ELSEVIER

Robotics and Autonomous Systems 40 (2002) 25–42

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

# Exploration method using harmonic functions

Edson Prestes e Silva Jr.[a], Paulo M. Engel[a], Marcelo Trevisan[b], Marco A.P. Idiart[b,*]

[a] *Instituto de Informática, Universidade Federal do Rio Grande do Sul, 91501-970 Porto Alegre, RS, Brazil*

[b] *Instituto de Física, Universidade Federal do Rio Grande do Sul, 91501-970 Porto Alegre, RS, Brazil*

Received 2 February 2001; received in revised form 6 February 2002

Communicated by T.C. Henderson

**Abstract**

Harmonic functions provide optimal potential maps for robot navigation in a previously explored static environment. Here we investigate the performance of an algorithm for exploration based on partial updates of a harmonic potential in an occupancy grid. We consider that while the robot moves it carries along an activation window whose size is of the order of the sensor's range. The activation window recruits grid points to participate in the potential calculation. By using simulations and experiments with the Nomad 200 robot we investigate the algorithm performance in respect to parameters such as the frequency of updates and the numerical method used to calculate the harmonic potential. © 2002 Elsevier Science B.V. All rights reserved.

*Keywords:* Mapping; Navigation; Mobile robots; Harmonic functions

## 1. Introduction

The process of building a stable and reliable representation of external world is made through a long process of interaction of the robot with the environment. The robot uses its sensors to sense the world and, by adding new information or replacing incorrect knowledge, it learns strategies to explore and exploit it.

Here we introduce a robust principle that allows a complete coverage of a predefined fraction of the state space. Using this principle we discuss the problem of designing a controller that guides a mobile robot to explore an unknown environment in order to produce an accurate map of its bearings.

For many tasks exploration and exploitation establish a trade-off. But here, since exploration itself is the task, the *exploration* $\times$ *exploitation dilemma* no longer applies. Exploitation facilitates exploration.

Our approach is based on the use of a potential field to guide exploration. It combines a path planning method that uses a special type of potential field—the harmonic functions—developed by Connolly and Grupen [1], a modified version of the mapping process proposed by Borenstein and Koren [2], and the notion of frontier-based exploration of Yamauchi [3].

The potential indicates the probability that the robot hits a previously mapped obstacle before reaching the unexplored region when engaging in random exploration [4]. Following greedily the gradient descent on the potential, and therefore minimizing the hitting probability, makes the robot to maximize knowledge gain while avoiding obstacles. This nice interplay of exploration and exploitation is an instance of *directed exploration* [5].

[*] Corresponding author.

*E-mail addresses:* prestes@inf.ufrgs.br (E. Prestes e Silva Jr.), engel@inf.ufrgs.br (P.M. Engel), tmarcelo@if.ufrgs.br (M. Trevisan), idiart@if.ufrgs.br (M.A.P. Idiart).

Our approach has the following features:

- Due to the use of harmonic functions it does not generate local minima that could trap the robot during exploration.
- The robot moves toward the closest unvisited space optimizing exploration.
- It integrates local and global information generating smooth trajectories.
- It does not use explicit depth-first or breadth-first search.
- It can be goal-driven or not. If it is goal-driven, the robot explores the environment until the goal be found. When the robot finds a goal, it knows how to reach the goal from the explored space; otherwise, the environment will be explored completely.

This paper is organized as follows. In Section 2 we formally introduce the problem of navigation. In Section 3 we present a picture of the state-of-art in robotic navigation. There we focus on the issues of internal representations of the world, exploration schemes and path planning methods. Section 4 is dedicated to introduce our method for exploratory navigation. In Section 5 we explain its implementation. The results are presented in Section 6 and they are discussed in Section 7.

## 2. Theoretical aspects on robotic navigation

Generically, a controller is an algorithm that has as input the temporal sequence of sensory readings detected by the robot in the near past represented by

$$\mathbf{S}_n(t) = (\mathbf{s}(t-(n-1)\Delta t), \mathbf{s}(t-(n-2)\Delta t), \ldots, \mathbf{s}(t)),$$

as well as the sequence of action taken

$$\mathbf{A}_n(t) = (\mathbf{a}(t-n\Delta t), \mathbf{a}(t-(n-1)\Delta t), \ldots, \mathbf{a}(t-\Delta t)),$$

where $1/\Delta t$ is the data acquisition rate. The states $\mathbf{s}(t)$ are instantaneous sensory states described by data coming from a variety of built-in sensors. The output is the action $\mathbf{a}(t)$ necessary to perform the transition to the next state $\mathbf{s}(t+\Delta t)$. Both $\mathbf{s}(t)$ and $\mathbf{a}(t)$ are multidimensional vectors whose dimensions are related to the number of variables involved in the input and output. The adaptative vector-valued function

$$\mathcal{F}[\mathbf{\Gamma}(t)] : (\mathbf{S}_n(t), \mathbf{A}_n(t)) \to \mathbf{a}(t),$$

that maps the input on the output is the core of the controller. The internal parameters $\mathbf{\Gamma}(t)$ constitute the model of the environment built by the robot from its experience.

For an external observer a given sensory state $\mathbf{s}(t)$ corresponds to a physical state $\mathbf{r}(t)$, like position in the space or configuration of joints, in the case of an arm. The controller has to decode $\mathbf{r}(t)$ from the instantaneous sensory data $\mathbf{s}(t)$, or from the extended time data $(\mathbf{S}_n(t), \mathbf{A}_n(t))$. This is called the *localization problem* and its solution is one of the challenges in robotic navigation. The difficulty in decoding $\mathbf{r}(t)$ comes from two sources: First, sensory readings can be ambiguous and noisy. For instance, sonar reflections can hide obstacles or indicate wrong distances. Second, the actions performed by the effectors not always have the predicted outcome. A typical case are the odometric errors in wheeled robots. Slippage and refraction like effects are prone to occur when the robot crosses the boundary between two different surfaces.

The main approaches developed in the literature to deal with the localization problem in a generic and robust form are based on optimal linear estimators—the Kalman filters and extended Kalman filters [6] —and full probabilistic approaches that use probabilistic models for sensors and effectors and the Bayes rule to write the expected localization probabilities in terms of the models [7].

The *mapping problem* consists of putting together in a coherent representation the features of the environment detected by the sensors. Mapping and localization are closely related problems. They share the same theoretical difficulties. In fact, the techniques described above can be used to allow concurrent localization and mapping [6,8]. Besides localization an important issue in mapping is how the data is organized. Choosing the right representation is important to guarantee a better integration between different components of the controller, an easier maintenance of the map, and the improvement of a series of other factors that may result in economy of computational resources.

The third major challenge in robotic navigation is the *path planning problem*. It can be divided in two sub-problems: global planning and local planning. Global planning is an optimization process that aims to bring the robot from its current position $\mathbf{r}_0$ to a goal position $\mathbf{r}_\mathrm{G}$ through a series of actions $\mathbf{A}^{\mathrm{op}} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_p)$. The set $\mathbf{A}^{\mathrm{op}}$ represents an optimal path that connects $\mathbf{r}_0$ to $\mathbf{r}_\mathrm{G}$ with a compromise between travel time and safety. The path is calculated using the global features represented in the environment map.

The local planner is responsible for adjustments in $\mathbf{A}^{\mathrm{op}}$ to account for unexpected events encountered during its realization. These events can be unmodeled obstacles that were overlooked in the mapping phase, dynamic obstacles or due to inaccuracies in the actions performed by the effectors. The local planner has to be reactive but with a good communication with the global planner.

The problems described above—localization, mapping, global planning and local planning—can be thought of as being solved in separated modules in a generic navigation controller. This helps us to create a formal structure to the generic problem of navigation, but these subdivisions are not essential. In the literature we find many implementations where some of these modules are fused or missing.

In what follows we review some of the ideas proposed in the literature for mapping, global and local planning and exploration. The important problem of localization will not be developed further. In this paper we assume that prior to our exploration and mapping routines there is an accurate localization system.

## 3. Previous and related works

### 3.1. Representations of the world

Roughly there are two main forms to represent the spatial relations in an environment: metric maps and topological maps. Metric maps are characterized by a representation where the position of the obstacles are indicated by coordinates in a global frame of reference. Some of them represent the environment with grids of points, defining regions that can be occupied or not by obstacles or goals [2,9]. Topological maps represent the environment with graphs that connect landmarks or places with special features [10,11].

In our approach we choose the grid-based map to represent the environment. Grid-based maps are very easy to build and maintain. Its only drawback is its bitmap-like representation of the world that makes the grids large objects with sizes scaling with the sizes of the environment. Another inconvenience is that it does not allow attribution of semantic features to the objects found in the environment. The clear advantage is that with grids we have the environment representation already discrete and ready to be used in conjunction with the harmonic function method for path planning.

