Journal of Intelligent and Robotic Systems **39:** 243–263, 2004.

Autonomous Learning Architecture for Environmental Mapping

EDSON PRESTES e SILVA Jr.
Centro Universitário La Salle, Av. Victor Barreto, 2288, 92010-000, Canoas, RS, Brazil;
e-mail: prestes@inf.ufrgs.br

MARCO A. P. IDIART and MARCELO TREVISAN
Instituto de Física-UFRGS, P.O. Box 15051, 91501-970 Porto Alegre, RS, Brazil;
e-mail: idiart@if.ufrgs.br, tmarcelo@if.ufrgs.br

PAULO M. ENGEL
Instituto de Informática-UFRGS, P.O. Box 15064, 91501-970 Porto Alegre, RS, Brazil;
e-mail: engel@inf.ufrgs.br

(Received: 4 April 2003; in final form: 21 October 2003)

Abstract. Here we propose an architecture for an autonomous mobile agent that explores while mapping a two-dimensional environment. The map is a discretized model for the localization of obstacles, on top of which a harmonic potential field is computed. The potential field serves as a fundamental link between the modeled (discrete) space and the real (continuous) space where the agent operates. It indicates safe paths towards non-explored regions. Harmonic functions were originally used as global path planners in mobile robotics. In this paper, we extend its functionality to environment exploration. We demonstrate our idea through experimental results obtained using a Nomad 200 robot platform.

Key words: navigation, exploration, mapping of unknown environment, harmonic functions.

1. Introduction

In a rescue operation, or in many tasks where human supervision is restricted and a precise map of the region involved in the task is lacking, exploratory behavior plays an essential role. A truly autonomous robot in these circumstances has to be able to efficiently chart the environment at some degree prior to engaging in any task oriented behavior. Even a purely reactive robot has to keep track of its bearings, otherwise there will be very few tasks where it can be useful.

In this paper we discuss a possible solution for the problem of exploration and mapping of local scale unknown environments. The typical practical task we are interested in solving might be, for instance, the problem of rescuing (by removing or medicating) a victim in the wreckage of a destroyed building. We consider that the extent of damage makes previous maps useless and the situation is too unstable to use human rescuers. In such circumstance there is a pressure for time, therefore the accuracy of the acquired map has to be only up to what is needed for the task.

The sensors are used to give an estimated position of obstacles that are placed in an internal geometric map. The obstacle position is the most important feedback from the environment. Exploration consists of covering a predefined region in space (here in this work, two dimensional space) localizing and incorporating obstacles to a map while searching for a target. We consider that its spatial coordinates completely determine the state of the robot. This is true for the Nomad 200 robot since its rotation and translation are independent. Therefore exploration of real space is akin to a search in state space. Furthermore, since we use a discrete model for the environment, the exploration of space can be reduced to a graph search, where obstacles represent non accessible nodes.

From the AI point of view there are two broad classes of exploratory strategies in graphs: uninformed search and heuristic search. In robotics, to some extent, such categories can be compared to reactive strategies and strategies based on models (Romero et al., 2000). Uninformed or reactive strategies imply that no *a priori* hypothesis about the environment (besides the essential ones like the existence of walls, and so on) are used to drive behavior. Moreover, behavior is not affected by the result of exploration. Heuristic or model exploration, on the other hand, carries such extra information that is used to modify behavior according to the specific distribution of the obstacles found as exploration evolves.

The most widely used reactive approach is the wall-following strategy (Mataric, 1990; Crowley and Coutaz, 1985). It consists of following walls while extracting the contours of obstacles. It can help to generate either topological (Mataric, 1990) or geometrical (Crowley and Coutaz, 1985) maps. Wall-following is not *per se* an exploratory strategy since after the complete contour of an obstacle it does not prescribe how to proceed. Therefore when non-explored walls are not within sensor range, wall-following has to be associated with a true exploratory strategy, like random search or adapted versions of breath-first or depth-first searches.

On the other hand, the strategies based on models use a function that describes which are the preferred actions for a given state, as the core of the exploratory process. This function, sometimes called a heuristic function, a cost function or a reward function, combines current knowledge (obtained by sensors) and some prior knowledge of how things should be in the environment. A heuristic function is expected to change during exploration.

There are many choices for such functions, most of them indicate the shortest path to an unknown region (Yamauchi, 1997; Choset et al., 2000). For instance, the frontier-based exploration of Yamauchi (1997), where the border lines between unknown and known regions, called *frontiers*, drives the robot behavior through a depth-first search. The Fisher information on the measurement and navigational uncertainties can also be used as heuristic functions (Feder et al., 1999). In this case the robot prefers actions that maximize its information gain in Fisher's sense. Similarly Thrun (1998) has devised an algorithm where a heuristic function, computed by *value iteration*, is in charge of producing paths of minimal cost, which indicate where the robot should go to diminish the quantity of unknown regions.

This algorithm, in some cases, is used in combination with the wall-following strategy. The hybrid approach (Romero et al., 2000) keeps the robot around the obstacles while it performs the exploration of the environment.

A successful exploratory behavior, therefore, consists in a good combination of a search procedure with a path planning approach. In this paper we introduce an architecture that harmoniously integrates these two functionalities. At its core there is the harmonic function method introduced by Connolly and Grupen (1993) for path planning. Our main contribution is to show how this idea can be articulated with the functions necessary for exploration and mapping in a efficient architecture.

In Section 2 we present our architecture and in 3 we describe its modules. Section 4 discusses the exploratory behavior produced by the architecture. The results, discussions and conclusions are presented in Sections 5–7, respectively.

2. An AI Approach for an Exploratory Agent

The AI approach considers the robot as an agent that interacts with the environment by executing actions based on information coming from its sensors. An agent corresponds to the union of a physical structure with a program (Russel and Norvig, 1995). The physical structure consists of a computer or a specific hardware that acts on and measures properties of a given environment that can be either real or simulated. The program corresponds to a function that implements the mapping of the perception onto actions.

The agent architecture describes how the different subfunctions, or modules, are organized in the program to produce the final mapping. In the AI literature there are several classical architecture approaches, for instance, NASREM (Albus et al., 1989), LAAS (Alami et al., 1998), Subsumption (Brooks, 1986) and so on. However, there is no widely accepted theory of architecture design that can be used to prove that one design is better than another (Russel and Norvig, 1995). Most of them have overlapping principles, which makes any tentative classification appear rather fuzzy (Medeiros, 1998; Pettersson, 1997).

