Path Planning via Skeletons on Grey-Level Maps

Giuseppe Oriolo* Giovanni Ulivi† Marilena Vendittelli*

* Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza"
via Eudossiana 18, 00184 Roma, Italy
{oriolo,vendittelli}@labrob.ing.uniroma1.it

† Dipartimento di Meccanica e Automatica, Terza Università di Roma
via Segré 2, 00146 Roma, Italy
ulivi@labrob.ing.uniroma1.it

Abstract

We present algorithms for robot motion planning on gray-level bitmaps. In this kind of maps, a number is associated to each cell, characterizing the possibility that it belongs to an obstacle. Such a representation is often used to deal with uncertainty when an unknown environment is reconstructed from sensor readings. Our approach proceeds from the well-known work of Barraquand and Latombe on navigation functions for deterministic maps. In particular, two methods are devised: both are based on a proper modification of the wavefront expansion *algorithm. The first is relatively simple, but produces paths passing close to the dangerous areas. Instead, the second method makes use of a properly defined* skeleton *of the map, that lies as far as possible from such areas. The performance of the two methods is illustrated by application to both simulated and experimental maps.*

1 Introduction

Research activity on mobile robots has built up momentum in the last decade, as witnessed by a constantly growing literature, see e.g. [1–5]. The central topic of this investigation is *autonomy*, the capability of planning and executing motion tasks without human guidance. In order to achieve a reasonable degree of autonomy, it is necessary to properly integrate *sensing* and *intelligence*. On-board sensory systems allow to gather information about the environment in the absence of exogenous knowledge. Machine intelligence is then needed for building a convenient representation of the environment based on sensor data, as well as for planning robot actions in accordance.

In principle, the most favorable situation is realized when the robot is endowed with sophisticated sensors and high computing power. On the other hand, to make the use of robots appealing in real-life applications, it is necessary to reach a trade-off between costs and benefits. In many cases, this prevents the use of expensive sensors (e.g., video cameras) in favor of cheaper sensing devices. Smart algorithms are then needed to extract significant information from data that may be insufficient or conflicting.

Often, rather than trying to reconstruct a deterministic model of the environment, one may adopt an intrinsically uncertain representation. In particular, *gray-level bitmaps* are a convenient choice. The workspace is discretized into a cell grid, and a number (*risk*) is associated to each cell, characterizing the possibility that it belongs to an obstacle.

In the literature, there are basically two approaches for building this kind of maps from sensor readings. The first makes use of stochastic techniques from probability theory [6–8], while the second is based on fuzzy logic [9, 10]. In both approaches, it is assumed that the robot is equipped with ultrasonic rangefinders. These sensors are relatively low-cost and easy to use, but their performance is affected by various phenomena [11]. First, the wide radiation angle of the ultrasonic beam makes it difficult to determine the exact position of the obstacle that originated the echo. Second, the occurrence of *specular* reflections for large angles of incidence may even prevent the detection of some obstacles.

In this paper, we shall present algorithms for planning safe robot paths on gray-level maps, independently from the chosen map building method. Our approach proceeds from the classical work of Barraquand and Latombe [12] on navigation functions for deter-

ministic maps. However, being the representation of the environment uncertain in nature, it is not possible to directly apply their algorithms, as these rely on a sharp distinction between the *free* space and the obstacles.

To overcome the above limitation, we shall introduce a proper modification of the *wavefront expansion* algorithm. Based on this, two methods are devised: the first is quite simple, and produces minimum-length paths inside constant-risk regions. However, the obtained paths typically pass close to the dangerous areas of the map. Instead, the second method makes use of a *skeleton* of the map, with the property of lying as far as possible from such areas. As a consequence, it provides paths that maximize the obstacle clearance. The satisfactory performance of the two methods will be illustrated by application to a simulated map as well as to a map built from experimental data using fuzzy logic.

2 A navigation function for gray-level maps

Assume that a gray-level bitmap $\mathcal{M}$ is given. For simplicity, we shall refer to the two-dimensional case, in which $\mathcal{M}$ is a grid of $p \times q$ square cells of side δ. Each cell C of $\mathcal{M}$ is labeled by a real number $\mu(C) \in [0,1]$, that represents the risk associated with the cell and has been computed on the basis of sensor measures.