The pioneer method for environment representation in a grid-based model was the certainty grid method developed at Carnegie Mellon University [9] by Moravec. The grid-based models were designed to cope with the spatial uncertainty generated by sonar sensors. They represent the environment as a 3D or 2D array of cells. Each cell stores the probability of the related region being occupied. The uncertainty related to the position of objects is described in the grid as a spatial distribution of these probabilities within the occupancy grid. The larger the spatial uncertainty, the greater the number of cells occupied by the observed object.

The update of these cells is performed during the navigation of the robot or through the exploration process by using an update rule function. Many researchers have proposed their own grid-based methods. The main difference among them is the function used to update the cell. Some of them are, for example: Fuzzy [12], Bayesian [13], Heuristic Probability [14], Gaussian [15], etc. In this work, we used a modified version of the Histogramic In-Motion Mapping (HIMM) method [2]. The update rule used in this method is a linear function of the number of observations made to each object in the environment. It is a simple rule and it is being used to show the efficiency and strength of our technique.

Each cell, in the HIMM method, has a certainty value, which is updated whenever it is being observed by the robot's sensors. The update is performed by increasing the certainty value by 3 (in the case of detection of an object) or by decreasing it by 1 (when no object is detected), where the certainty value is an integer between 0 and 15.

The HIMM was originally used in combination to a goal-driven mapping process. The core of this process is a system for obstacle avoidance called virtual force field (VFF) [16], which permits a real-time reactivity to treat dynamic obstacles.

### 3.2. Path planning approaches

Potential field methods for path planning are based on the idea that an optimal path can be obtained as a result of the linear superposition of fictitious forces $\mathbf{f}_i(\mathbf{r})$, or their potentials $p_i(\mathbf{r}) = -\int \mathbf{f}_i\, \mathrm{d}\mathbf{r}$, acting on the robot. Obstacles would exert repulsive forces while goals would apply attractive forces. The optimal path follows the gradient descent on the resulting potential $p(\mathbf{r}) = \sum_i p_i(\mathbf{r})$. It keeps the robot at safe distances from the obstacles and in most cases leads the robot to the goal. The method is relatively easy to implement, has good reactivity in dynamic environments, and it is computationally inexpensive.

There are, however, substantial problems with the method. Specially in the case of complex environments. One of these shortcomings is the presence of local minima in the resulting potential. In these minima the driving force vanishes due to specific obstacle and goal configurations and the robot gets trapped.

The use of harmonic functions for the potential solves this problem. Harmonic functions are the solutions of the Laplace's equation

$$\nabla^2 p(\mathbf{r}) = \sum_i \frac{\partial^2 p(\mathbf{r})}{\partial x_i^2} = 0.$$

Connolly and Grupen [1] showed that the solution of convenient boundary value problems involving the Laplace's equation generates optimal paths without spurious minima. In this method, using the idea of an occupancy grid, the environment is divided in a number of identical cells. Cells that show high probability to contain an obstacle have their potential value set to 1 while cells that contain the target have their potential value set to 0. A numerical algorithm is then employed to calculate the harmonic potential in the free space between the obstacles and the target. Therefore, all free space cells will have potential values between 0 and 1. Because of the mathematical properties of the harmonic functions, the gradient descent of the potential taken at any cell will indicate the direction that leads to the target [17].

In addition to potential field methods there are many other path planning approaches. Most of them are based on an explicit search on a graph representing the adjacency relations between features of the environment. They compute all possible accessible paths to the goal and weight them according to performance measures. A typical exemplar of this class is the Trulla algorithm [18].

Harmonic functions and algorithms like Trulla were developed to work with a fixed a priori map and a known goal. But these methods suit to dynamic environments as well. When the robot detects that an usual path has been interrupted it automatically generate a new alternate route. But it can be costly. Because of that there are many strategies that can be added to avoid total replanning of trajectories in the presence of a few unmodeled obstacles. For the harmonic potential method one of these techniques is to follow equipotential lines until the unexpected obstacle is cleared from the gradient descent path [1]. For Trulla any reactive routine can be added to avoid obstacles in order to find an unobstructed path. An interesting development of Trulla is the incorporation of a method to switch between reactive control and total replanning [19]. In this method the dot product between the displacement over the planned path and the displacement in real path is used as a measure of the deviation caused by the unmodeled obstacles. Small dot products serve as an indication for replanning.

Recently, Zelek [20] proposed a hybrid method that combines a local planner based on a harmonic function calculation in a restricted window with a global planning module that performs a search in a graph representation of the environment created from a CAD map. The harmonic function module is employed to generate the best path given the local conditions of the environment. The goal is projected by the global planner in the local windows to direct the robot.

### 3.3. Approaches to exploration

This section relates some interesting techniques used to exploratory mapping. They mix different localization methods, data structures, search strategies and map representations.

Kuipers and Byun [21] proposed an approach to explore an environment and to represent it in a structure based on layers, called Spatial Semantic Hierarchy

(SSH) [11]. The algorithm defines distinctive places and paths, which are linked to form an environmental topological description. After this, a geometrical description is extracted. The traditional approaches focus on geometric description before the topological. The distinctive places are defined from their properties and the distinctive paths are defined from the twofold robot control strategy: follow-the-mid-line or follow-the-left-wall. The algorithm uses a lookup table to keep information about the place visited and the direction taken. This allows a search in the environment for unvisited places.

Lee [22] developed an approach based on Kuipers work [21] on real robot. This approach is successfully tested on indoor office-like spaces. This environment is relatively static during the mapping process. Lee approach assumes that walls are parallel or perpendicular to each other. Furthermore, the system operates in a very simple environment comprised of cardboard barriers.

Mataric [10] proposed a map learning method based on a subsumption architecture. Her approach models the world as a graph, where the nodes correspond to landmarks and the edges indicate topological adjacencies. The landmarks are detected from the robot movement. The basic exploration process is wall-following combined with obstacle avoidance.

Oriolo et al. [12] developed a grid-based environment mapping process that uses fuzzy logic to update the grid cells. The mapping process runs on-line [23], and the local maps are built from the data obtained by the sensors and integrated to the environment map as the robot travels along the path defined by the A$^*$ algorithm to the goal. The algorithm has two phases. The first one is the perception phase. The robot acquires data from the sensors and updates its environment map. The second phase is the planning phase. The planning module re-plans a new safe path to the goal from the new explored area.

Thrun and Bücken [24,25] developed an exploration system which integrates both evidence grids and topological maps. The integration of the two approaches has the advantage of disambiguating different positions through the grid-based representation and to perform fast planning through the topological representation. The exploration process is performed through the identification and generation of the shortest paths between unoccupied regions and the robot. This approach works well in dynamic environments, however, the walls have to be flat and cannot make angles that differ more than 15$^\circ$ from the perpendicular.

Feder et al. [6] proposed a probabilistic approach to treat the concurrent mapping and localization using a sonar. This approach is an instance of feature-based approaches. It uses the extended Kalman filter to estimate the localization of the robot. The essence of this approach is to take actions that maximize the total knowledge about the system in the presence of measurement and navigational uncertainties. This approach was tested successfully in wheeled land robot and autonomous underwater vehicles (AUVs).

Yamauchi [3,26] developed the Frontier-Based Exploration to build maps based on grids. This method uses a concept of frontier, which consists of boundaries that separate the explored free space from the unexplored space. When a frontier is explored, the algorithm detects the nearest unexplored frontier and attempts to navigate to it by planning an obstacle-free path. The planner uses a depth-first search on the grid to reach that frontier. This process continues until all the frontiers are explored.

## 4. Exploration method using harmonic functions

We propose an approach based on the incremental calculation of a potential function using the harmonic functions method for path planning.

We define an activation window, that travels with the robot, with roughly the size of its average sensor range. This window indicates the new grid cells that are recruited for update, i.e., if a cell was at a given time in the activation window, it becomes part of the explored space by participating in the harmonic potential calculation for all times. The set of activated cells that compose the explored space we call activated region or potential region. Cells that were never inside the activation window indicate unexplored regions. Their potential values are set to zero and define the knowledge frontier of the state space—the real space in our case.

The detection of the nearest unexplored frontier comes naturally from the harmonic potential calculation. It can also be understood from the physical analogy with electrical potentials—obstacles repel while frontiers attract.

Consider that the robot starts from a given position in an initially unknown environment. Its activation window covers a portion of the field activating all cells localized there. The potential is then calculated using the values in the limits of the activated region as boundary conditions. Obstacles boundaries are set to 1 and non-visited, or knowledge limits, set to zero. The robot moves according to the gradient of the potential in its current position, that initially corresponds to the center of the activation window. In this case the gradient will point to the direction of the closest frontier with unvisited sites or to the largest one in the case there are two or more frontiers at the same distance. Therefore the robot moves maximizing knowledge gain [5]. If no obstacle is detected the limits of the activated region are zero and the potential is zero in all cells. In this case or in any other situation where there is no gradient to guide the robot, it simply follows the forward direction.