For an autonomous agent, a convenient organization of functions is the SMPA (Sense-Model-Plan-Act) paradigm (Nilsson, 1998). It can be considered as a broad organizational strategy upon which several architectures are based. In this paradigm, a typical world representation is the *state-space graph*. It represents a discrete set of world configurations that an agent might encounter while moving through its environment and the actions that link every two states. However, in real world application most of requirements demanded by this paradigm are not met by the current state of technology and, in particular, for mobile robots, the real state-space is a continuous space, therefore, the actions in this context should be continuous displacement commands in order to produce smooth paths and, consequently, to reduce odometric errors.

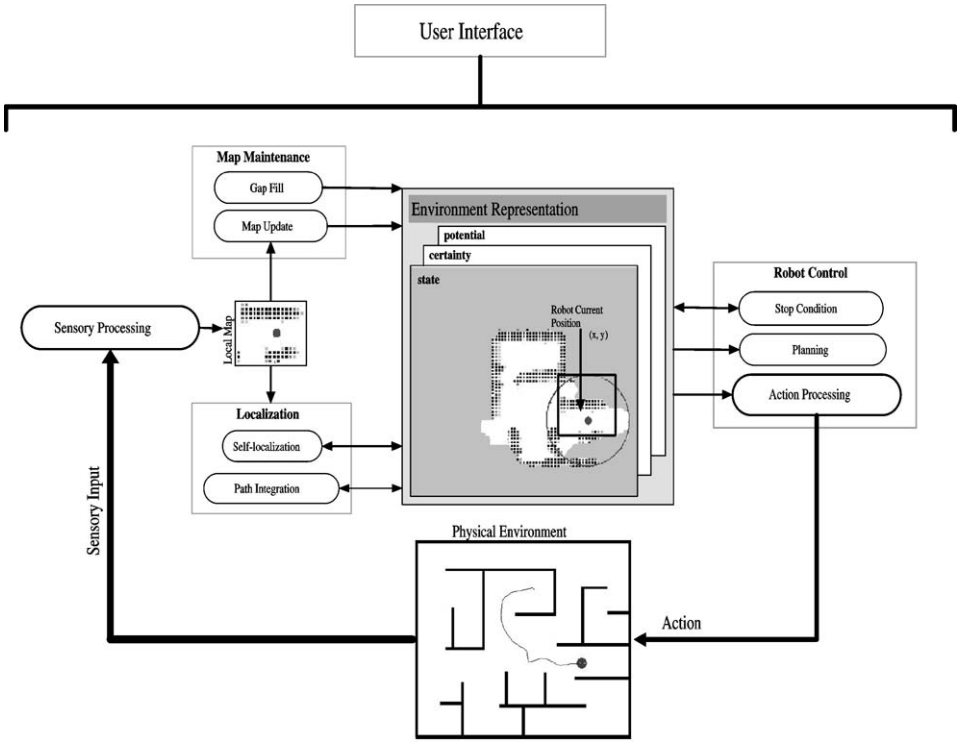


Figure 1. The architecture of our exploratory agent.

Our architecture, illustrated in Figure 1, has an embedded *blackboard* where some modules run concurrent and asynchronously, generating a hybrid architecture – reactive and deliberative at the same time.

The blackboard architecture is composed of a data structure called the *blackboard* and several experts, called knowledge sources (KS) that act independently on the data of the blackboard, updating and checking it for consistency. This architecture became popular because of its flexibility in structuring problem solving. The main idea behind this architecture is that all communication among the modules is made through the blackboard data structure. This makes it easy to add a new KS or to remove an old one whenever necessary. Furthermore, the blackboard allows an incremental and opportunistic solution of the problem. However, its flexibility may make control difficult. Fortunately, this architecture is well-studied and there exist variations of the traditional version (Medeiros, 1998; Pettersson, 1997).

Our architecture uses some concepts of a traditional blackboard such as: modularity, common data structure shared between several KS, and independent activation of a KS. At the highest level, the exploration process can be viewed as a sequential SMPA-like algorithm.* The blackboard structure is used at the level

* Here, different from the traditional forms of SMPA, the action of a given module does not exclude the action of the others. For example, information can be fed into a module while it is computing its task.

of map updating and it can be truly asynchronous. There are several experts that interact with its data components in order to update them or to extract information that is used in the subsequent planning or action phases. These experts perform the following functions:

- For updating the robot's position coordinates:
  - Path integration;
  - Self localization.
- For the maintenance of the world's map:
  - Map update;
  - Gap filling.
- For robot control:
  - Planning;
  - Action processing;
  - Halt instruction.

3. Architecture Description

3.1. ENVIRONMENT REPRESENTATION

The data stored in our blackboard is composed of two complementary and essential components:

- The robot's coordinates in a global frame of reference;
- The grid representation of the world, where the environment properties are stored. The grid (discrete) information can be mapped onto the real (continuous) space using the same global frame of reference.

The agent internally represents the environment using an iconic representation based on an array of $L_x \times L_y$ cells, where each map cell is centered in real world coordinates $\mathbf{r} = (r_i, r_j)$ and corresponds to a real-world square area, storing the following attributes:

- **state** (s_{ij}): indicates the current status of a cell, which can be: *not explored*, *free space* or *occupied*. The attribute *not explored* indicates that the position corresponding to cells of indexes ij has not yet been visited and its potential value is set to 0. The attribute *free space* indicates a visited cell found empty in which the potential can be updated. The attribute *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_{ij}): gives a measure of the probability of finding an obstacle in that cell;
- **potential** (p_{ij}): refers to a heuristic function, in the discrete space-state, that when translated to continuous space indicates navigation paths to the agent.

In robotics, this representation is commonly called a *grid-based map*. Grid-based maps were first designed to cope with the spatial uncertainty generated by sonar sensors (Moravec and Elfes, 1985). The main problems are: granularity, scalability and extensibility. There are several representations based on grids, the main difference among them is the function used to update the cells, for example: fuzzy certainty (Oriolo et al., 1997), Bayesian (Thrun, 1998), frequency (Borestein and Koren, 1991) and so on.

3.2. SENSORY PROCESSING MODULE

In our application there are two major sources of information from the environment:

1. The odometric readings. These are real valued single precision coordinates that indicate the displacement in egocentric bearings.
2. The sonar readings. 16 real value single precision returns from range finder (sonar) sensors.