A particular instance of the path planning problem is specified by a start cell S and a goal cell G, that identify respectively the initial and the desired final position of the robot. Our analysis will be limited to the case of a 'point' robot, but it is easy to build an *augmented* gray-level map $\mathcal{M}_a$ to keep into account the actual robot dimensions [10].

A natural approach, suggested by the nature itself of the map $\mathcal{M}$, would be to consider $\mu(C)$ as a repulsive potential. By superposing to $\mu(C)$ an attractive field centered at G, we obtain an artificial potential field that could be used for planning. A simple way to build an attractive field on a cell grid is given by the *wavefront expansion* algorithm [12], that sets to zero the value of the potential at G, to 1 at all 1-neighbors[1] of G, to 2 at all 1-neighbors of these cells that have not been visited yet, and so on. The resulting algorithm [13] is very simple, but it displays the typical drawback of potential field methods, namely the presence of local minima in the total field.

[1] In general, the *m-neighbors* of a cell C are defined as those cells having at most m coordinates differing from those of C, with the amount of difference exactly equal to $\pm\delta$, being δ the cell side. In $\mathbb{R}^2$, each cell has four 1-neighbors.

Instead, if the wavefront expansion algorithm is applied to a *deterministic* map (i.e., a map in which cells are labeled with either 0 —empty— or 1 —occupied), the resulting field has no local minima. A grid potential with this property is called a *numerical navigation function*, and provides a powerful planning method [12]. The absence of local minima is guaranteed by the fact that the wavefront expansion takes place in the *free* space only—namely, cells labeled with 0. Such a partition does not hold in our case, because of the uncertain nature of the environment map.

However, it is possible to obtain a behavior similar to that of the original method, by slowing down the wavefront expansion in correspondence of unsafe cells, characterized by high values of μ. Below, we give an algorithm that implements this idea of 'propagating' the risk in order to compute a navigation function U_1.

Algorithm NF1

```
begin
  for every cell C ∈ M do
    U1(C) ← ∞;
  U1(G) ← 0;
  insert G in L;
  while L is not empty do
   begin
     C ← FIRST(L);
     for every 1-neighbor C' of C do
       if U1(C') > 1 + w1 · μ(C') + U1(C) then
         begin
           U1(C') ← 1 + w1 · μ(C') + U1(C);
           insert C' in L;
         end
   end
end
```

In this algorithm, L is a list sorted by increasing values of U_1 and it is initially empty; FIRST(L) returns the first element of L. Besides, $w_1 > 0$ acts like a weighting factor.

Once the navigation function U_1 has been computed, a graph search algorithm implementing a steepest descent method can be used to produce a path from any cell to G. In particular, one may adopt the BEST-FIRST procedure described in [3]. Of course, U_1 must be recomputed whenever the goal cell G is changed.

Some remarks are in order with reference to the above algorithm.

- As already mentioned, the fundamental difference between the original version of the algorithm and NF1 is that in the former the wavefront expansion takes place in the free space only, while in the latter a potential value is assigned to *each* cell

of $\mathcal{M}$. This fact might have undesirable consequences, both in terms of efficiency and safety of the planned paths. Thus, it is advisable to consider an *α-cut* of $\mathcal{M}$, i.e., to ignore cells with a risk value μ larger or equal to a threshold α when expanding the waves. Equivalently, one could perform a pre-processing step, in which a partially deterministic representation of the environment is obtained by labeling all cells with $\mu \geq \alpha$ as *occupied*. Still, NF1 will produce a navigation function U_1 that takes into account gray areas of $\mathcal{M}$, while the application of the original method would not. As for the value of α, it should be selected on the basis of the particular representation of the uncertainty. For example, when gray-level maps are built using fuzzy logic, values of $0.7 \div 0.8$ are appropriate [10].

- If the wavefront expansion takes place over an α-cut of $\mathcal{M}$, it may happen that the start cell S and the goal cell G fall into two disjoint components of $\mathcal{M}_\alpha$, with $\mathcal{M}_\alpha = \{C \in \mathcal{M} : \mu(C) < \alpha\}$. This indicates that there exists no path leading from S to G with maximum risk lower than α. In such cases, one could choose to increase α in order to find a solution path.