This approach differs from Zelek's similar idea on local updates of harmonic functions in two fundamental points. First, Zelek considers a global path planner that indicates the general direction to the goal. With that information it is possible to project the goal inside his local window to orient the robot's behavior. The harmonic calculation made in the local window is devoted merely to the task of avoiding obstacles of the robot's local vicinity. In our algorithm the potential region increases while the robot explores thus defining a global map with optimal paths that characterizes the harmonic potential approach. For us the goal location is unknown, and the robots behavior is truly exploratory.

A second difference is in Zelek's local window dynamics. It does not move in real-time with the robot. Once the window is defined, the goal projection is made and the potential calculated, the robot travels following gradient descent on the potential surface, with the windows position fixed until it attains its border. Then a new window, centered in the robot's coordinate, is selected and the process starts again. The result is a poor reactive behavior. In average, the robot ends up getting too close to the obstacles. Closer than what is necessary to incorporate them in its path strategy.

Fig. 1 illustrates the exploratory behavior generated by our controller. The robot starts in the position indicated by the dashed circle, and sequentially occupies the positions a, b, c and d, with the final position indicated by the black circle. Fig. 1 also shows snapshots of the internal map at the corresponding positions. At any instant, the vector field points to the closest unexplored place in the environment. In Fig. 1(c) and (d) we see how the vector field changes pulling the robot from a dead end.

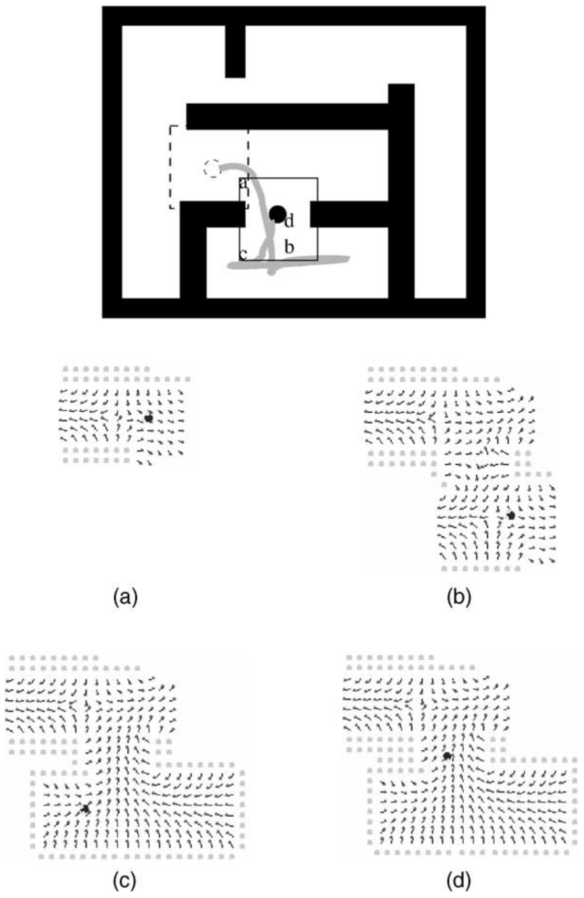


Fig. 1. Exploration process: on the top the trajectory followed by the robot, with the activation window. (a)–(d) show snapshots of the field (arrows) and grid configurations (gray squares) for each corresponding position in the trajectory.

## 5. Implementation

In order to test our method we have implemented it in the Nomad 200 platform (Nomadic). The Nomad 200 is an integrated system with six sensory modules: tactile, infrared, ultrasonic, basic vision, structured light vision and compass systems. We use only its sonar capabilities, which is comprised of 16 channels that give range information from 17 in. to 255 in. with 1% accuracy over the entire range. The system is mounted on a three wheels base. The wheels are aligned and they translate and rotate synchronously. Because of the 360° wheel rotation the robot has a zero gyro-radius.

In addition to using the real robot we have also performed experiments on the Nomad simulator as well as simulations with a simple model for the robot and its sensors. The Nomad Simulator allows a systematic study of how the performance of our method depends on details of the algorithm. On the other hand, with the simple model it is possible to compare typical performances of the method to alternative benchmark methods like random exploration and wall following.

In what follows we describe the necessary ingredients for the implementation of our algorithm.

### 5.1. Mapping

The internal map is represented in an $N_x \times N_y$ grid. Each grid cell represents a $\Delta L \times \Delta L$ real-world square area and stores the following attributes:

- Potential ($p(\mathbf{r})$): represents the potential value in the cell centered in position $\mathbf{r} = (i, j)$.
- State: indicates the cell situation, which can be: *not explored*, *free space* or *occupied*. The attribute value *not explored* indicates that the cell was never visited and its potential value is set to 0. The value *free space* indicates that the cells was explored, and it is free, therefore its potential value can be updated. The attribute value *occupied* indicates that the cell is occupied by an object in the real-world and its potential value is set to 1 in the case of an obstacle and 0 in the case of a goal.
- Certainty ($c(\mathbf{r})$): represents the certainty value defined in the HIMM for the cell centered in the position $\mathbf{r} = (i, j)$. The larger this value, the greater the certainty of the object presence.

### 5.2. Running

When the experiment begins the robot sets its position equal to $(0, 0)$, which corresponds to the center of the 2D grid map. All cells have their state attribute set to *not explored*; certainty and potential values initialized to 0.

Throughout the exploration process the robot performs the algorithm described below:

(1) Activates and reads the sonar sensors.
(2) Performs local update of the map inside the activation window.
(3) Updates the harmonic potential over the potential region.
(4) Calculates the gradient vector at the current position.
(5) Moves following the direction defined by the gradient descent on the potential.
(6) Repeats this procedure until all the environment is explored.

The robot activates the sonar sensors to get information over the environment around it. Each one of the 16 sonar sensors returns a scalar value $d_i$ that corresponds to the distance between the sensor and the closest object in that particular direction. Using the knowledge about the robot's position and the orientation of the sensors the values $d_i$, corrected for the robot's radius, are used to update the certainty value in the cells corresponding to the object positions if they are inside the activation window. Readings from outside the activation window are not used to update the map. Fig. 2 shows how the sonar signal from a wall is interpreted and incorporated into the map. The certainty value of each cell in a region $\partial\gamma$, corresponding to the limits of the sensors view cone, is increased by 3. The other cells inside the view cone are decreased by 1. The certainty value is clipped between 0 and 15. A cell has its state attribute changed from *free space* to *occupied*, when its certainty value is bigger than 2. This process allows an easy treatment of dynamic and static objects.

After the map updating, the potential field update of the visited cells is performed. Two relaxation methods for updating the potential field where used in this paper: the Gauss–Seidel and Successive Over-Relaxation (SOR) methods. The Gauss–Seidel method updates the map cells according to the iteration rule:

$$p_{i,j}^{\text{new}} = \tfrac{1}{4}(p_{i-1,j}^{\text{new}} + p_{i+1,j}^{\text{old}} + p_{i,j-1}^{\text{new}} + p_{i,j+1}^{\text{old}}), \qquad (1)$$

whereas SOR method uses the following iteration

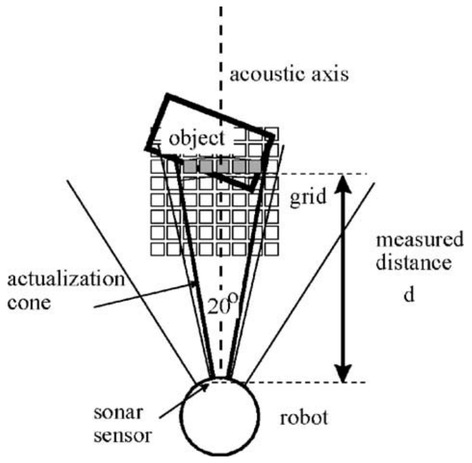


Fig. 2. Actualization of the grid: the figure shows an schematic diagram of an object detection by one of the robot sonar sensors. The map is actualized by adding 3 to the certainty value of the cells inside the region $(\partial\gamma)$, and by decreasing by 1 the certainty value of the remaining cells inside the sensors view cone.

rule:

$$p_{i,j}^{\text{new}} = p_{i,j}^{\text{old}} + \tfrac{1}{4}w(p_{i-1,j}^{\text{new}} + p_{i+1,j}^{\text{old}} + p_{i,j-1}^{\text{new}} + p_{i,j+1}^{\text{old}} - 4p_{i,j}^{\text{old}}) \qquad (2)$$

with the over-relaxation parameter

$$w = \frac{2}{(1+\sqrt{1-\rho^2})}, \qquad (3)$$

where

$$\rho = \frac{1}{2}\left[\cos\left(\frac{\pi}{L_x}\right) + \cos\left(\frac{\pi}{L_y}\right)\right]. \qquad (4)$$

