important requirement for building metric maps is the accurate determination for the robot position and orientation. That does not pose a big problem for outdoor applications where global positioning systems (GPS and compass) are available. However, for indoor applications or space applications these global positioning systems are usually not available.

Early results used dead-reckoning and filtering to locate the robot in its environment [1][5][10] Recently, a new set of probabilistic approaches has emerged based on *Markov Decision Processes* [13]. Instead of using a single position estimate, a probability distribution is used to represent the robot's belief of being in a particular position/orientation. This method requires an 'a priori' map to be supplied to the robot, which may cause some undesired cyclic behavior. Moreover, the basic assumptions for a Markov process are not satisfied in highly dynamic environments. To overcome these problems, researchers have introduced modifications to the basic MDP, such as the *(Partially Observable Markov Decision Process)* [3] and the work in [7]. Instead of combining user supplied maps with sensor readings, recent work at Carnegie Mellon University has focussed on combining map-based localization and autonomous map-building [6][14][15].

As part of the CyberScout project, we are also developing a sonar-based indoor localization system. In this system, each robot is equipped with a sonar beacon and an RF transmitter, as illustrated in Figure 1. A conical reflector is mounted above the sonar transceiver to reflect the sound into a horizontal plane and provide 360° coverage. Periodically, each robot will transmit a sonar and an RF pulse simultaneously. All the other robots in the vicinity will receive the RF pulse almost instantaneously and the sonar pulse after a delay proportional to the distance from the transmitting robot. When the distances between all the robots are known, one can use trilateralization to determine their relative positions. This localization system will allow us in the future to implement the mapping algorithm described in this article on the small robot platforms shown in Figure 1.

3 Continuous probabilistic mapping — CPM

This section describes the details of our approach for building metric maps called Continuous Probabilistic Mapping (CPM). By *continuous* we emphasize the fact that CPM allows one to update the map continuously—every execution cycle. The approach requires only limited computation each time-step and can easily be implemented on the small robot platforms with limited computational resources that we use in the lab. The applications that we consider are cooperative exploration and reconnaissance by a team of autonomous robots in unknown and possibly dynamic environments.

In CPM, the environment is divided into uniform cells $c_{i,j}$ (with center at $[x_i, y_j]$). We define a metric Occup(c) that defines the occupancy likelihood of the cell c:

In the further derivation of the CPM algorithm, we assume that the robot's position (x, y, θ) can be measured sufficiently accurately.

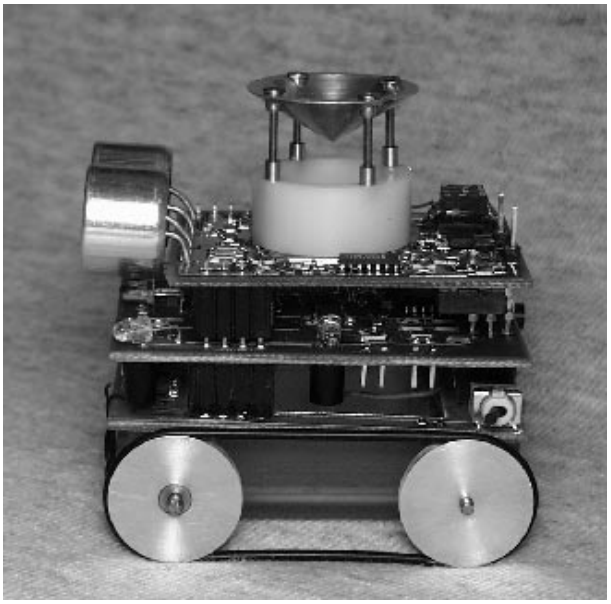

Fig. 1. A Millibot (size $6 \times 6 \times 6$cm) with a sonar-beacon.

Table 1. Occupancy metric

Value	Description		
0.0	minimum occupancy	free space	no uncertainty
0.5	medium occupancy	unknown	maximum uncertainty
1.0	maximum occupancy	obstacle	no uncertainty

We further assume that the sensor readings consist of proximity measurements (e.g. sonar or laser-range-finder) and that the characteristics of the sensor are well understood (e.g. field of view and maximum range).

3.1 The model for 'a priory' occupancy likelihood

CPM uses a simplified 'a priori' occupancy likelihood based on the concept of *field of view* (FOV) of a sensor. By FOV we mean the *spatial region where a sensor is able to perceive*. As is illustrated in Figure 2, the FOV can be characterized by two parameters: the maximum range (r), and the perception angle (α). For example, for the sonar used in our lab (Polaroid 6500 series) the maximum range is 2 meters and the perception angle is 20 degrees.