- As for the original algorithm of Barraquand and Latombe, the complexity of the algorithm NF1 is linear in $p \cdot q$, i.e., the number of cells of the bitmap, and does not depend on other parameters.

To illustrate the performance of the planning algorithm, we have built the simple gray-level map $\mathcal{M}_1$ shown in Fig. 1. Here, an α-cut was performed with $\alpha = 0.9$; cells with $\mu \geq \alpha$ are shown in black. $\mathcal{M}_{1,\alpha}$ consists of white cells ($\mu = 0$) and gray cells ($0 < \mu < 0.9$). The isopotential contours of the navigation function U_1 obtained for $w_1 = 10$ are shown in Fig. 2. Note how the contours are deformed in the vicinity of gray areas. The path from S to G produced by the BEST-FIRST procedure on U_1 is also shown. Since the cells of the traversed area have constant risk μ, the contours are not deformed and the solution path has minimum L^1-length, i.e., minimizes the Manhattan distance computed along the path.

3 Computing skeletons of gray-level maps

The planning method for gray-level maps of the previous section is simple and efficient, but suffers from a drawback similar to that of the original methods for deterministic maps, that is, it induces paths grazing zones with high values of μ (see Fig. 2 for an example). To solve this problem, Barraquand and Latombe proposed an improved navigation function computed in three steps [12], that produces paths lying as far as possible from the obstacles. The fundamental element of their method is a one-dimensional *skeleton* of the free space, that is built by expanding the wavefronts from the boundary of the obstacle region. This skeleton can be considered as a sort of numerical *Voronoi diagram* [14]. Below, we shall modify the original algorithm, in order to obtain a skeleton suitable for gray-level maps.

In the following, we assume that an α-cut of $\mathcal{M}$ has been performed in advance, and define the 'obstacle' region $\overline{\mathcal{M}}_\alpha = \{C \in \mathcal{M} : \mu(C) \geq \alpha\}$. The algorithm below builds a *modified* skeleton Σ for $\mathcal{M}_\alpha$.

Algorithm MOD-SKELETON

```
begin
  for every cell C ∈ M_α do
    d(C) ← ∞;
  for every C in M̄_α do
    if there exists a 1-neighbor of C in M_α then
      begin
        d(C) ← 0;
        P(C) ← C;
        insert C in L;
      end;
  while L is not empty do
    begin
      C ← FIRST(L);
      for every 1-neighbor C' of C in M̄_α do
        if d(C') > 1 + d(C) − w2 · μ(C') then
          begin
            d(C') ← 1 + d(C) − w2 · μ(C');
            P(C') ← P(C);
            insert C' in L;
          end
        else if η(P(C'), P(C)) > β then
            if C ∉ Σ then insert C' in Σ
    end
end
```

To understand how this algorithm works, note the following points:

- L and Σ are initially empty lists. L is sorted by increasing values of d.

- $P(C)$ is the *parent* of cell C, defined as the cell on the boundary of $\overline{\mathcal{M}}_\alpha$ from which the wavefront has reached C.

- $w_2 \in (0, 1]$ acts like a weighting factor.

- The quantity $d(C)$, as determined by the algorithm MOD-SKELETON, can be considered as a sort of clearance for the cell C. In particular, it is computed as the L^1-distance between C and the boundary of $\overline{\mathcal{M}}_\alpha$, diminished by a term that amounts to the (weighted) sum of the risks of the cells traversed by the corresponding wavefront. Equivalently, we may say that the wavefront is *accelerated* at cells with high values of μ. Hence, the skeleton Σ is farther from dangerous areas than the skeleton that would be obtained by considering $\mathcal{M}_\alpha$ as the free space and applying the original algorithm of [12].

- A cell C where two waves meet is inserted in $\mathcal{S}$ only if *(i)* the L^1-distance $\eta(P(C), P(C'))$ between $P(C)$ and $P(C')$ is larger than a threshold β, typically set to an integer value between 2 and 6, and *(ii)* the cell C does not belong to Σ already (this guarantees that the generated skeleton is one-dimensional).

To illustrate the outcome of the algorithm MOD-SKELETON, we have computed Σ for the same α-cut of the gray-level map $\mathcal{M}_1$ of the previous section, with $w_2 = 1$. The resulting Σ is shown in Fig. 3 together with the skeleton produced by the original algorithm of [12]. Note that Σ is 'deformed' by the presence of gray areas. As a result, the skeleton built by MOD-SKELETON is safer. For example, in the corridor-like zone on the right of the map, the skeleton Σ is translated downwards, due to the presence of a large gray area close to the upper wall.