A single actualization of the potential field consists in the update of all grid points inside the potential region. The procedure is repeated a number of times before the robot moves. Then the negative gradient is computed on the cell containing the current position of the robot, and the robot moves a fixed displacement $\Delta d$ after adjusting its head direction to the angle $\theta$ given by

$$\theta = \arctan(D_x, D_y), \qquad (5)$$

where $\arctan(x, y)$ is the inverse tangent taken in the interval $[-\pi, \pi]$ with

$$D_x = (p_{i-1,j} - p_{i+1,j}), \quad D_y = (p_{i,j-1} - p_{i,j+1}). \qquad (6)$$

The above procedure with alternating calculation and motion is applied only in the simulated experiments. In the experiments involving the Nomad 200 robot, due to its physical inertia, it is more convenient to keep it in motion while the whole process of reading sensors, map update, potential and gradient calculations takes place. Its speed is adjusted according to

$$v = v_{\max}\left(0.2 + 0.8\frac{90 - |\Delta\theta|}{90}\right), \qquad (7)$$

where $v_{\max} = 10\,\text{in./s}$. The angular correction $\Delta\theta$ is the difference between the robot's orientation and the negative potential gradient direction $\theta$ for the current position, reduced to the first quadrant. Therefore, its maximal absolute value is 90°, and any turn exceeding 90° in absolute value is implemented as a combination of a $\Delta\theta$ turn and the reversion of the wheels translation. The functional form of (7) and its parameter values were chosen heuristically to propitiate a reasonably smooth motion.

### 5.3. Update rate

When harmonic functions are used for path planning the potential is actualized until it stabilizes within a predefined precision $\epsilon < 10^{-p}$ for the entire map. Only then the robot is put in motion [1,4]. That can take a large number of iterations. They are of the order of $pM$ for Gauss–Seidel or $p\sqrt{M}$ for SOR, where $M = N_x \times N_y$ is the number of cells in the environment [27]. In our algorithm for exploration, the map grows as the robot explores. We found that the complete relaxation of the potential is not always required. Therefore, in order to save computing time we work with a fixed iteration rate $r$. In the simulator, since inertia is not an issue, we stop the robot during that update process and the robot followed the map in steps. Therefore, the update rate $r$ is better expressed in iterations/step. For the real robot, on the other hand, the process of sensor reading, map update and potential calculation is performed in a fixed time frequency, therefore for this case the update rate $r$ is expressed in iterations/s.

## 6. Results

In this section we present the results obtained using our approach in a series of experiments where different aspects of the problem of exploring and mapping an environment are illustrated and discussed. Both simulated experiments and experiments with the Nomad 200 robot were performed.

### 6.1. *Exploration experiments*

These experiments aim to demonstrate that, as long as enough relaxations are performed, full exploration of the environment is always accomplished. Fig. 3 shows an experimental result captured from the Nomad Simulator. The simulated robot is inserted in a rectangular environment of 400 in. $\times$ 330 in. with several obstacles forming a maze. Its internal grid map is composed by $80 \times 80$ cells, where each cell is square with dimensions 6 in. $\times$ 6 in. Therefore, the internal map can represent environments up to 480 in. $\times$ 480 in., what is adequate for our particular simulation. The activation window that represents the maximal distance from the robot where sensory readings are used to update the grid is a circular region of radius 80 in. A robot step corresponds to a displacement of $\Delta d = 3$ in. The potential field is updated at a rate of 30 iterations/step with Gauss–Seidel updates.

For each snapshot in Fig. 3, Fig. 4 shows the partial map where obstacles already detected are displayed along with the corresponding vector field that guides the robot in the free space. Several local minima are present in the vector field. This is a consequence of the fact that the potential did not relax globally to the harmonic solution with 30 iterations/step. In the vicinity of the robot, however, the vector field points to the closest unexplored area and this is what interests for exploration.

Difficulties arise when the closest unexplored region is in fact very far from the actual position of the robot, and therefore the potential needs to be globally relaxed to indicate optimal paths. In this case there is

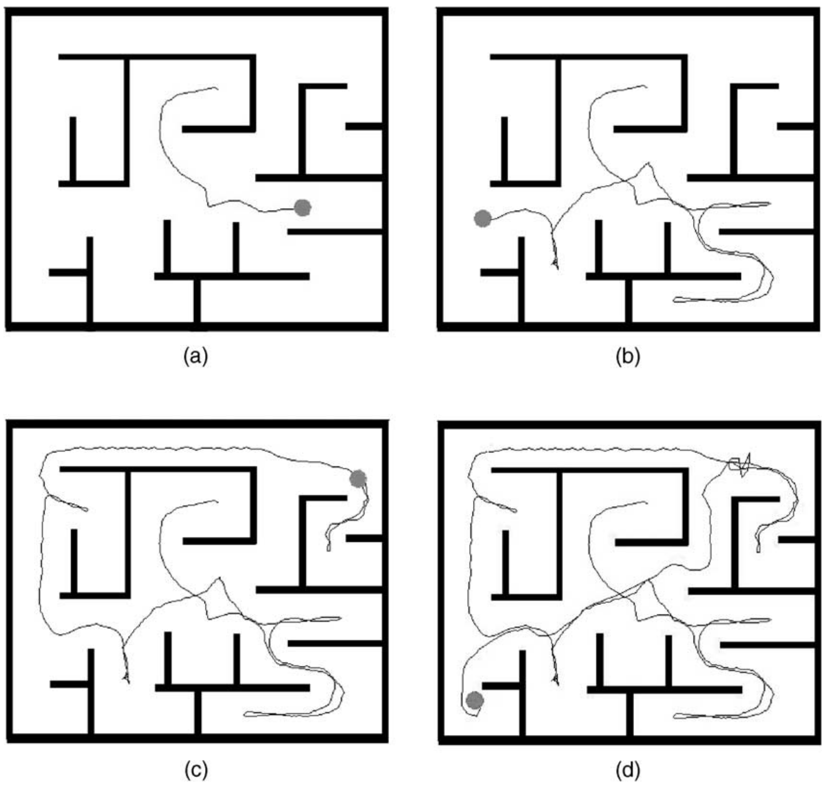


Fig. 3. Environment exploration sequence from the Nomad Simulator: trajectory followed by the robot. The environment is 400 in. $\times$ 330 in., and the robot moves in steps of $\Delta d = 3$ in. The potential field is updated using Gauss–Seidel at an iteration rate of $r = 30$ iterations/step.

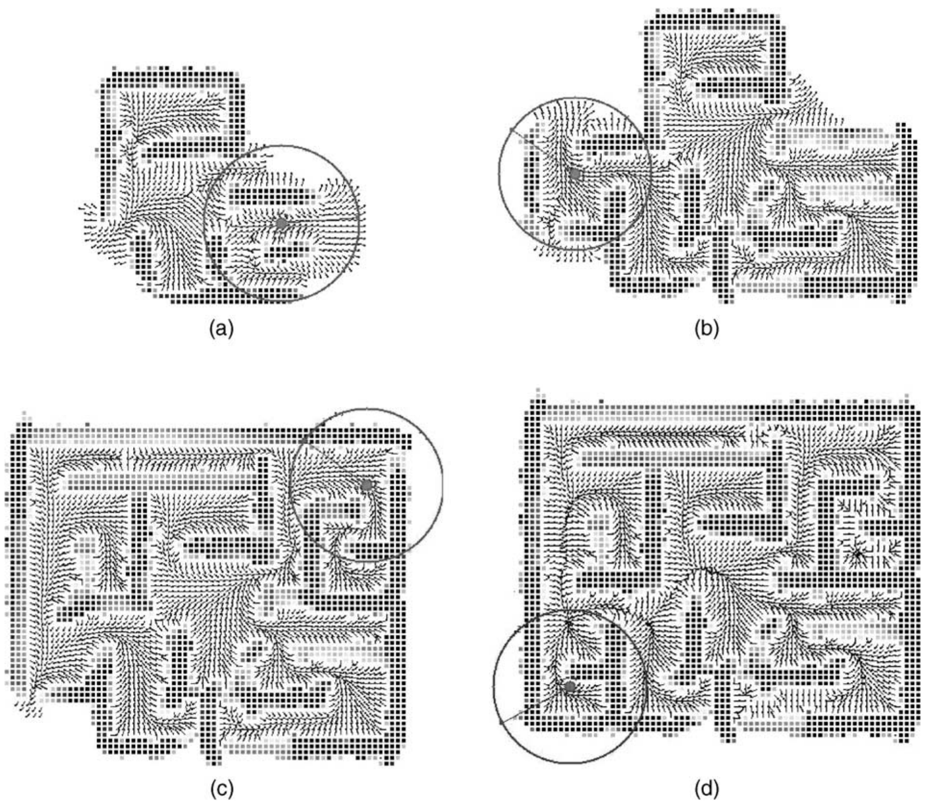


Fig. 4. Environment exploration sequence: partial instances of the vector fields and maps during exploration. The vector field represents the gradient descent in each free space cell and it is normalized to facilitate view. In the grid map the certainty value of a cell is represented by gray levels with white corresponding to 0 and black corresponding to 15. The grid size is $80 \times 80$ cells and each cell corresponds to a 6 in. $\times$ 6 in. real space region. The activation window is a circle with radius 80 in.

a considerable decrease in performance. It occurs in general after the exploration of a dead end. That can be observed on the top right of Fig. 3(d) where the robot's trajectory oscillates. Such behavior is not visible in Fig. 3(a)–(c) because in these cases the robot is relatively close to the unexplored regions. A possible solution for this specific problem is to consider a dynamic update rate that increases when the average velocity of the robot falls below a predefined threshold.