3.3. MAP UPDATE MODULE

In this paper we employ a method based on frequency of observations called HIMM (Histogramic In-Motion Mapping) proposed by Borenstein and Koren (1991). It uses a linear map update function that counts the number of sensory observations made of each object in the environment. From that it can mark cells that represent the estimated position of obstacles using a certainty criterion. If the rate of positive observations per unit of time exceeds a predefined threshold the cell is marked "occupied" otherwise it is marked "free-space".

This mapping procedure is done during the robot navigation. While in motion, it activates the sonar sensors. Each one of the 16 sonar sensors returns a scalar value $0.4\text{ m} \leqslant d_i \leqslant 6.5\text{ m}$, 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 and position of the sensors, the values d_i are used to update the certainty value c_{ij} in the cells corresponding to the object positions only if they are inside the activation window (see Section 4), i.e., readings from outside the activation window are not used to update the map.

Figure 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.

* These parameters were chosen heuristically based on observations made during the experiments. We reinforce the presence (through the value 3) of an obstacle stronger than its absence (through the value 1), because the latter is easily generated by noisy and more dangerous to the robot navigation.

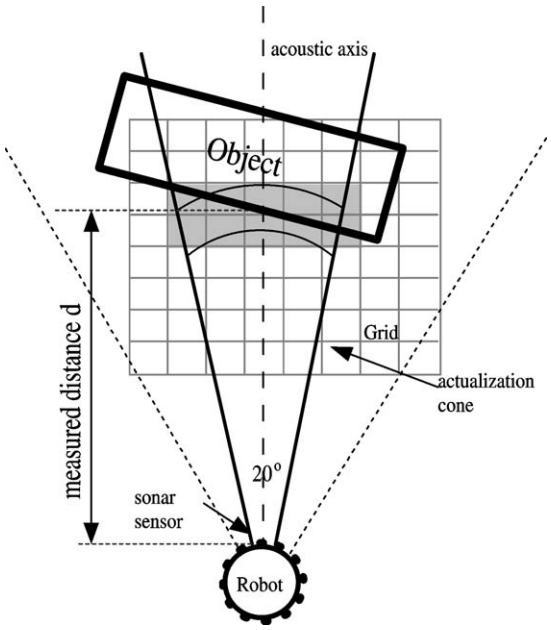


Figure 2. Actualization of the Grid: the figure shows a schematic diagram of an object detection by one of the robot sonar sensors.

Since the observation rate is hidden behind a threshold operation, the HIMM produces a binary version of the map. It does not spread a *pdf* (probability density function) around the obstacles indicating the obstacle hitting probability like the function used by Thrun (1998) and others. However, the association of the binary map with the harmonic potential, described ahead, automatically generates a *pdf* represented by the potential values of the free space cells surrounding the obstacles. Connolly (1994) showed that in the specific case that obstacles are represented by a fixed potential value equal to 1 and the goal by a value of 0, the harmonic potential in a cell gives the probability that a random walker departing from that cell hits an obstacle before hitting the target, the so-called *hitting probability*.

The update is done in two steps. First, a local map is drawn from the current sensor readings inside the activation window. Then the local map is clipped to the global map and the corresponding cells are updated.

Furthermore we make refinements in order to eliminate misclassified cells that reduce exploration performance. For example, a misclassified *free space* cell may indicate passages that do not exist. It causes the robot to visit the place (to correctly classify it) before engaging the correct path. A misclassified *not explored* cell pulls the robot to regions with little gain of relevant environment information. The refinements are done by *gap filling modules* as follows:

– cells classified as *free space* are changed to *occupied* if their immediate neighbors on both sides are occupied so that the robot cannot pass through. Observe that the typical cell size we use is smaller than the robot's diameter. The

properties s_{ij},c_{ij} and p_{ij} changed to *occupied*, maximum value defined by the user and 1, respectively;
- cells classified as *not explored* change to *free space* if most of their neighbor cells are explored (either free space or occupied). The properties s_{ij}, c_{ij} and p_{ij} are changed to *free space*, 0 and 1, respectively.

3.4. LOCALIZATION MODULE

The robot's localization is currently provided by a path integration (dead-reckoning) module. It is not a self-correcting strategy and important errors are introduced by slippage of wheels and faulty readings from the robot's odometers. These errors grow unbounded as the robot explores and become visible in large environments.

In order to avoid this error, pure odometric information has to be combined with a self-localization strategy. It is ordinarily based on a comparison between a supposedly more accurate global map and the partial map freshly extracted from the sensors (Fox et al., 1999; Olson, 1997; Olson and Matthies, 1998; Olson, 1999; Schultz et al., 1999; Schiele and Crowley, 1994).

In this work we do not implement a self-localization module. Through careful control of the accelerations of the robot and due the natural smoothness of the paths generated by the potential field we manage to keep the odometric errors within acceptable levels as shown in Section 5. Abrupt changes in the robot's direction are usually the main source of odometric errors. Insofar as odometric errors are low, unexpected events like hits with an undetectable object or a movable object can still be very damaging to our technique without an appropriate self-localization module.

3.5. PLANNING MODULE

We were firstly motivated by the work developed by Connolly and Grupen (1993) for path planning in an environment where objects and walls are known with accuracy. In their method, once a target position is defined, a potential field is calculated throughout the environment. After its convergence, the gradient descent on the potential steers the robot around obstacles toward the goal.

Potential fields are a known technique in path planning (Latombe, 1993; Barraquand et al., 1992; Rimon and Koditschek, 1992) 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 $p_i(\mathbf{r}) = -\int \mathbf{f} \cdot \mathrm{d}\mathbf{r}$, 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. Especially 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 becomes trapped.

The breakthrough in Connolly and Grupen's method is the use of *harmonic functions* for the potential field. Harmonic functions are the solutions of Laplace's equation

$$\nabla^2 p(\mathbf{r}) = \frac{\partial^2 p(\mathbf{r})}{\partial x^2} + \frac{\partial^2 p(\mathbf{r})}{\partial y^2} = 0. \tag{1}$$

They do not present local minima (Courant and Hilbert, 1962) except for the boundaries defined by the obstacles and goals.

The harmonic functions can have many interpretations. They can be viewed as the result of a principle of minimal energy or least action. They can have a probabilistic meaning like the hitting probability, or the probability that a random walker departing from a given cell hits an obstacle before hitting the desired target. They can come out of learning procedures as temporal differences (Sutton, 1988; Connolly, 1994) or the Monte Carlo method for reinforcement learning (Sutton and Barto, 1998).