Once Σ has been obtained, the planning problem from any start cell S to any goal cell G can be easily solved as indicated in [12]. In particular, G is first connected to Σ by a path that follows the steepest ascent of d on $\mathcal{M}_\alpha$, and the cells of this path are included in Σ. Then, a potential function U_2 is computed on the cells of Σ by a standard wavefront expansion restricted to Σ and starting from G. The previously computed field d is used to guide the expansion, as higher values of d identify areas with larger clearance. At the end of this step, all the cells of Σ that are connected to S have been given a potential value U_2; denote this subset of the skeleton as Σ_c. Finally, U_2 is computed in the rest of $\mathcal{M}_\alpha$ by a standard wavefront expansion starting from Σ_c.

It can be readily verified that also U_2 is a navigation function. Hence, one can use the BEST-FIRST algorithm in order to produce a solution path, that will be composed by three parts: a subpath connecting S to the skeleton Σ, a subpath on Σ, and a subpath connecting Σ to G. While the path will stay as far as possible from dangerous areas, it will no longer be optimal in the L^1 sense.

By properly choosing the data structures, the complexity of MOD-SKELETON (the core of the algorithm for computing U_2) reduces to $O(p \cdot q + s \log s)$, where $p \cdot q$ is the number of cells of the bitmap and s is the number of cells of Σ. As a result, the complexity of the algorithm for computing U_2 can be shown to be linear in $p \cdot q$.

4 Results for an experimental map

The described algorithms for path planning have been applied to the case of a gray-level map built from ultrasonic readings in our laboratory. The experiment area is a rectangle of 18×12 m, discretized into a grid of 180×120 square cells of side $\delta = 0.1$ m. The measures have been collected by a Nomad 200 mobile robot, that is equipped with a ring of 16 ultrasonic rangefinders. The open space is delimited by smooth surfaces (walls and closed glass cabinets) with poor diffraction properties, which respresent an adverse condition for ultrasonic sensing.

A detail of the gray-level bitmap $\mathcal{M}_2$ obtained with the map building method proposed in [10] is shown in Figs. 5–6. Here, an α-cut has been performed with $\alpha = 0.8$.

Figure 5 shows the path P_1 generated by the BEST-FIRST algorithm on the navigation function U_1 over $\mathcal{M}_{2,\alpha}$. Instead, the path P_2 generated by BEST-FIRST on the navigation function U_2 is displayed in Fig. 6. For the latter method, the skeleton of $\mathcal{M}_{2,\alpha}$ has been computed by the algorithm MOD-SKELETON.

As expected, P_2 stays in the middle of the open space. On the other hand, its L^1-length is increased with respect to P_1, since the skeleton of $\mathcal{M}_{2,\alpha}$ is followed. Note that Σ is rather erratic in the corridor, due to small variations of μ in $\mathcal{M}_2$. A smoother skeleton can be easily obtained by performing a preliminary quantization procedure on the map.

Finally, the total time needed on a 486 PC at 66 Mhz for computing the navigation function and generating the solution path was of 0.87 s for U_1 and 1.50 s for U_2, respectively.

5 Conclusions

We have presented two algorithms for building robot numerical navigation functions on gray-level bitmaps, where darker cells are more likely to be occupied by obstacles. The interest in this kind of representation comes from the large amount of uncertainty typically introduced by the sensing process.

Our approach to the planning problem extends the original work of Barraquand and Latombe for deterministic maps. In particular, two methods have been introduced, both based on a proper modification of the wavefront expansion algorithm. The first is very simple, but produces paths that graze the dangerous areas of the map. The second method, which is slightly more complicated, makes use of a properly defined skeleton of the map, that lies as far as possible from such areas. The performance of the two methods has been illustrated by application to both simulated and experimental maps.

In this paper, we have considered for simplicity the case of two-dimensional maps. The extension to higher-dimensional spaces is straightforward, but the complexity of the bitmap representation—and thus, of the algorithms for building the navigation functions—makes the proposed approach practical only for two- or three-dimensional spaces. Indeed, the use of these navigation functions in *configuration* spaces with dimension larger than 3 is not convenient. Nevertheless, one can use these algorithms for planning the motion of robots with many degrees of freedom by defining *control points* on the robot body, and building the navigation functions directly in the workspace, as discussed in [3].