Fig. 5 shows an exploratory sequence for a simulated environment with polygonal obstacles and generically oriented walls. The dimensions and parameters are the same found in Fig. 3. This demonstrate the robustness of the method, in particular of grid-based representations, when applied in more generic environments. Some approaches in literature have good performances only for certain obstacle configurations and shapes. Here this limitation is not present.

Fig. 6 shows an experiment where the Nomad 200 robot explores an office environment in our Institute. It consists of a room divided by two cardboard walls, connected to an L-shaped corridor. The room has tables and chair placed along the walls. In order to map the environment a $100 \times 100$ cells grid was used, where each cell corresponds to a 6 in. $\times$ 6 in. region in real space. The robot traveled a total of 1441 in. to explore the environment in approximately 5 min with an average speed of 5 in./s. The actualization window has a 80 in. radius and it is shown in the figure.

In the case of Figs. 3, 5, 6 and 8, since a target is not present the potential field is of no use after full exploration. In fact, if enough time is given the update leads to a constant potential in which case the

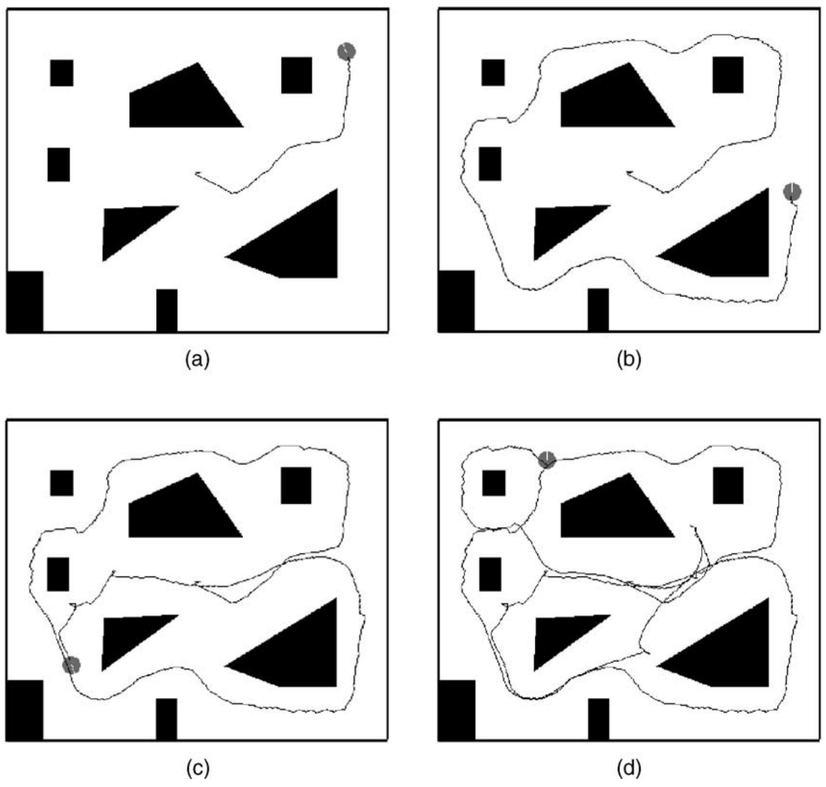


Fig. 5. Environment exploration sequence: polygonal environment. The environment is 400 in. × 330 in., and the robot moves in steps of $\Delta d = 3$ in. The potential field is updated using Gauss–Seidel at an iteration rate of $r = 30$ iterations/step.

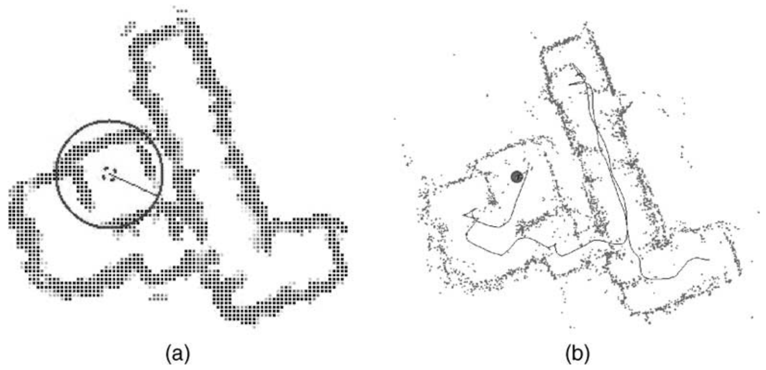


Fig. 6. Exploration of an office environment by the Nomad 200 robot. (a) Grid map obtained after exploration. The grid size is 100 × 100 cells and each cell corresponds to a 6 in. × 6 in. real space region. The activation window is a circle with radius 80 in. (b) The trajectory followed during exploration. Its total length was 1441 in. and it was traveled at an average velocity of 5 in./s. The sonar readings are also shown. Observe that they are noisier than the map. This is due to the fact that readings outside of the activation windows are discarded and only consistent readings update the certainty attribute.

gradient vanishes in all points. Hence the outcome of the exploration is only the geometrical map that displays the localization of all obstacles through the certainty value of each cell in the occupancy grid, $c(\mathbf{r})$ for $\mathbf{r} = (i, j)$, $i \in [1, N_x]$, and $j \in [1, N_y]$.

### 6.2. Exploration performance experiments

Here we compare the performance of different techniques for exploration. First, we run simple benchmark exploration algorithms relative to which our approach can be measured. Then we use the Nomad Simulator to test the two different update methods for the potential function, Gauss–Seidel and SOR, as well as to measure how performance depends on the iteration rate $r$. Finally, we performed some experiments with the Nomad robot to test the robustness of the algorithm in realistic situations.

#### 6.2.1. Benchmarks

In order to calculate benchmark performances we consider a very simple model where a square $L \times L$ environment is divided in $N \times N$ cells that are either occupied or not, and the robot is a point that moves in the continuous space. While it moves it explores a circular region of diameter $A$. Errors associated to detection of obstacles are not considered. When a cell occupied by an obstacle is inside the activation window its certainty value is set to the maximum value.

The algorithm performance for a given environment can be measured by the total length ($\ell$) of the path used by the robot to explore it. Strictly, this is a measure that mix two different performance aspects: the quality of the path (its smoothness) and the strategy of coverage of the environment (the exploration sequence followed by the robot). But we choose not to distinguish between them.

Our benchmark algorithms are:

- *Random exploration*. The robot moves randomly in the environment. We consider that the robot follows straight lines according to

$$x_{t+1} = x_t + \Delta d \cos(\theta),$$
$$y_{t+1} = y_t + \Delta d \sin(\theta), \tag{8}$$

  and changes direction $\theta$ randomly when its distance from an obstacle is less than a minimum distance $d_{\min}$. This proved to be more efficient than a random trajectory where in every step the robot chooses a new direction.
- *Ideal wall following*. The robot moves following the walls within a certain distance. Here we simply measure the total length of the wall-following trajectory necessary to explore the environment. This is not a realistic implementation of a wall-following algorithm. It is just intended to give a geometrical estimate of the order of magnitude of this procedure.

Table 1
Simulation result for a random exploration, wall-following and the harmonic function method in the environment of Fig. 7 represented in a $100 \times 100$ grid over 30 trials[a]

| Method | $A/L$ | $\bar{\ell}/L$ | $\sigma_\ell/L$ |
|---|---|---|---|
| Random exploration | 0.1 | 150.86 | 57.60 |
| | 0.4 | 48.41 | 27.82 |
| Wall following | 0.1 | 8.31[b] | 0.25 |
| | 0.4 | 6.69 | 0.69 |
| Harmonic function | 0.1 | 21.50 | 2.40 |
| | 0.4 | 6.15 | 0.42 |

[a] For random exploration the robot cannot be closer than $d_{\min} = 0.5A$ from the walls. For wall-following the robot follows the wall at $d_{\min}/L = 0.05$ for $A/L = 0.1$ and $d_{\min}/L = 0.08$ for $A/L = 0.4$.