Based on the FOV, we define a function $Occup_0(s, c_{i,j})$ for each cell $c_{i,j}$ around the sensor s. If a cell c covers only a fraction f of the FOV of the sensor s, the function $Occup_0$ is proportional to $(1 - f)$ or

$$Occup_0(s, c_{i,j}) = \frac{1}{2}\left(1 - \frac{\text{Area}(FOV \cap c)}{\text{Area}(c)}\right) \tag{1}$$

One can interpret this function as the probability that a cell is occupied when no obstacles are detected. As a result, any cell that does not overlap with the FOV of the sensor is assigned a value of 0.5 meaning that the occupancy is completely unknown.

For the further development of the algorithm, we only need to consider the cells that overlap with the FOV. As indicated in Figure 2, we call this set of cells the *Sensor Occupancy Pattern* or SOP. The $Occup_0(s, c_{i,j})$ values for the SOP are independent of the sensor readings and can be computed beforehand. An SOP requires little storage and can be applied to any type of range sensor.

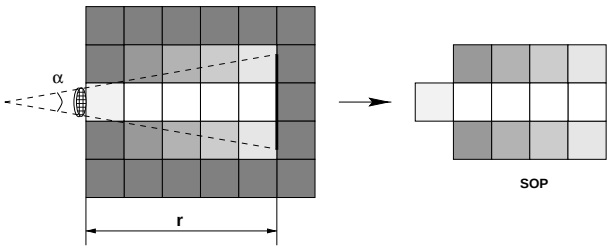

Fig. 2. The a priori sensor occupancy values (r=range, α=perception angle), and the Sensor Occupancy Pattern (SOP)

3.2 Computation of the occupancy function

In this section, we will explain how the SOP is combined with a sensor reading to obtain the occupancy value for each cell in the FOV. These values will later be combined with the occupancy probabilities in the global map to update the robot's world view.

The computation of the occupancy function consists of two steps. In a first step, we classify the cells in the field of view into three categories, depending on whether they are within the range of the current sensor reading or not. Next, we combine this information with the values in the SOP to obtain the final occupancy value of sensor.

The classification process divides the cells in the FOV into three categories: *occupied, free or unknown*. Consider the example in Figure 3. Each cell is a square of dimension C_s. The distance $d = D_c - S$ is the difference between the sensor reading S and the distance to the center of the cell D_c. The classification function $h(c, S)$ is than computed as:

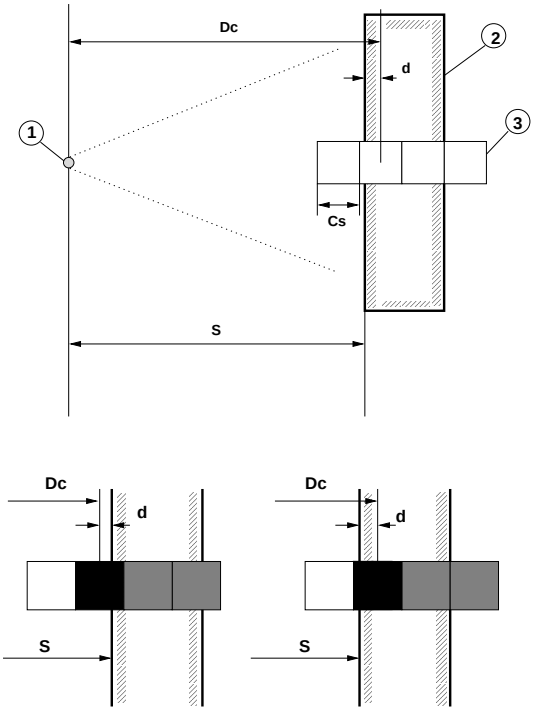

Fig. 3. Classification of cells: (1) sensor, (2) obstacle, and (3) cell

$$h(c,S) = \begin{cases} \textbf{Occupied}, & \text{if} \quad \mid d \mid < Cs/2 \\ \textbf{Free}, & \text{if} \quad S = r \quad \vee \quad (\mid d \mid \geq Cs/2 \wedge d < 0) \\ \textbf{Uknown}, & \text{if} \quad \mid d \mid \geq Cs/2 \quad \wedge \quad d \geq 0 \end{cases} \tag{2}$$

The classification is conservative in that it overestimates the probability of occupation. From the range measurement, we know that somewhere at a distance r there is an obstacle in the FOV. However we do not know in which cell exactly. Therefore, we label every cell at distance r as occupied, even though it is not very likely that all these cells are actually occupied. Our conservative approach still provides accurate results in the long term because unoccupied cells that were wrongly labeled as occupied based on one sensor reading will most likely be labeled correctly when the robot changes positions.