We are currently working to extend the applicability of this approach to the case of *sensor-based* motion planning. In this situation, the gray-level map is not known a priori, and is incrementally built from sensor readings as the robot moves toward the goal. As indicated in [15], a proper sequencing of *perception* and *planning* phases is necessary to obtain an effective navigation method.

Acknowledgments

This work was partially supported by the Italian Government under MURST 60% funds and by the European Community within the ESPRIT BR Project 6546 (PROMotion).

References

[1] H. F. Durrant-Whyte, *Integration, Coordination and Control of Multi-Sensor Robot Systems*. Kluwer Academic Publishers, Norwell, MA, 1988.

[2] I. J. Cox and G. T. Wilfong, Eds., *Autonomous Robot Vehicles*. Springer-Verlag, New York, NY, 1990.

[3] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[4] J. F. Canny, *The Complexity of Robot Motion Planning*. The MIT Press, Cambridge, MA, 1988.

[5] Z. Li and J. F. Canny, eds., *Nonholonomic Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1993.

[6] A. Elfes and H. P. Moravec, "High resolution maps from wide angle sonar," in *Proc. 1985 IEEE Int. Conf. on Robotics and Automation*, St. Louis, MO, pp. 116–121, 1985.

[7] A. Elfes, "Occupancy grids: A stochastic spatial representation for active robot perception," in *Autonomous Mobile Robots: Perception, Mapping, and Navigation* (S.S. Iyengar and A. Elfes, eds.), IEEE Computer Society Press, Los Alamitos, CA, pp. 60–71, 1991.

[8] D. W. Cho, "Certainty grid representation for robot navigation by a Bayesian method," *Robotica*, vol. 8, pp. 159–165, 1990.

[9] M. Poloni, G. Ulivi, and M. Vendittelli, "Fuzzy logic and autonomous vehicles: Experiments in ultrasonic vision," *Fuzzy Sets and Systems*, n. 69, pp. 15–27, 1995.

[10] G. Oriolo, G. Ulivi, and M. Vendittelli, "Motion planning with uncertainty: Navigation on fuzzy maps," in *Proc. 4th IFAC Symp. on Robot Control (SY.RO.CO.'94)*, Capri, I, pp. 71–78, 1994.

[11] J. J. Leonard and H. F. Durrant-Whyte, *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Publishers, Norwell, MA, 1992.

[12] J. Barraquand and J. C. Latombe, "Robot motion planning: a distributed representation approach," Rep. No. STAN-CS-89-1257, Stanford University, 1989.

[13] G. Oriolo, G. Ulivi, and M. Vendittelli, "Potential-based motion planning on fuzzy maps," in *Proc. 2nd European Congr. on Intelligent Techniques and Soft Computing (EUFIT'94)*, Aachen, D, pp. 731–735, 1994.

[14] C. O'Dúnlaing and C. K. Yap, "A retraction method for planning the motion of a a disc," *Journal of Algorithms*, vol. 6, pp. 187–192, 1982.

[15] G. Oriolo, G. Ulivi, and M. Vendittelli, "On-line map building and navigation for autonomous mobile robots," in *Proc. 1995 IEEE Int. Conf. on Robotics and Automation*, Nagoya, J, 1995.