In the grid, the environment cells that show high probability to have an obstacle store a potential value equal to 1, whereas cells that contain the target have their potential value set to 0. Obstacles and the goal are regarded as boundary conditions of a Laplace's boundary value problem (BVP).

The harmonic potential is calculated numerically by the relaxation method similar to *value iteration* (Sutton and Barto, 1998) with computational cost $\mathcal{O}(n^2)$, where n is grid size. This method interpolates the potential value between the obstacles and the target. Among the several relaxation methods, we have chosen the Gauss–Seidel (GS) because of it generates stable and smooth trajectories as shown in Prestes et al. (2002). The GS method updates the map cells according to the iteration rule:

$$p_{i,j}^{\text{new}} = \frac{1}{4}\left(p_{i-1,j}^{\text{new}} + p_{i+1,j}^{\text{old}} + p_{i,j-1}^{\text{new}} + p_{i,j+1}^{\text{old}}\right). \tag{2}$$

This algorithm is an example of an anytime algorithm, whose output quality is gradually improved over time, so that is has a reasonable decision ready whenever it is interrupted. We use this flexibility to obtain reasonable gradient values before total relaxation of the potential. This saves us some computation time and makes the cost of calculating the path effectively $\sigma(rn)$, where r is the number of relaxation steps prior a gradient calculation.

3.6. ACTION MODULE

Actions are extracted from the potential part of the grid data structure. The negative gradient is computed on the cell containing the current position of the robot. It is

a discrete gradient calculation that results in a continuous valued vector. The robot adjusts its head direction to the angle θ given by

$$\theta = \arctan(p_{i-1,j} - p_{i+1,j}, p_{i,j-1} - p_{i,j+1}), \tag{3}$$

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

The speed v is calculated as follows:

$$v_t = \eta \hat{v}_t + (1 - \eta) v_{t-1}, \tag{4}$$

where v_{t-1} is the speed at instant $(t - 1)$; $\hat{v}_t$ is the estimated value of the robot speed at instant t and η defines a smooth transition between subsequent speeds. All experiments use $\eta = 0, 5$. The estimated speed $\hat{v}_t$ is based on the rotation to be performed by the robot, where $\hat{v}_t$ decrease linearly with $\Delta\theta$. $\hat{v}_t$ is calculated by

$$\hat{v} = v_{\max}\left(0.2 + 0.8\frac{90 - |\Delta\theta|}{90}\right), \tag{5}$$

where $v_{\max} = 25.4$ cm/s. The angular correction $\Delta\theta$ is the difference between the robot's orientation and the negative potential gradient direction θ for the current position, reduced to the first quadrant. Therefore its maximal absolute value is 90 degrees, and any turn exceeding 90 degrees in absolute value is implemented as a combination of a $\Delta\theta$ turn and the reversion of the wheels' translation.

The functional form of (5) and its parameter values (0.2 and 0.8) were chosen heuristically to cause a reasonably smooth motion. Equation (5) reduces the robot's speed during its rotation by controlling the curvature of its path.

The factor 0.8 indicates that 80% of the robot speed will be controlled by the $\Delta\theta$, i.e., the bigger the variation the lower the robot speed. However, this speed is near zero when this variation approaches 90 degrees. In this case, we need a ground speed to keep the robot in movement. That is the idea behind the factor 0.2. The factor 0.2 guarantees a ground speed, which corresponds to 20% of the maximum speed, to the robot when it needs to rotate an angle near 90 degrees.

3.7. HALT MODULE

The *halt* module can stop the exploration process in two ways depending on the task. In the case of exploration and mapping of an unknown environment, the halt instruction is based on a segmentation image algorithm, called *region growing*, which runs concurrently with the update process. First, this module extracts the free space and checks frontiers (borderline between unknown region and free space), identifying if there exists any free-space cell near a *not explored* cell. If this condition is false then exploration ends, otherwise, the process continues. This strategy is more robust than considering that the process is supposed to stop when all cells have their attributes ($s_{ij} \neq$ *not explored*), because, depending on the environment

shape, a subset of map cells is never reached and the algorithm fails. Furthermore, it automatically eliminates phantom frontiers caused by specular reflection or changes in the environment, like a door that is closed. For exploration in search of a target the process is similar except that it stops the robot when this module infers that the target is attained.

4. Exploratory Behavior

Exploratory behavior arises naturally from the dynamics of the attributes of the environment representation. Not explored cells are equivalent to goal cells, with the difference that they change status when inside the sensor range. This strategy generates an autonomous exploratory behavior since the goals are placed on the frontiers and they recede as the space is explored. The robot follows the non-explored cells until none of them are left.

The exploration process starts by scaling the internal grid so that it can represent the largest environment to which the agent will be exposed. Initially, the agent is in a zero knowledge state, thus all environment cells are classified as *not explored cells*. As the agent moves, it assigns the visited cells with the appropriate state, according to the response of its sensors.

The agent has range finder sensors that detect obstacles in the environment. These sensors feed to the *perceptual processing module* (see Figure 1), which produces a region called the *activation window* that has roughly the size of its average sensor range. This is used to update the internal representation of the agent and to indicate the grid cells that will be recruited for update, i.e., if a cell is at a given time in the activation window, it becomes part of the explored space by participating in the harmonic potential calculation at all times. We call *activated region* or *potential region* the set of activated cells that comprises the explored space. The borderline of the activated region corresponds to the limit between the known and unknown regions of the environment.

The harmonic potential for the current explored (activated) region is calculated with the not-explored cells as temporary targets. Note that the target potential value is equal to 0 (see Section 3). These cells will attract the agent and when they are reached, they change to occupied or free space depending on their sensor's response, therefore becoming part of the *activated region*. Consequently, other *not explored cells* will become attractive and the process runs continuously.

The *action* generated by the architecture is the displacement of the agent following the negative potential gradient at the current position in the internal map, which initially corresponds to the center of the activation window. In this case the gradient guides the robot to the largest unknown region or the closest unknown regions depending on which is more attractive. 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 agent, it simply follows the forward direction.