[b] For $A/L = 0.1$ the wall-following procedure does not cover completely the environment. The harmonic function method used Gauss–Seidel at a rate of $r = 30$ iterations/step.

Table 1 shows the average results $\bar{\ell}$ and the standard deviations $\sigma_\ell$ of the total trajectory length necessary to complete exploration of the environment in Fig. 7. To better compare with the results from the Nomad Simulator we present the results scale by the size of the environment ($L$). For each method 30 trials where performed with the robot starting from random initial positions.

The results indicates that the proposed harmonic function exploration method has performances comparable to an ideal wall-following procedure when the sensory range is large enough. For $A/L = 0.1$ the comparison is not appropriated since the environment is not completely explored by following the walls at a fixed distance. It is important to notice that, even though an ideal wall-following procedure is competitive with our method, a real wall-following algorithm might not be. The reason is that in general it has to

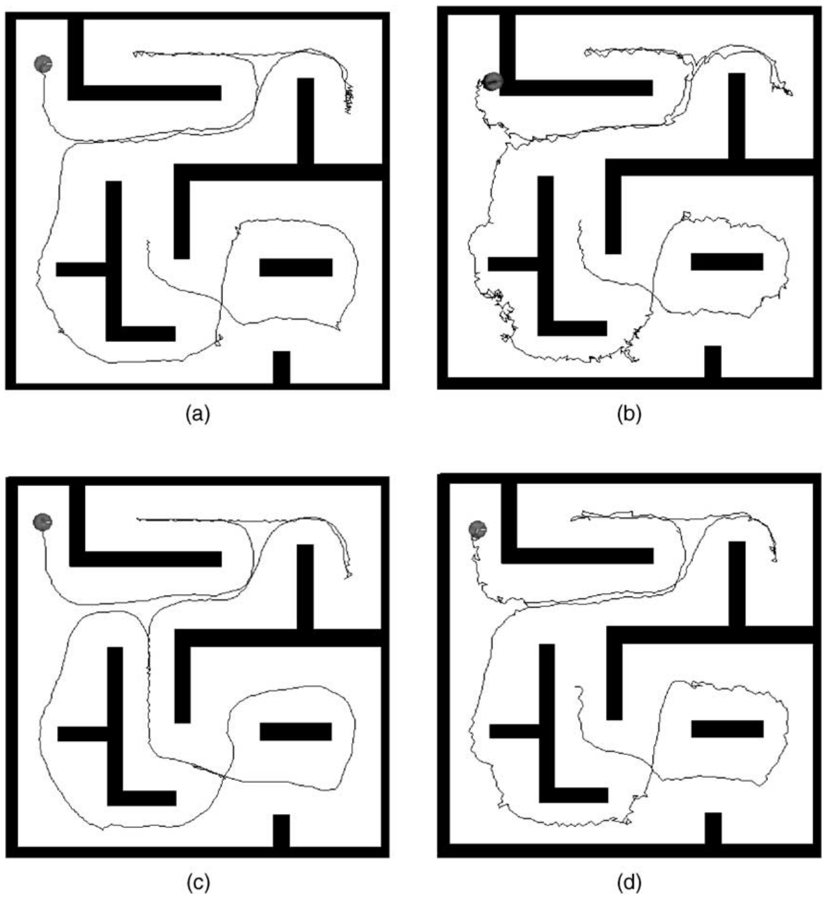


Fig. 7. Exploration sequence. (a) Trajectory followed using Gauss–Seidel method for $r = 10$ iterations/step. (b) Trajectory followed using SOR method for $r = 10$ iterations/step. (c) Trajectory followed using Gauss–Seidel method for $r = 30$ iterations/step. (d) Trajectory followed using SOR method for $r = 30$ iterations/step. The environment is 400 in. × 400 in., and the robot moves in steps of $\Delta d = 3$ in. The grid size is 80 × 80 cells and each cell corresponds to a 6 in. × 6 in. real space region. The activation window is a circle with radius 80 in.

cope with non-connected walls. Therefore, an additional method for searching non-charted walls has to be introduced and that can decrease the wall-following performance.

### 6.2.2. *Update methods and rates*

Fig. 7(a) and (b), taking from the Nomad Simulator, show examples of the exploratory trajectory of the simulated robot when the potential field is updated by Gauss–Seidel and SOR, respectively, at a rate of 10 iterations/step. With Gauss–Seidel updates the robot finishes the exploration of the environment with $\ell = 2353$ in. while with SOR it ends up colliding with the wall.

If we increase the update rate to 30 iterations/step we obtain the result shown in Fig. 7(c) and (d). With this rate the robot using SOR no longer fails to explore the environment. But Gauss–Seidel is still a better method traveling $\ell = 2335$ in. while SOR used $\ell = 2419$ in.

Table 2 presents the average performances of our method for the exploration of the environment in Fig. 7 after 10 trials. In each trial the robot was initialized in a different position and orientation. The relatively small standard deviation gives a measure of the dependability of the algorithm. Compare it, for instance, to the standard deviations of the random exploration in Table 1

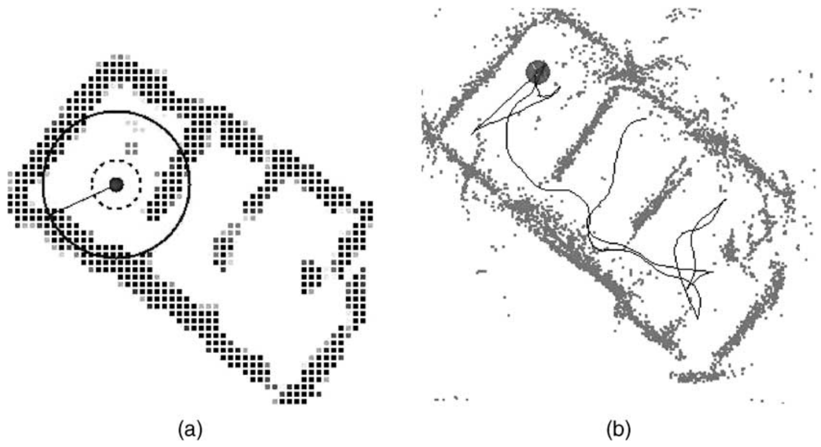


Fig. 8. Exploration experiment with the Nomad 200 robot in a cardboard maze. The environment dimensions are approximately 138 in. × 308 in. The robot vision radius is 80 in., and the iteration rate is $r = 20$ iteration/s using Gauss–Seidel updates.

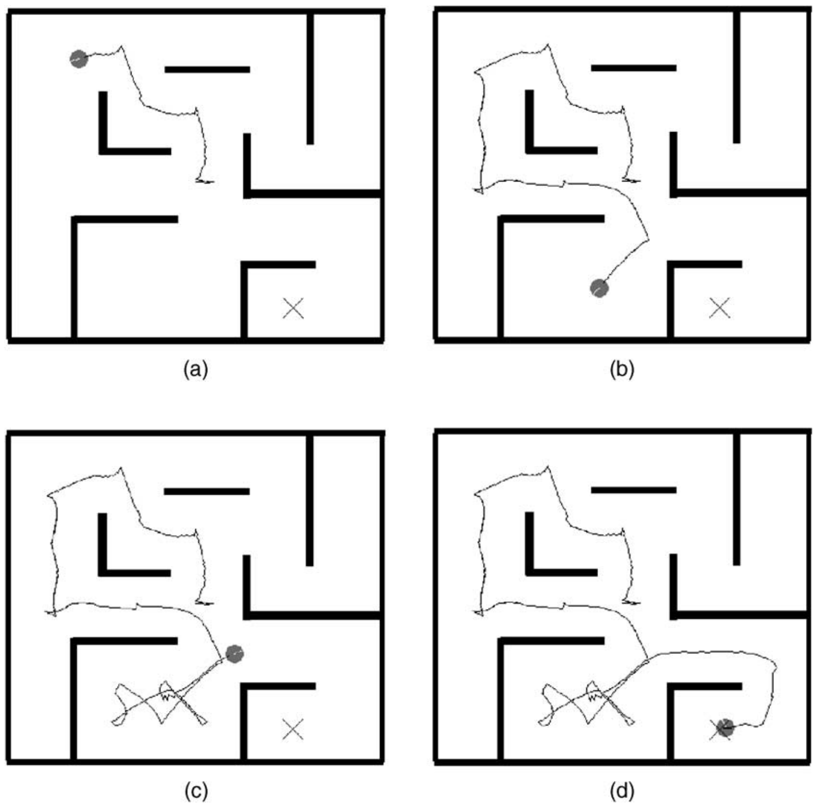


Fig. 9. Goal finding: trajectory followed by the robot. The environment is 400 in. × 400 in., and the robot moves in steps of $\Delta d = 3$ in. The grid size is 80 × 80 cells and each cell corresponds to a 6 in. × 6 in. real space region. The activation window is a circle with radius 80 in. The iteration rate is $r = 30$ iterations/step using Gauss–Seidel updates.