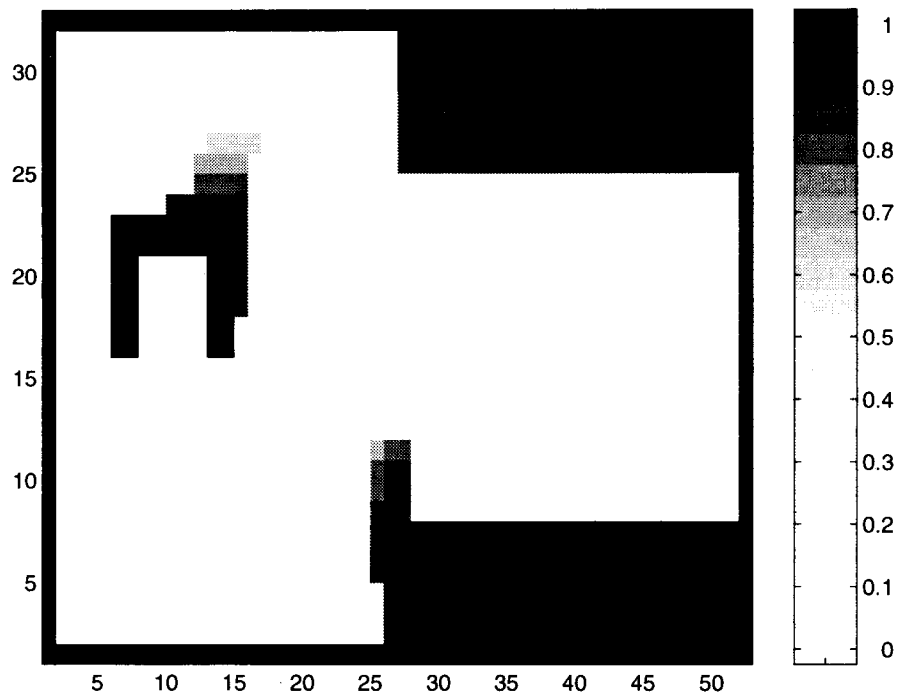

Figure 1. A simple gray-level map $\mathcal{M}_1$. The α-cut was performed with $\alpha = 0.9$. Cells with $\mu \geq \alpha$ are shown in black.

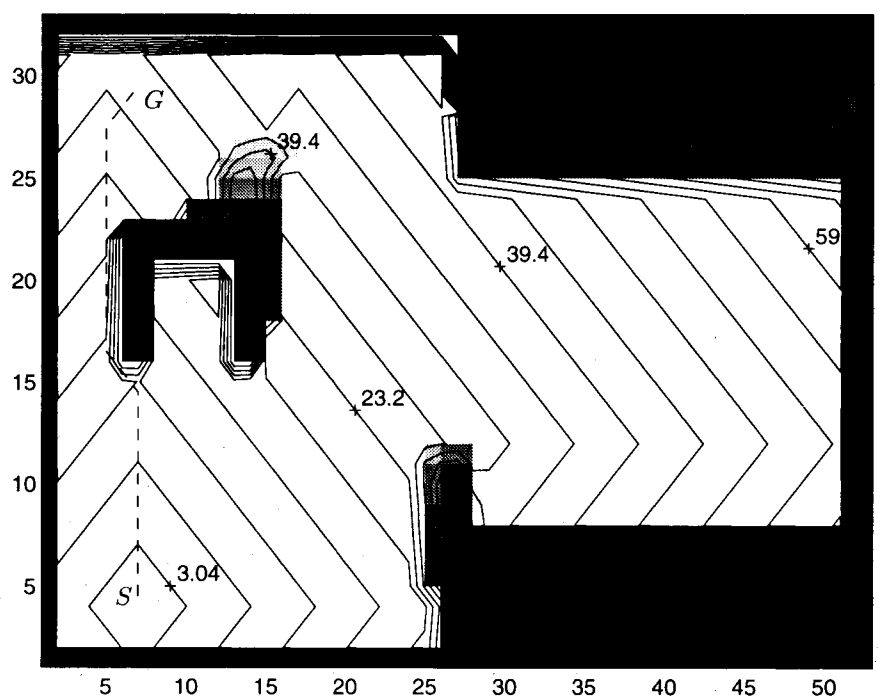

Figure 2. Isopotential contours of U_1 on the map $\mathcal{M}_1$, corresponding to the goal configuration G. The solution path shown is generated by a BEST-FIRST procedure.