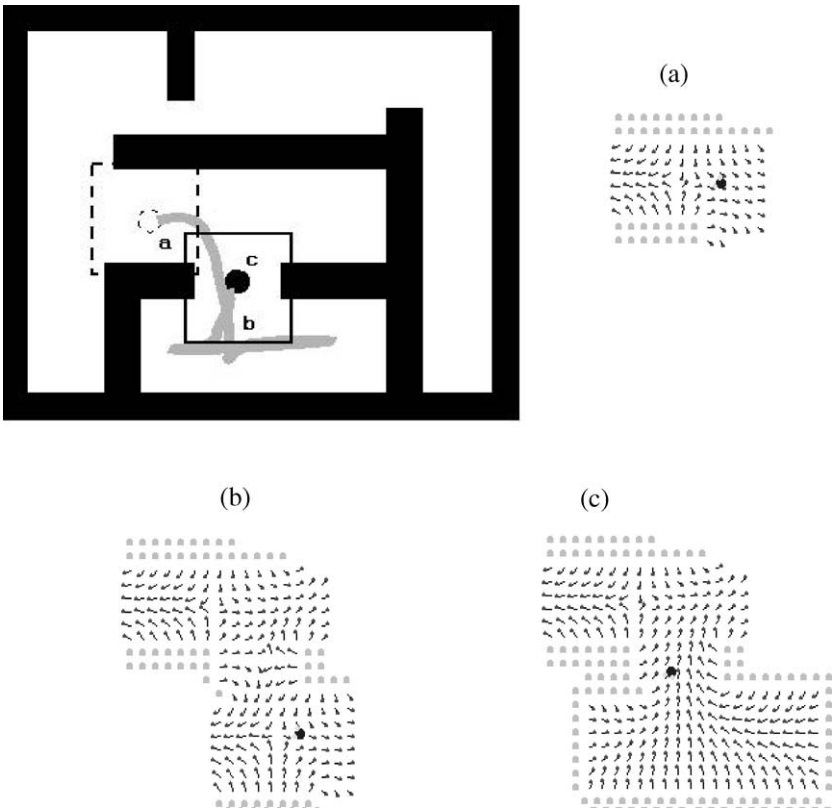


Figure 3. Exploration process: On the top the trajectory followed by the agent. (a)–(c) show snapshots of the field and grid configurations for each corresponding position in the trajectory.

Figure 3 illustrates the exploratory behavior generated by our controller. The agent sequentially occupies the positions a, b and c, with the final position indicated by the black circle. Figure 3 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 Figure 3(c) we see how the vector field changes to pull the agent back from a dead-end.

The robot performs the exploration strategy summarized below:

1. Sets its position to $(0, 0)^{\star}$ and the state, certainty and potential values of all cells to *not explored*, 0 and 0, respectively.
2. Activates and reads the sonar sensors (see Section 3.3).
3. Performs a local update of the map inside the activation window (see Section 3.3).
4. Updates the harmonic potential over the potential region** (see Section 3.5).

* The position (0, 0) corresponds to the center of the 2D grid map.

** A single actualization of the potential field consists of the update of all grid points inside the potential region. The procedure is repeated a number of times before the robot moves.

5. Calculates the gradient vector at the current position.
6. Moves following the direction defined by the gradient descent on the potential.
7. If all environment is explored go to 8 else go to 2.
8. Stop exploration.

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 the calculation shown in Section 3.6.

5. Results

In this section we present the results obtained using the Nomad 200 robot. The experiments were performed on a Celeron 475 MHz running Linux over a radio Ethernet. All the experiments had a grid with maximum size equal to 400×400 cells, where each cell corresponds to a squared region of 15 cm $\times$ 15 cm in real space. Furthermore, we consider an iteration rate of 20 iterations/sec and robot vision radius equal to 2 m.

5.1. ENVIRONMENTAL EXPLORATION

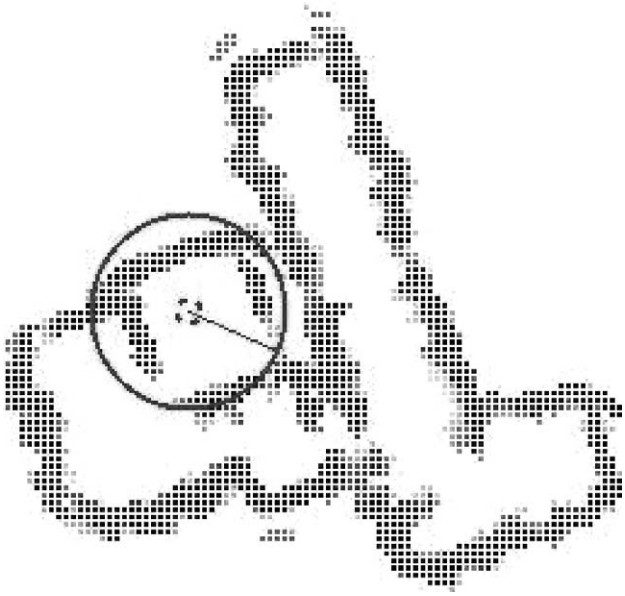
Figure 4 shows an experiment performed with the NOMAD 200 robot, where it explores an office environment in our institute. This environment consists of a room divided by two cardboard walls connected to a L-shaped corridor. The room has tables and chairs placed along the walls. In this experiment, the robot traveled a total of 36.6 m to explore the environment in approximately 5 min, with an average speed of 12.7 cm/s. The activation window is shown in the figure as a circle around the robot.

Figure 5 shows another environment configuration built in our institute. It corresponds to a small maze with approximately 8.9 m $\times$ 19.8 m, with 3 rooms divided by cardboard. The thin line corresponds to the path followed by the robot Nomad 200 during the exploratory process. Table I shows the average result of 4 experiments in the same environment with the robot starting at different initial positions.

5.2. ODOMETRIC ERRORS

One of the factors that generates odometric errors is a sudden change in the direction of motion. In our experiments, this factor is kept within acceptable limits because:

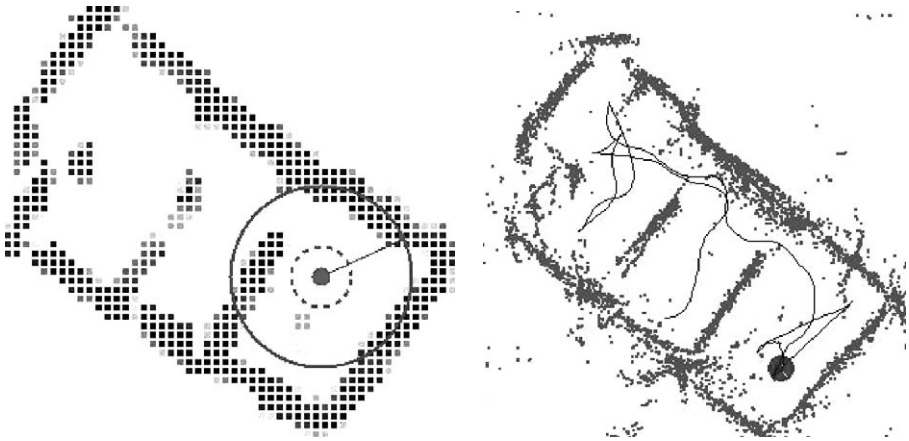
- the robot movement is computed from a continuous gradient of harmonic potential, which generates smooth trajectories automatically;
- the refined control of speed (see Section 3.6), which avoids jerky movements.