In the classification, we do not take into account the extent to which a cell is inside the FOV of the sensor. Therefore, the classification value is now combined with the a priori occupancy to obtain the actual occupancy value $\text{Occup}(c,S)$ of each cell:

$$\text{Occup}(c,S) = \begin{cases} Occup_0 & \text{if } h(c,S) = \textbf{Free} \\ 0.5 & \text{if } h(c,S) = \textbf{Unknown} \\ 1 - Occup_0 & \text{if } h(c,S) = \textbf{Occupied} \end{cases} \tag{3}$$

It is clear that the computation of the occupancy values requires minimal computation. The total number of operations needed is proportional to the total number of cells in the SOPs of all sensors. Compared to other occupancy grid methods, the computations per cell are much simpler and the number of cells is much reduced by only considering the cells in the FOV. In the next section, these occupancy values will be used to update the global map of the environment using a Bayes' update rule.

3.3 Spatial-temporal integration

So far, we have illustrated how to obtain an approximate probability distribution for a single range sensor based on a single sensor reading. However, mapping is a dynamic process in which sensor readings from different locations and at different time instants are fused into a single global representation. Bayesian probability theory has been applied successfully to the mapping problem [5][10][9][15]. In this section, we describe how Bayes' rule is applied in our approach.

After having computed the function $\text{Occup}(c)$ for each of the sensors, there are two steps remaining. In a first step, the cells in the grid around each sensor are mapped to cells in the global map using the position and orientation information from the robot. Then, in a second step, the occupancy value of each sensor cell is used to update the estimated occupancy of the corresponding cell in the global map. Because the robot moves, over time cells in the global map are covered multiple times by different sensors from different sensing locations and most likely.

Using Bayes' rule, all these sensor readings are integrated into the global map, improving the estimates for the occupancy of each cell. This integration is accomplished using the following equation [15]:

$$\text{Occup}(c \mid S^{(1)}, S^{(2)} \cdots S^{(T)}) = 1 - \left(1 + \prod_{\tau=1}^{T} \frac{\text{Occup}(c \mid S^{(\tau)})}{1 - \text{Occup}(c \mid S^{(\tau)})}\right)^{-1} \tag{4}$$

This equation expresses the a posteriori probability of occupation of cell c given a sequence of readings $(S^{(1)}, S^{(2)} \cdots S^{(T)})$. In the equation, we assume an initial occupancy value of $\text{Occup}(c) = 0.5$ for all the cells. The sequence of readings may include readings at different time instants as well as readings from different sensing locations, different sensors on the same robot, or sensors on multiple robots.

Notice that Equation 4 can also be evaluated iteratively, which reduces the store requirements significantly; only one value per cell needs to be stored.

$$\text{Occup}(c \mid S^{(1)}, S^{(2)} \cdots S^{(T)}) = 1 - \left(1 + \frac{\text{Occup}(c \mid S^{(T)})}{1 - \text{Occup}(c \mid S^{(T)})} \frac{\text{Occup}(c \mid S^{(1)} \cdots S^{(T-1)})}{1 - \text{Occup}(c \mid S^{(1)} \cdots S^{(T-1)})}\right)^{-1} \tag{5}$$

Finally, the same equation can also be used to integrate maps from two different robots. One can write Equation 4 as

$$\text{Occup}(c \mid S^{(1)}, S^{(2)} \cdots S^{(N)}) = 1 - \left(1 + \frac{\text{Occup}(c \mid S^{(1)} \cdots S^{(k)})}{1 - \text{Occup}(c \mid S^{(1)} \cdots S^{(k)})} \frac{\text{Occup}(c \mid S^{(k+1)} \cdots S^{(N)})}{1 - \text{Occup}(c \mid S^{(k+1)} \cdots S^{(N)})}\right)^{-1} \quad (6)$$

This allows one to combine maps from different robots (sensor readings 1 through k from robot 1; sensor readings $k+1$ through N from robot 2).