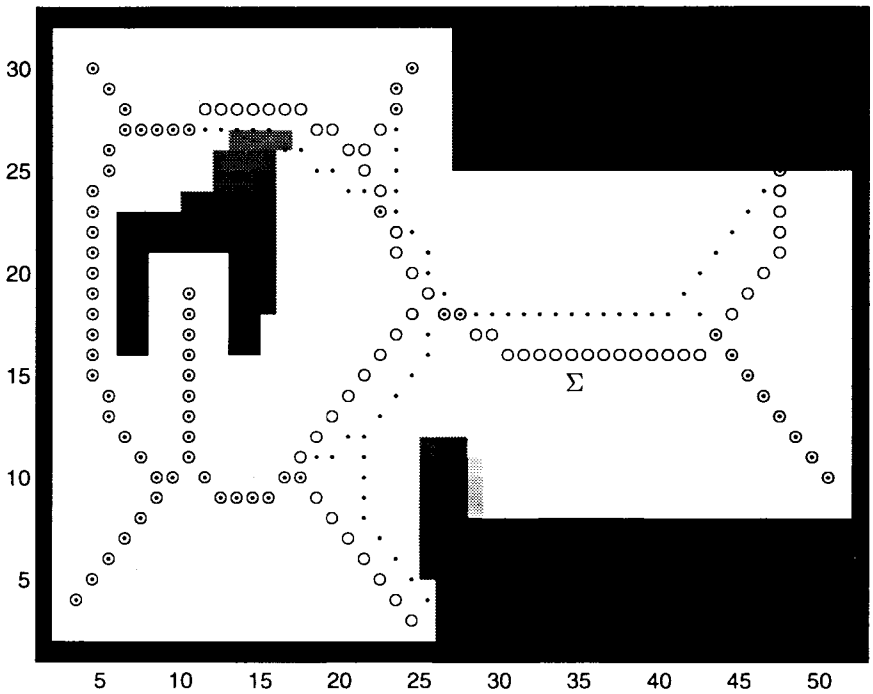

Figure 3. The skeleton Σ of $\mathcal{M}_{1,\alpha}$ produced by the algorithm MOD-SKELETON (circles) and the one obtained with the original algorithm of Barraquand and Latombe (dots).

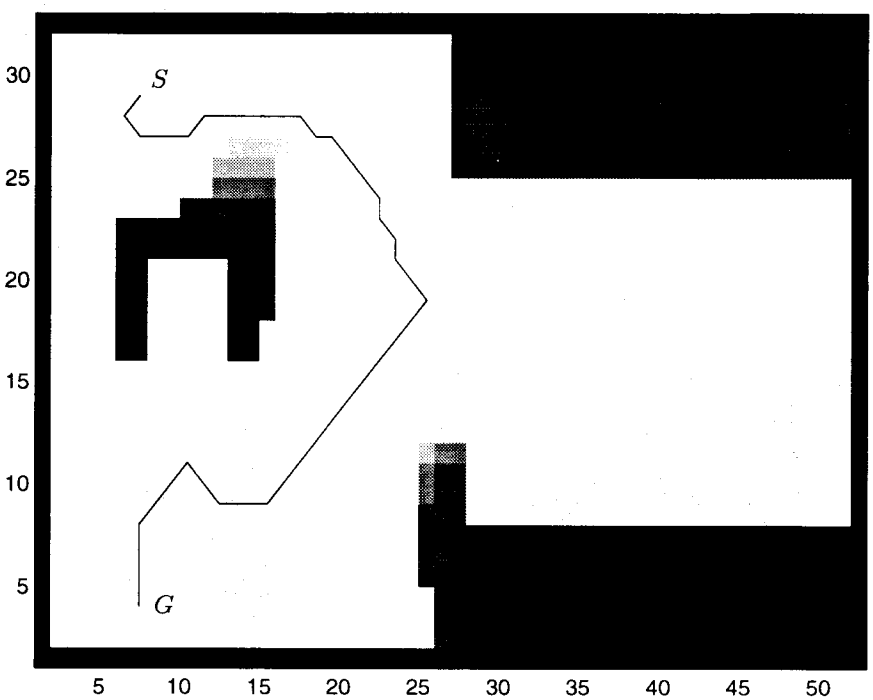

Figure 4. The path induced by the navigation function U_2 on $\mathcal{M}_{1,\alpha}$.