(a)

(b)

Figure 4. Exploration of an office environment by the Nomad 200 robot. (a) Grid map obtained after exploration. (b) The trajectory followed during exploration. The sonar readings are also shown. Observe that they are noisier than the map. This is due to the fact that readings outside the activation windows are discarded and only consistent readings update the certainty attribute.



(a) (b)

Figure 5. Exploration experiment with the Nomad 200 robot in a cardboard maze.

Table I. Average results for 4 exploration experiments in the maze in Figure 5 with the Nomad 200 robot

Relax. method	Iter. rate	$\bar{\ell}$	σ_ℓ
Gauss–Seidel	20	13.68 m	0.62 m

To demonstrate this we performed five experiments to evaluated the odometric errors. In these experiments the robot explores the entire environment from the same initial position spotted on the floor. The robot returns to the initial position after complete the exploration.

Each cell of the map represents a square area of 10.1 cm $\times$ 10.1 cm of the real environment. The robot moves with maximum speed of 25.4 cm/s and with an activation window of 2 m.

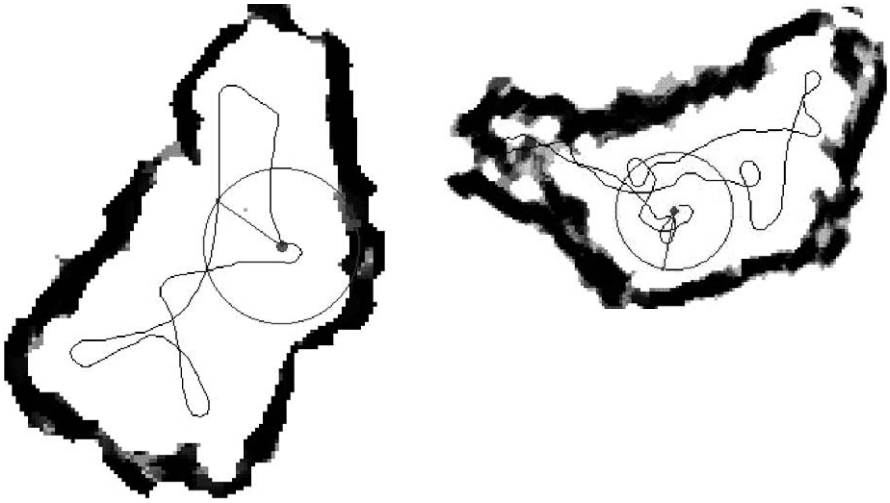
Due to discretization of the environment and communication delay between the robot and its base computer we define a circular region of 25 cm radius around the initial position where the robot has to return after the exploration. The distance between the center of the robot after returning and the its initial position is called discrepancy. For practical purposes, any discrepancy $<$25 cm indicates perfect home return and therefore insignificant odometric errors.

Table II shows the average ($\bar{\zeta}$) and standard deviations (σ_ζ) of the discrepancy, and the average ($\bar{\ell}$) and standard deviations (σ_ℓ) of the path length followed by the robot.

The average discrepancy is smaller than the radius (25 cm) of the circular region around the initial position previously defined. It represents approximately 0.6% of the total path length followed by the robot. In the experiment illustrated in Figure 6,

Table II. Discrepancy and average path length followed by the robot during 5 experiments

$\bar{\ell}$	σ_ℓ	$\bar{\zeta}$	σ_ζ
31.7 m	9.1 m	22 cm	2.7 cm



(a) (b)

Figure 6. Robot exploration in our Institute. (a) Environmental map produced and the path followed by the robot returning to its start position. (b) Environmental exploration with dynamic obstacles.

the robot followed a trajectory of 24.35 m with a discrepancy of 20 cm. Figure 6(b) shows a situation where the robot had to dodge dynamic obstacles (people). The path length was 44.0 m with a discrepancy of 20 cm.

6. Discussion

The efficiency of an exploration algorithm depends on a series of factors, among them,

- the length of the exploration paths that it produces,
- the smoothness of the paths,
- the computation time to generate such solutions.

The technique as a whole also depends on the subsidiary ingredients such as the map representations, the interpretation of the sensory readings, etc. Therefore to decide if a technique is locally efficient or globally efficient all these factors have to be taken into account.

The potential technique to generate the exploratory paths is certainly globally efficient if we consider the path length without worrying about its computational cost.

Its computation time, however, imposes some limitations on the size of the environment as the number of grid points increases. However, it is weakly dependent on the complexity of the environment since the associated relaxation algorithm depends on the number of cells in the grid and not on the specific placement of obstacles on it. Bug like approaches (Lumelsky and Skewis, 1990; Lumelsky and Stepanov, 1986), for example, focus on completeness rather than optimality and tend to deteriorate as the intricacy of the environment mounts. They guarantee to find a path if it exists but pay little attention to minimizing its length. Our method produces short and smooth exploratory paths reducing significantly the odometric errors and the hitting probability in the presence of noise. This is an advantage over cell decomposition methods or wavefront methods (Batavia and Nourbakhsh, 2000; Connolly and Grupen, 1993; Connolly, 1994). They produce jagged paths that are difficult to follow by the robot due to its limited mobility.

The computation time for the potential field is not to be viewed as a limitation of the architecture but as an open and promising research front.