Implementational considerations. The main purpose for the integration process discussed in this section is to produce a map in a global frame of reference. However, the sensor occupancy values are obtained in the robot's frame of reference. Although the local cells can be mapped to the global frame of reference by applying a translation and rotation transformation,the cells do not line up with the cells of the global map in general. This problem is solved by using a simple interpolation scheme that determines the sensor occupancy value at each of the centers of the cells in the global map.

A second important implementational issues has to do with dynamically changing environments. When a cell in the global map is occupied by an obstacle, the occupancy value of that cell will move closer and closer to one as more and more sensor reading are incorporated. Due to the finite precision of floating point computation, the occupancy value may actually become equal to one at some point in time. However, this causes problems in the evaluation of Equation 5. Even with a careful implementation of Equation 5, there still remains a problem with dynamic environments. Once, a cell reaches a value of either zero or one, the value cannot change anymore, regardless of the current sensor readings. Therefore, in dynamic environments, where the position of obstacles may change over time, the occupancy value should never be allowed to reach zero or one completely. In our implementation we limit the occupancy value to remain inside the interval $[10^{-6}, 1 - 10^{-6}]$.

4 Experiments and results

We performed experiments in simulation as well as with real robots. All the experiments were executed using the CyberRAVE environment developed at Carnegie Mellon University [8] [12]. CyberRAVE is a general purpose framework for simulation and command and control of multiple robots. It allows algorithms developed in a simulated environment to be transfered transparently to the real robot, and even allows real and simulated robots to interact with each other in a single experiment. CyberRAVE allows us to quickly develop algorithms for multi-robot systems, evaluate different sensor configurations in simulation, and transition the software to real hardware one robot at a time.

4.1 Experiments in simulated environments

The main purpose for the simulation experiments is to determine how well CPM performs in terms of real-time computation and its ability to integrate maps in a multi-robot system.

The experiment consists of a collaborative mapping task with at team of five robots. The simulated robots are modeled after the RC-tank based robots developed in our lab (Figure 5). They are equipped with 8 sonars: 5 in the front, 1 left, 1 right, and 1 in the back. The sonars are modeled to have a sensing cone with $r = 2$m. and $\alpha = 20^{\circ}$). The environment the robots need to map consists of a flat terrain with polygonal obstacles.

Figure 4 illustrates some of the results. Initially, the robots start without any knowledge of the environment: the complete map is initialized with occupancy values of 0.5. While the user maneuvers the robots around using a simulated joystick, the robots build up information about the environment through their sonar sensors. This results in occupancy values moving towards one (black) in cells containing obstacles, and toward zero (white) in free space. Each robot builds up a local map containing occupancy information for the area it has traversed. Periodically, the local maps for each of the robots is transmitted to a "Team Leader" who uses CPM to integrate the local maps into a single global map.

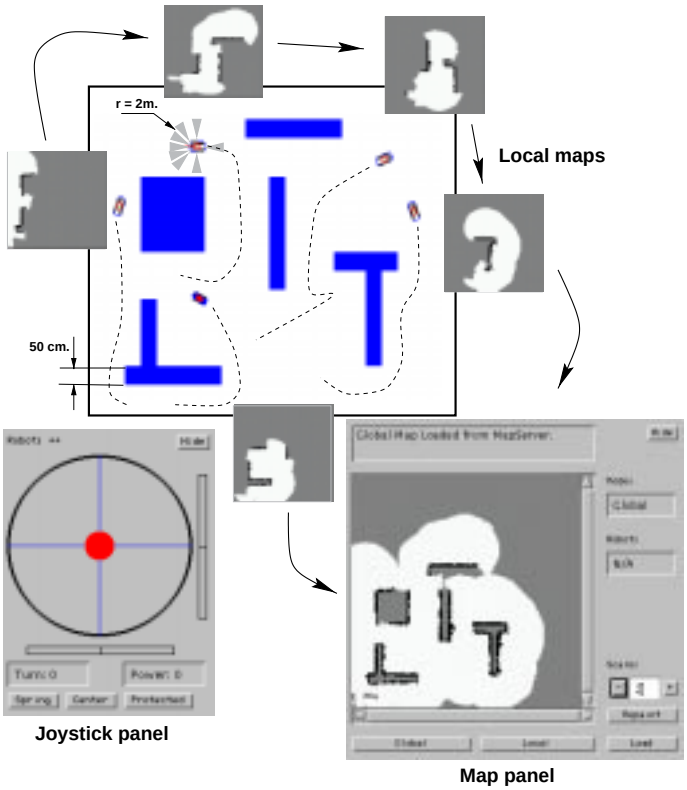

Fig. 4. Cooperative mapping results in simulation using CyberRAVE

4.2 Experiments with real robots

To perform mapping experiments with real robots, we have equipped our lab with a vision-based localization system, as is illustrated in Figure 6. The CPM algorithm requires that the global position of the robot be known. Since GPS sensors cannot be used inside, we use two overhead cameras and a cognachrome tracking system to measure the position and orientation of the robots with an accuracy of about 5 cm and 5 degrees. Relative to the size of the robots (30x50cm), the tracking system provides sufficient accuracy to build maps collaboratively.