Table 2
Simulation results from the Nomad Simulator in the environment of Fig. 7[a]

| Relaxation method | Iteration rate | $\bar{\ell}/L$ | $\sigma_\ell/L$ |
|---|---|---|---|
| Gauss–Seidel | 10 | 6.35 | 0.50 |
| | 30 | 5.60 | 0.34 |
| SOR | 10 | 4 hits in 10 trials | – |
| | 30 | 6.53 | 0.90 |

[a] In the simulator the environment dimensions were 400 in. × 400 in., the vision radius 80 in., and the average was performed over 10 trials.

Fig. 8 shows an instance of an exploratory path on a small three rooms maze performed by the Nomad 200 robot using Gauss–Seidel with iteration rate of 20 iteration/s. Table 3 shows the result of four experiments in the same environment with the robot starting at different initial positions.

## 6.3. *Target finding experiments*

In this phase we illustrate a goal-driven exploration, where the robot must explore the environment to locate a target. There is no a priori knowledge of its location. The environment parameters are the same used in Section 6.1. The robot is inserted in a 400 in. × 400 in. environment represented in a 80 × 80 grid. The activation window is circular with radius 80 in. and the map is updated with Gauss–Seidel at a rate of 30 iterations/step.

Table 3
Average results for four exploration experiments in the maze in Fig. 8 with the Nomad 200 robot[a]

| Relaxation method | Iteration rate | $\bar{\ell}$ (in.) | $\sigma_\ell$ (in.) |
|---|---|---|---|
| Gauss–Seidel | 20 | 538.5 | 24.346 |

[a] The environment dimensions are 138 in. × 308 in. The robot vision radius 80 in., and the iteration rate is 20 iteration/s.

Figs. 9 and 10, from the Nomad Simulator, show the result. The exploration stops as soon as the target is found, and a partial map is built in the region known to the robot. In this case the target indicated by the "X" on the right of Fig. 9. This experiment does not differ fundamentally from the one in Section 6.1. The main

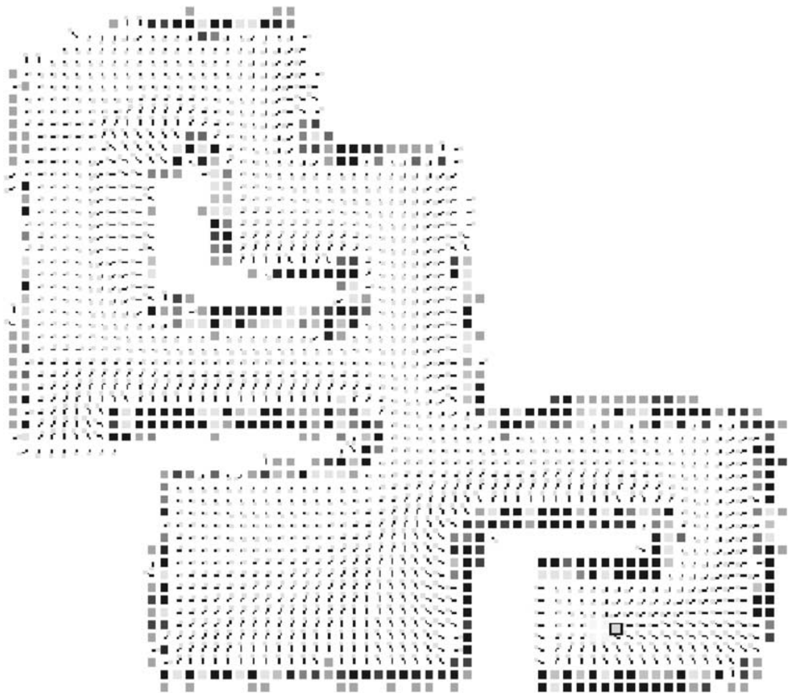


Fig. 10. Goal finding: grid map and vector field. Like before the vector field represents the gradient descent in each free space cell and it is normalized to facilitate view. In the grid map the certainty value of a cell is represented by gray levels with white corresponding to 0 and black corresponding to 15.

point here, however, is that the goal is a static attractive object that stops search as soon as it is reached.[1]

Moreover, the potential field, that was discarded at the end of the targetless exploration, here carries important information about how to reach the target. Therefore we can say that in the end of an exploration with a target the final products of the process are: the partial map of the environment $c(\mathbf{r})$ and the potential field $p(\mathbf{r})$ that can be used to plan the best paths to the target.

The same experiment could be carried out prioritizing exploration, in which case the target is recognized but the correspondent cells are still marked as *free space*. The result would be much more interesting. The robot would be strongly attracted to the boundaries of the explored region and it returns to the target only after full exploration. The resulting map would be a complete map of the environment that allows navigation to the target.

## 7. Discussions and conclusions

The motivation of this paper was to answer the following question: can a unified framework—a single principle—provide the basis for a robust exploratory algorithm? The main idea is to develop algorithms that avoid local binary decision processes that switch between different heuristic solutions. The harmonic potential technique for path planning is an example of this. At present most of the applications in robotics of this technique aimed at path planning. In these cases, the environment configuration and the goal location are known. We show herein that it is possible to extend the application of harmonic functions to exploration using partially relaxed versions of the potential field. The experiments using the Nomad 200 robot demonstrate that, by appropriated parameter adjusting, acceptable performances in real time are obtained using relatively low computational resources. The experiment in Fig. 6 was performed on a Celeron 475 MHz running Linux and the robot took approximately 5 min to complete the 36.6 m (1441 in.) exploratory trajectory, on a average speed of 5 in./s, what is a reasonable office speed.

The key parameters for the algorithm performance are the update rate $r$, the length traveled in one step $\Delta d$, the size of the activation window $A$, and the update method. The optimal value for the update rate is an important issue for practical implementation, since this value is directly related to the execution time. To address this point we have tested values for $r$ between $r = 10$ iterations/step and $r = 30$ interaction/step. We also varied the update method using in some cases Gauss–Seidel and in others SOR. The number of iterations necessary to reduce the error between the actual Laplace's solution and the iterated solution below $10^{-p}$ is $n_{\mathrm{GS}} \sim pM$ for Gauss–Seidel and $n_{\mathrm{SOR}} \sim p\sqrt{M}$, where $M = N_x \times N_y$ is the number of grid points [27]. But despite the clear advantage of SOR our results demonstrate that when partial relaxed version of the potential is required (that is what we get when we perform 10 iterations before a step) SOR performs poorly when compared to Gauss–Seidel. This is due to the fact that SOR makes shortcuts to attain the equilibrium solution ($\nabla^2 p(\mathbf{r}) = 0$) what implies that its intermediate solutions are not to be trusted. In fact, during SOR relaxations, we observe wave-like patterns that distort the vector field causing paths that sometimes end in a wall. The resulting behavior was shown in Fig. 7 (b). Gauss–Seidel, on the other hand, solves the Laplace problem as a discrete version of the diffusion problem $\partial p(\mathbf{r}, t)/\partial t = \nabla^2 p(\mathbf{r}, t)$ what means that the intermediate solutions $p(\mathbf{r}, t)$ are consistent with the physical problem of repulsive obstacles and attractive goals. Consequently, it will never produce paths that collide with detected walls.

A second issue is the quality of the paths. For the very reason discussed above, Gauss–Seidel produces paths that are smoother than the paths produced by SOR. It reflects in the performances observed in Table 1 where we see that Gauss–Seidel with $r = 10$ iterations/step its comparable to SOR with $r = 30$ iteraction/step. The conclusion here is that Gauss–Seidel is the appropriate method to update the potential field when partial updates are required.

The results presented here demonstrate that an algorithm that uses harmonic functions to guide exploration is robust and of simple implementation. Much can be done, however, to improve it even more

[1] In the simulation the robot is not programmed to stop so it moves around the goal. Furthermore, to built a potential field connecting all the points in the explored space to the goal the frontiers have to cease to be attractive. Therefore, when the goal is recognized all cells in the frontier have their potential set to 1.

for practical use. A complete search in the parameter space can help to establish optimal limits of the method. Parameters like the size of the activation window and the step size that were not optimized here are candidates to have important impacts on performance. The idea of the adaptive rate has yet to be tested. Besides, the intrinsic limitations of the subsidiary methods that compose our exploration method have to be circumvent. For instance, the modified version of HIMM method we use to update the map from sonar readings is very crude. Right now its crudeness help us to make the point that the method is robust enough to perform well even with this crude map making. However, much improvement can be obtained if the sonar is better interpreted.