The original harmonic potential approach runs the update process several times until a relaxed version of the map is produced. This is done mainly because they aim to compute a path between two positions based on a previously known static global map of the environment. This process can be computationally expensive, because cells far from the goal position suffer little attraction, consequently, they take many iterations to relax. In our case, we build the map incrementally. While this is done, the potential must be calculated many times. Fortunately, we found that complete relaxation of the potential is not required for a good and smooth performance (Prestes et al., 2001, 2002). Most of the time the robot is found near a boundary, which makes relaxation of its driving local potential very fast. A few iterations are enough. Furthermore, we showed that Gauss–Seidel (GS) updates generate more stable vector fields than Successive Over-Relaxation (SOR) updates for a partially relaxed version of the potential (Prestes et al., 2002). Presently we are running the algorithm in a notebook Athlon 1.2 GHz, and the update cycle takes on average 30 milliseconds to calculate the potential in a 100×100 grid, which is reasonable for the typical speeds of the Nomad robot ($\sim$20 cm/s).

We are also developing methods that further improve the computation time by switching between local and global computation (Trevisan et al., 2002). It is also important to remember the perspective of having hardware specially designed to make harmonic potential calculations (Tarassenko and Blake, 1991; Marshall and Tarassenko, 1994), in which case it can be reduced by many orders of magnitude.

The other important factor that limits the use of the method in large environments is the sometimes poor quality of the map generated from sonar readings, plus the accumulation of odometric errors that is prone to occur for long explorations.

Both limitations, however, can also be reduced by better hardware, for example, a laser range finder or an improved odometric control.

7. Conclusions

In our architecture, an iconic representation of the environment is stored as the exploration takes place. The representation is a discrete model of the position of obstacles and candidate goals. It can be thought of as a graph of position states, i.e., each node corresponds to a position in real space. In the particular example we treated the graph as a square lattice and the coordinates are implicitly stored by the array index of the nodes. In a more general situation the coordinates can be explicitly stored as an attribute of the node. Besides position, the nodes can store a set of other properties called states. The states describe the characteristics of the environment in that particular node, its accessibility, its status in the exploration process, if there is a target present, etc.

A central role in the path planning aspect of our exploration method is played by the potential values at the nodes. The potential is equivalent to a value function that indicates the success or the amount of expected reward (in a reinforcement learning language) taken by the agent by the choice of a given action. The solution of the discrete version of Laplace's equation for a set of obstacles and goals automatically generates a value function that accounts for obstacle collision (Connolly and Grupen, 1993; Connolly, 1994).

One of the great advantages of this potential calculation is the smooth trajectories generated from following the gradient of the potential. This smoothness is responsible for the small odometric errors obtained in the experiments. Probably, the errors are even smaller because we defined a circular region of 25 cm radius around the initial position where the robot had to return and stop after the exploration. The robot stopped when this region was reached even if the end position was not the exactly initial position. Another important property of this approach is the fact that information is shared globally once a new feature of the environment is discovered. The iteration process communicates changes from cell to cell in a diffusion-like process.

In this framework there is much room to improve the complexity and the accuracy of the agent's behavior towards the environment. Most of what can be done involves operation on the potential values at the nodes, for example:

- There is a family of potentials that can be used besides harmonic functions that do not present local minima. Although they might not have the collision interpretation of harmonic functions they can be designed to produce specific behaviors (Prestes et al., 2002).
- Harmonic functions are rapidly decaying functions that 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 (Idiart and Trevisan, 2002).

- The boundary conditions can be modified in a way that not all non-explored cells become attractive, but only specific combinations of them. Following some features of the boundaries the robot can mimic a series of known behaviors like wall-following, or space-filling strategies (Trevisan et al., 2002).
- The grid representation can be thought of as a discrete rough model for the environment on top of which continuous actions can be computed. The robot moves in real space but its actions are planned from discrete gradients taken on the potential values on the graph. This introduces an extra uncertainty in the order of the cell size. We can reduce this uncertainty if we go to finer grids in crowded regions of the space while sparsely representing the empty regions. Another possible solution is to use more efficient grid sampling, like quad-trees, to account for non-homogeneous distributions of objects (Zelek, 1998).
- Expected hypotheses about the environment can be introduced via the expert systems that operate on the blackboard to help to fill the gaps left by incomplete or faulty sensor readings.
- The blackboard representation is easily extensible. In the near future we intend to add modules to asynchronously handle voice recognition and image processing.

The results reported here demonstrate a simple and coherent principle used as the core of an architecture robustly to guide exploration in a real environment. Furthermore, this extension of the functionality of the harmonic functions provides the basic idea for the development of extensions of current path planners based on potential functions in order to generate integrated systems, where specific modules aggregate several functions. The main advantage is the unified understanding of the problems related to exploration and path planning.

Acknowledgement

This work has been partially supported by Brazilian agencies CNPq, FINEP and FAPERGS.

References

Alami, R., Chatila, R., Fleury, S., Ghallab, M., and Ingrand, F. F.: 1998, An architecture for autonomy, *Internat. J. Robotics Res*.

Albus, J. S., McCain, H. G., and Lumia, R.: 1989, Nasa/nbs standard reference model for tele-robot control system architecture (nasren). Technical Report 1235, National Institute of Standards and Technology.

Barraquand, J., Langlois, B., and Latombe, J. C.: 1992, Numerical potential field techniques for robot path planning, *IEEE Trans. Systems Man Cybernet.* **22**(2), 224–241.

Batavia, P. and Nourbakhsh, I.: 2000, Path planning for the cye robot, in: *Proc. of the IEEE/RSJ Internat. Conf. on Intelligent Robots and Systems*, Takamatsu, Japan.

Borenstein, J. and Koren, Y.: 1991, Histogramic in-motion mapping for mobile robot obstacle avoidance, *IEEE J. Roboitcs Automat.* **7**(4), 535–539.

Brooks, R. A.: 1986, A robust layered control system for a mobile robot, *IEEE J. Robotics Automat.*

Choset, H., Walker, S., Eiamsa-Ard, K., and Burdick, J.: 2000, Sensor-based exploration: Incremental construction of the hierarchical generalized Voronoi graph, *Internat. J. Robotics Res.* **19**(2), 126–148.

Connolly, C. I.: 1994, Harmonic functions and collision probabilities. *Internat. J. Robotic Res.* 497–507.

Connolly, C. I. and Grupen, R. A.: 1993, On the application of harmonic functions to robotics, *J. Robotic Systems* **10**, 931–946.

Courant, R. and Hilbert, D.: 1962, *Methods of Mathematical Physics*, Vol. 2, Wiley, New York.

Crowley, J. L. and Coutaz, J.: 1985, Navigation et modélisation pour un robot mobile, Technical Report 28, Laboratoire d'informatique fondamentale et d'intelligence artificielle, Grenoble, France.