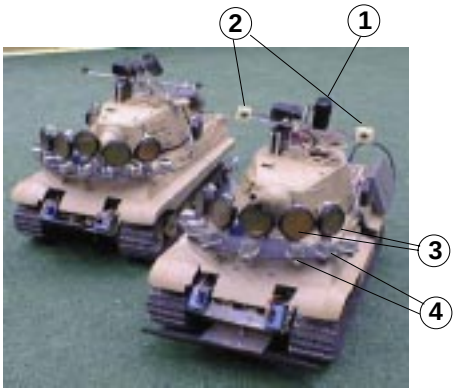

Fig. 5. Scouts: (1) B/W camera, (2) microphones, (3) sonars, and (4) IRs.

The robots shown in Figure 5 are small RC-tanks that have been retrofitted with a PC-104 stack containing a 486 computer running Linux OS, a digital IO board for sensor and actuator interaction, a wireless Ethernet connection, and a hard disk. They are further equipped with 8 sonar sensors, 5 IR obstacle detectors, a B/W camera mounted on a pan/tilt mechanism, and stereo microphones. For the mapping experiment, we only use the 8 Polaroid sonars (series 6500) with a sampling rate of about 5Hz. In their current configuration the sonars have a reliable sensing range of 0.50m to 2.0m with a precision of $\pm$0.02m.

The area in which the experiments are conducted is delimited by fiberboard panels. When the robot is at an oblique angle with respect to the paneling, the sonar pulse may not reflect back to the robot directly but bounce back via one of the other four panels. This multi-path effect may erroneously cause the robot to sense obstacles further than they really are or even stop the robot from sensing the obstacle at all. However, from a mapping perspective, infrequent erroneous measurements pose no problem. Over time, only the correct measurements contain enough correlated information to show up in the map.

Figure 7 shows the results obtained in a mapping experiment compared to the actual obstacle layout. The quality of the map is not only determined by the size of the grid cells, but also by the trajectory followed by the robot and the the number of multi-path readings. In this experiment, the robot moved through the environment relatively quickly and experienced a several number of multi-path readings. The result is that the map marks the inside of the obstacles and even areas outside the

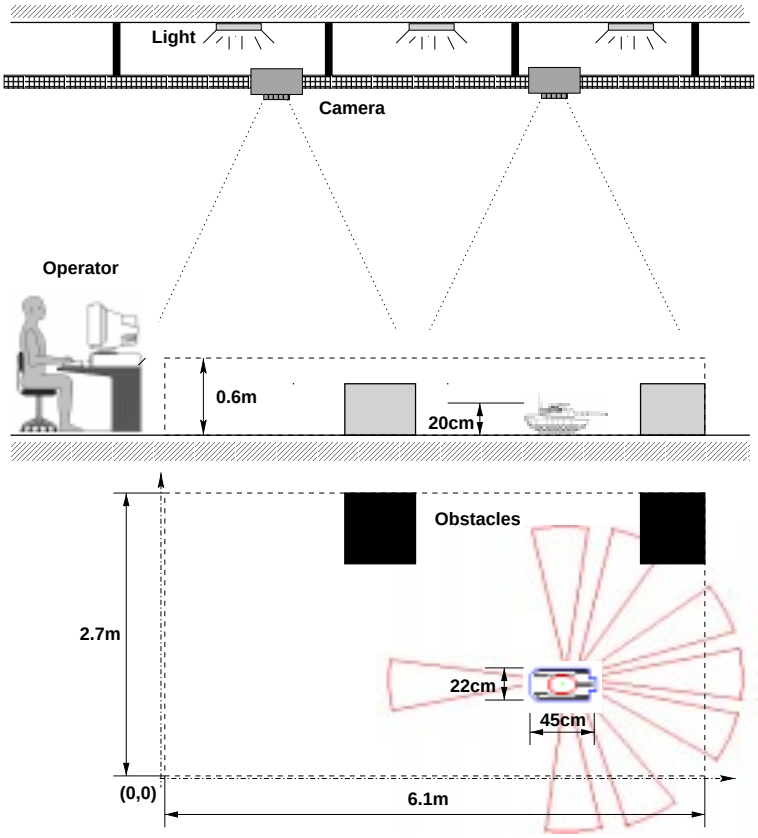

Fig. 6. The physical layout of the testing area.

boundary of the experimental area as unoccupied. The obstacles on the right side of the map are not marked, because the area has not yet been explored by the robot.