The harmonic potential have its problems as well. First, its computation time is proportional to the grid size. This could have impact in our controller performance when the potential regions grows. A possible solution is to use a more efficient grid sampling like quad-trees to account for inhomogeneous distributions of objects [20]. Another limitation is the fact that harmonic functions are rapid decaying functions what reduce their usefulness for path planning of large environments. To address that we have been investigating a family of functions with similar properties regarding absence of local minima that can be used in their place for the potential field update [28].

The comparison with techniques proposed by other authors depends on the availability of codes and it is difficult due to the lack of common standards for environments and robot platforms. Therefore, we choose to make comparisons with simple techniques that are generic enough to be implementation independent. Random exploration is certainly the most inefficient of all exploratory techniques. There is no merit in outperforming random exploration. Nonetheless due to its generality random exploration can serve as a standard measure for the complexity of an environment. Wall-following is an interesting technique to explore connected obstacles. It is not per se a complete technique to explore environments since they can be composed of non-connected obstacles, like islands of walls or objects. Therefore, to be able to fully explore an environment the wall-following procedure has to be associated to a search strategy. This implies a diagnostic of some sort that instruments the switch between the two behaviors. The advantage of our approach is that a binary decision process, which is very non-linear in nature and therefore error prone, is absent. A single principle is at work during all the process. Besides the potential shape that guides exploration is the result of the distributed action of all detected obstacles. It means that the errors in the detection of single obstacles have low impact in behavior.

## Acknowledgements

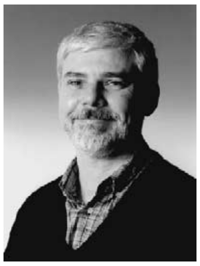
The authors wish to thank Prof. C. Scherer for his comments on the subtle differences between Gauss–Seidel and SOR methods for the numerical calculation of Laplace's problem. This work has been partially supported by Brazilian agencies Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Financiadora de Estudos e Projetos (FINEP) and Fundação de Amparo à Pesquisa do Rio Grande do Sul (FAPERGS).

## References

[1] C.I. Connolly, R.A. Grupen, On the application of harmonic functions to robotics, Journal of Robotic Systems 10 (1993) 931–946.
[2] J. Borenstein, Y. Koren, Histogramic in-motion mapping for mobile robot obstacle avoidance, IEEE Journal of Robotics and Automation 7 (4) (1991) 535–539.
[3] B. Yamauchi, A frontier-based exploration for autonomous exploration, in: Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey, CA, 1997, pp. 146–151.
[4] C.I. Connolly, Harmonic functions and collision probabilities, International Journal of Robotic Research 16 (1997) 497–507.
[5] S.B. Thrun, The role of exploration in learning control, in: D.A. White, D.A. Sofge (Eds.), Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches, Van Nostrand Reinhold, Florence, KY, 1992.
[6] H.J.S. Feder, J.J. Leonard, C.M. Smith, Adaptive mobile robot navigation and mapping, International Journal of Robotics Research 18 (7) (1999) 650–668.
[7] S. Thrun, Probabilistic algorithms in robotics, Tech. Rept. CMU-CS-00-126, Carnegie Mellon University, Pittsburgh, PA, 2000.
[8] S. Thrun, W. Burgard, D. Fox, A probabilistic approach to concurrent mapping and localization for mobile robots, Machine Learning and Autonomous Robots (Joint Issue) 31 (5) (1998) 1–25.
[9] H.P. Moravec, A. Elfes, High resolution maps from wide angle sonar, in: Proceedings of the IEEE International Conference on Robotics and Automation, St. Louis, MO, 1985.

[10] M. Mataric, Integration of representation into goal-driven behavior-based robots, IEEE Transaction on Robotics and Automation 3 (1992) 304–312.
[11] B. Kuipers, T. Levitt, Navigation and mapping in large-scale space, AI Magazine 9 (1988) 25–43.
[12] G. Oriolo, G. Ulivi, M. Vandittelli, Fuzzy maps: A new tool for mobile robot perception and planning, Journal of Robotic Systems 14 (3) (1997) 179–197.
[13] A. Howard, L. Kitchen, Generating sonar maps in highly specular environments, in: Proceedings of the Fourth International Conference on Control, Automation, Robotics and Vision, 1996.
[14] A. Elfes, Sonar-based real world mapping and navigation, IEEE Journal of Robotics and Automation RA-3 (3) (1987) 249–265.
[15] A. Elfes, Using occupancy grids for mobile robot perception and navigation, Computer Magazine (June 1989) 46–57.
[16] J. Borenstein, Y. Koren, The vector field histogram-fast obstacle avoidance for mobile robots, IEEE Journal of Robotics and Automation 7 (3) (1991) 278–288.
[17] R. Courant, D. Hilbert, Methods of Mathematical Physics, Vol. 2, Wiley, New York, 1962.
[18] K. Hughes, A. Tokuta, R. Ranganathan, Trulla: An algorithm for path planning among weighted regions by localized propagations, in: Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems, Raleigh, NC, 1992, pp. 7–10.
[19] R.R. Murphy, K. Hughes, A. Marzilli, E. Noll, Integrating explicit path planning with reactive control of mobile robots using Trulla, Robotics and Autonomous Systems 27 (1999) 225–245.
[20] J.S. Zelek, A framework for mobile robot concurrent path planning and execution in incomplete and uncertain environments, in: Proceedings of the AIPS-98 Workshop on Integrating Planning, Scheduling and Execution in Dynamic and Uncertain Environments, Pittsburgh, PA, 1998.
[21] B. Kuipers, Y.-T. Byun, A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations, Robotics and Autonomous Systems 8 (1991) 47–63.
[22] W.Y. Lee, Spatial semantic hierarchy for a physical robot, Ph.D. Thesis, Department of Computer Sciences, The University of Texas, Austin, TX, 1996.
[23] G. Oriolo, G. Ulivi, M. Vandittelli, On-line map-building and navigation for autonomous mobile robots, in: Proceedings of the IEEE International Conference on Robotics and Automation, Nagoya, Japan, 1995, pp. 2900–2906.
[24] S.B. Thrun, A. Bücken, Integrating grid-based and topological maps for mobile robot, in: Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96), Portland, OR, 1996, pp. 944–950.
[25] S. Thrun, A. Bücken, Learning maps or indoor mobile robot navigation, Tech. Rept. CMU-CS-96-121, Carnegie Mellon University, Pittsburgh, PA, 1996.
[26] B. Yamauchi, A. Schultz, W. Adams, K. Graves, Integrating map learning, localization and planning in a mobile robot, in: Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, Gaithersburg, MD, 1998, pp. 331–336.
[27] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C—The Art of Scientific Computing, 2nd Edition, Cambridge University Press, Cambridge, 1998.
[28] M.A.P. Idiart, M. Trevisan, Directing a random walker with optimal potentials, Physica A 307 (2002) 52–62.

**Edson Prestes e Silva** received his B.Sc. degree in Computer Science from the Federal University of Pará (UFPa), Brazil, in 1996 and M.Sc. in CS from the Federal University of Rio Grande do Sul (UFRGS), Brazil, in 1999. Since 1999, he is a Ph.D. student at UFRGS. His main field is artificial intelligence applied to mobile robots.



**Paulo M. Engel** received his B.S. degree in Electrical Engineering, in 1978, from the Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brazil, M.Sc. from the Universidade de São Paulo, Brazil, in 1981, and Ph.D. degree from the Technische Universität Munchen, Germany, in 1986. Currently he is a Adjunct Professor from the Institute of Informatics from the UFRGS, where he leads the Artificial Intelligence Group. His main research interests include neural networks, fuzzy systems, pattern recognition, autonomous robotics and data mining.

**Marcelo Trevisan** received his Bachelor's degree in Physics and Computer Science from the Universidade Federal de Santa Maria, and M.Sc. from the Universidade Federal do Rio Grande do Sul, Brazil. He commenced his Ph.D. at the Physics Institute at the Universidade Federal do Rio Grande do Sul early in 2001 and is currently working on his thesis on navigation learning and localization neural systems.

**Marco A.P. Idiart** received his Ph.D. degree in Physics on statistical mechanics of neural networks, in 1992, from the Physics Institute of the Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brazil. He had his post-doctoral training in computational neuroscience on the Center for Complex Systems at Brandeis University, USA. Currently he is a Adjunct Professor in the Physics Institute at UFRGS and his main research interests are neural networks, computational neuroscience and modeling of biological systems.