Feder, H. J. S., Leonard, J. J., and Smith, C. M.: 1999, Adaptive mobile robot navigation and mapping, *Internat. J. Robotics Res.* **18**(7), 650–668.

Fox, D., Burgard, W., and Thrun, S.: 1999, Probabilistic methods for mobile robot mapping, in: *Proc. of the IJCAI Workshop on Adaptive Spatial Representations of Dynamic Environments*.

Idiart, M. A. P. and Trevisan, M.: 2002, Directing a random walker with optimal potentials *Phys A: Stat. Mech. Appl.* **307**, 52–62.

Latombe, J.-C.: 1993, *Robot Motion Planning*, Kluwer Academic, Norwell, MA.

Lumelsky, V. and Skewis, T.: 1990, Incorporatin range sensing in the robot navigation function, *IEEE Trans. Systems Man Cybernet.* **20**(5), 1058–1069.

Lumelsky, V. and Stepanov, A.: 1986, Dynamic path planning for a mobile automaton with limited information on the environment, *IEEE Trans. Automat. Control*, 1058–1063.

Marshall, G. F. and Tarassenko, L.: 1994, Robot path planning using VLSI resistive grid, *IEEE Proc. Vis. Image Signal Process* **141**(4), 267–272.

Mataric, M. J.: 1990, A model for distributed mobile robot environment learning and navigation, Master thesis, MIT Artificial Intelligence Laboratory.

Medeiros, A. A. D.: 1998, A survey of control architectures for autonomous mobile robots, *J. Brazilian Computer Soc.* **4**(3).

Moravec, H. P. and Elfes, A.: 1985, High resolution maps from wide angle sonar, in: *Proc. of IEEE Internat. Conf. of Robotics and Automation*.

Nilsson, N. J.: 1998, *Artificial Intelligence: A New Synthesis*, Morgan Kaufmann.

Olson, C. F.: 1997, Mobile robot self-localization by iconic matching of range maps, in: *Proc. of Internat. Conf. on Advanced Robotics*, pp. 447–452.

Olson, C. F.: 1999, Subpixel localization and uncertainty estimation using occupancy grids, in: *Proc. of the IEEE Internat. Conf. on Robotics and Automation, ICRA*, pp. 447–452.

Olson, C. F. and Matthies, L. H.: 1998, Maximum likelihood rover localization by matching range maps, in: *Proc. of Internat. Conf. on Advanced Robotics*, pp. 272–277.

Oriolo, G., Ulivi, G., and Vandittelli, M.: 1997, Fuzzy maps: A new tool for mobile robot perception and planning, *J. Robotic Systems* **14**(3), 179–197.

Pettersson, L.: 1997, Control system architecture for autonomous agents, Technical Report, Department of Machine Design, KTH.

Prestes, E., Engel, P. M., Trevisan, M., and Idiart, M. A. P.: 2002, Exploration method using harmonic functions, *Robotics Autonom. Systems* **40**, 25–42.

Prestes, E., Idiart, M. A. P., Engel, P. M., and Trevisan, M.: 2001, Exploration technique using potential fields calculated from relaxation methods, in: *Proc. of the IEEE/RSJ Internat. Conf. on Intelligent Robots and Systems, IROS*, Havaí, EUA, October, pp. 2012–2017.

Prestes, E., Idiart, M. A. P., Engel, P. M., and Trevisan, M.: 2002, Oriented exploration in non-oriented sparse environment, in: *Proc. of the IEEE/RSJ Internat. Conf. on Intelligent Robots and Systems, IROS*, Lausanne, Switzerland, October, pp. 2353–2358.

Rao, N. S. V., Kareti, S., Shi, W., and Iyengar, S. S.: 1993, Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms, Technical Report ORNL/TM-12410, Oak Ridge National Laboratory, Oak Ridge, TN.

Rimon, E. and Koditschek, D. E.: 1992, Exact robot navigation using artificial potential functions, *IEEE Trans. Robotic Autonom. Systems* **8**(5), 501–517.

Romero, L., Morales, E., and Sucar, E.: 2000, A robust exploration and navigation approach for indoor mobile robots merging local and global strategies, in: M. C. Monard and J. S. Sichman (eds), *IBERAMIA-SBIA: Advances in Artificial Intelligence*, Springer, Berlin, pp. 389–398.

Russell, S. and Norvig, P.: 1995, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, Englewood Cliffs, NJ.

Sabersky, R., Acosta, A., and Hauptmann, E.: 1971, *Fluid Flow*, MacMillan, New York.

Schiele, B. and Crowley, J. L.: 1994, A comparation of position estimation techniques using occupancy grids, *Robotics Autonom. Systems* **12**, 163–171.

Schultz, A. C., Adams, W., Yamauchi, B. and Jones, M.: 1999, Unifying exploration, localization, navigation and planning through a common representation, *Autonom. Robots* **6**(3), 293–308.

Sutton, R. S.: 1988, Learning to predict by the method of temporal differences, *Machine Learning* **3**, 9–44.

Sutton, R. and Barto, A. G.: 1998, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA.

Tarassenko, L. and Blake, A.: 1991, Analogue computation of collision-free paths, in: *Proc. of IEEE Internat. Conf. on Robotics and Automation*, Sacramento, April, pp. 540–545.

Thrun, S.: 1998, Learning maps for indoor mobile robot navigation, *Artificial Intelligence* **99**(1), 21–71.

Trevisan, M., Idiart, M. A. P., Silva Jr., E. P., and Engel, P. M.: 2002, Principles of exploratory navigation based on dynamical boundary value problems, *IEEE Trans. Robotics Automat*. submitted.

Yamauchi, B.: 1997, A frontier based exploration for autonomous exploration, in: *Proc. IEEE Internat. Symp. on Comp. Intel. in Robotics and Automation*, Monterey, CA, pp. 146–151.

Yamauchi, B., Schultz, A., Adams, W., and Graves, K.: 1998, Integrating map learning, localization and planning in a mobile robot, in: *Proc. of the IEEE Internat. Symposium on Computational Intelligence in Robotics and Automation*, Gaithersburg, pp. 331–336.

Zelek, J. S.: 1998, A framework for mobile robot concurrent path planning and execution in incomplete and uncertain environments, in: *Proc. of AIPS-98 Workshop on Integrating Planning, Scheduling and Execution in Dynamic and Uncertain Environments*, Pittsburgh, PA.