The second and third experiment were produced with 10×10cm cells, and b

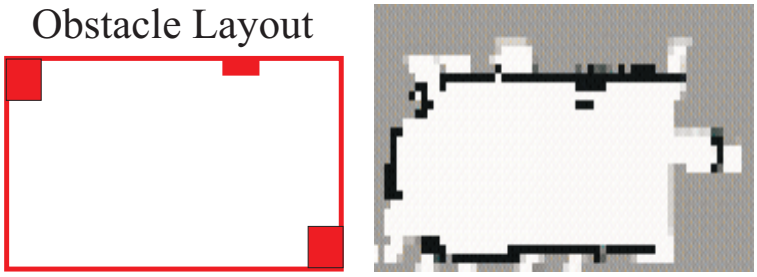

Fig. 7. Mapping result with a real robot

5 Summary

In this paper, we have described a novel method for continuous mapping in unknown environments: Continuous Probabilistic Mapping. The novelty of CPM resides in two properties: simplicity for real-time computation, and scalability for multiple robots each with multiple sensors.

The algorithm requires only limited computational resources so that it can be executed at a high frequency, which allows the robot to move quickly while still avoiding obstacles. This simplicity is also important for our future work in small distributed robotic systems in which computation is at a premium.

With respect to distributed robotics, the scalability of the algorithm is also very important. The algorithm allows robots to integrate their sensing data locally for collision avoidance, but at the same time to combine this local information with that of other robots to obtain a global view of the environment.

Acknowledgement

This research was supported in part by DARPA/ETO under contract DABT63-97-1-0003, and by the Institute for Complex Engineered Systems at Carnegie Mellon University.

References

1. J. Borenstein, H. R. Everett, and L. Feng. *Navigating Mobile Robots: Sensors and Techniques*. A. K. Peters, Ltd., first edition, 1996.
2. J. Borenstein and Y. Koren. Histogramic in-motion mapping for mobile robot abstacle avoidance. *IEEE Trans. on Robotics and Automation*, 7(4):535–539, 1991.
3. Anthony R. Cassandra, L. Kaelbling, and J. Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *IEEE/IROS Int. Conf. on Intelligent Robotics and Systems*, 1996.
4. H. Choset. *Sensor based motion planning: The hierarchical generalized Voronoi graph*. PhD thesis, California Institute of Technology, USA, 1996.
5. Alberto Elfes. *Occupancy grids: A probabilistic framework for mobile robot perception and navigation*. PhD thesis, Electrical and Computer Engineering Dept., Carnegie Mellon Univ., 1989.
6. W. Burgard et al. The interactive museum tour-guide robot. In *AAAI, Nat. Conf. on Artificial Intelligence*, 1998.
7. Dieter Fox, W. Burgard, S. Thrun, and A. Cremers. Position estimation for mobile robots in dynamic environments. In *AAAI, Nat. Conf. on Artificial Intelligence*, 1998.
8. Wes Huang et al. CyberRAVE: A real and virtual environment for CyberScout, August 1998.
9. Martin C. Martin and Hans P. Moravec. Robot evidence grids. Technical Report CMU-RI-TR-96-06, Carnegie Mellon University, March 1996.
10. H.P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, Summer:116–121, 1988.

11. Athanosios Papoulis. *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, third edition, 1991.
12. J. Salido, J.M. Dolan, J. Hampshire, and P. Khosla. A modified reactive control framework for cooperative mobile robots. In *SPIE International Symposium on Sensor Fusion and Decentralized Control in Autonomous Robotic Systems*, volume 3209, pages 90–100, October 1997.
13. Reid Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *International Joint Conference on Artificial Intelligence*, pages 1080–1087, August 1995.
14. S. Thrun, D. Fox, and W. Burgard. Probabilistic mapping of a environment by a mobile robot. In *AAAI, Nat. Conf. on Artificial Intelligence*, 1998.
15. Sebastian Thrun. Learning maps for indoor mobile robot navigation. *AI Magazine*, 1997.