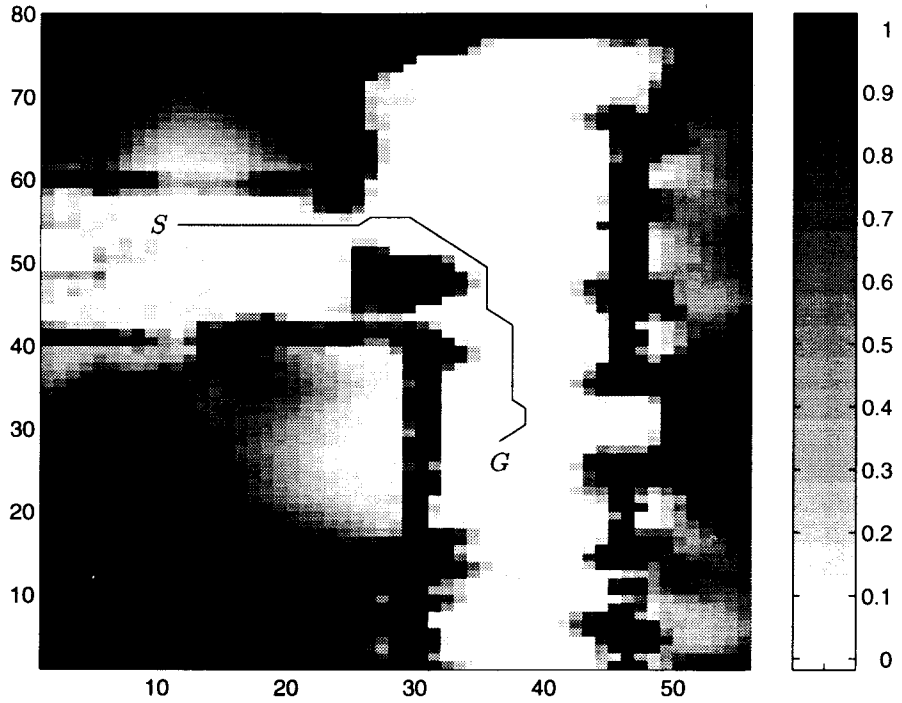

Figure 5. Experimental map $\mathcal{M}_2$: the path induced by the navigation function U_1 on $\mathcal{M}_{2,\alpha}$.

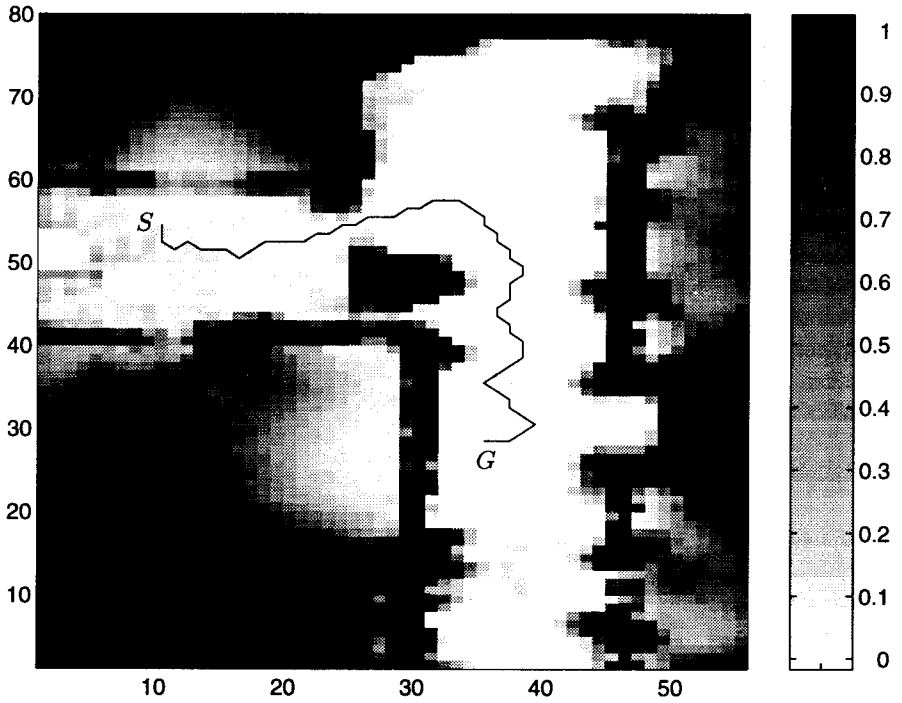

Figure 6. Experimental map $\mathcal{M}_2$: the path induced by the navigation function U_2 on $\mathcal{M}_{2,\alpha}$. The skeleton Σ of $\mathcal{M}_{2,\alpha}$ (not shown) was produced by the algorithm MOD-SKELETON.