Map Generation by Co-operative Autonomous Robots using Possibility Theory

Maite López-Sánchez, Ramon López de Màntaras, and Carles Sierra

1 Introduction

We treat the problem of generating maps of unknown office-like environments. Our approach is based on, first, a troop of low-cost autonomous robots that explore an indoor environment and, second, a host computer that receives the information gathered by the robots. The host uses this information to generate a global map of the environment.

Robots explore the environment looking for walls and other environmental features. Robots' movement is described by rectilinear displacements and turns are performed without changing robots' positions. In free space, robots explore by travelling random distances and making random turns. A robot stops its free movement when it detects a wall or an obstacle; in this case the robot follows the wall, or the edge of the obstacle, for a certain distance (which is randomly chosen). Robots' capabilities are very limited. During exploration, a robot gathers information about the environment by storing its trajectory. A trajectory is a sequence of consecutive rectilinear segments in global coordinates. If a segment corresponds to a wall following trajectory, it is labelled with the detected wall direction. We call this simple information a *partial map*. For each new partial map received, the host processes it and updates the *global map* (a grid representation), so that it is incrementally generated.

In our approach, there is no guarantee that all robots will successfully return to the host after their exploratory runs. To minimise the loss of information, the robots cooperate by transferring to each other the information they have gathered - this transfer of information occurs when when robots meet. Therefore, sharing information allows the host to build the global map using not only the information from the returning robots, but also the information from those whom they previously met, but who did not return. Robots and their partial maps are numbered. Therefore, after two robots have met, each robot will contain its own partial map and the partial map that the other robot has acquired until that moment (that is, the trajectory that the other robot has executed until the meeting). The host processes the partial map of the robot that returns first, and stores the partial map of the other robot. If, after a reasonable amount of time, the other robot does not return, it is considered to be lost, and the host processes the stored partial map.

Robots store their trajectories during their exploration runs. Nevertheless, their odometry error generates imprecise information about the positions of detected environmental features as well as about the locations of free space. We make use of Possibility Theory [7] to model the uncertainty of the this information. In fact, we use Possibilistic Logic, which is a particular case of Fuzzy Logic [8]. Given some vague information, Possibilistic Logic evaluates the truth of crisp propositions whereas Fuzzy Logic evaluates the truth of vague propositions. In our case, propositions are of the kind "the position of the robot is (x, y)" and "the robot is following a wall at (x, y)" where x and y are crisp numbers. We evaluate the truth of these propositions on the basis of odometry errors from which degrees of possibility Π and necessity $\mathcal{N}$ are derived. Therefore, the resulting global map models the environment in terms of degrees of possibility and necessity of the position of detected walls and obstacles.

When choosing a representation of environmental features, there are several facts we must take into account:

- On the one hand, the mapping process must consider how to represent the shape and distribution of obstacles. It is very common to represent indoor environments with discretized grid-based maps [13,14,9] and outdoor environments with more abstract representations such as graphs [11,20,12]. Each cell of the map grid represents an area of the environment and has information about its occupancy. On the contrary, graphs represent obstacle areas as nodes and relations between them as edges. One of the reasons for making this distinction is that very often the size of the grid cells is constant. This implies a homogeneous resolution of the space that is not appropriate for extensive outdoor environments with low obstacle density. Nevertheless, grid maps have the advantage of independence from the shape of the objects in the environment.
- On the other hand, it is important to deal with the uncertainty associated with each map element when choosing the right map representation. Although there are several alternative solutions to deal with uncertainty, probability measures are the most commonly used for both grid-based and abstract maps. An example of an abstract map is described in [2]: it uses landmarks, defined as object features, to model natural environments. The uncertainty associated with landmarks position is estimated by means of probabilistic techniques assuming a Gaussian distribution of the uncertainty. In the case of certainty grid representations, the probabilistic approach is used to estimate the probability of cell occupancy [17,18].

Uncertainty can also be treated using Possibility Theory or Evidential Theory, and we use the former one. Nevertheless, it is worth noting that Possibility and Necessity are in fact particular cases of Belief and Plausibility. In our case, our frame of discernment is $\Omega = \{wall, \neg wall\}$ where *wall* is the evidence of a detection and $\neg wall$ is the evidence of free space (further details are given in Sect. 3).

We focus on the uncertainty derived from odometry errors. This is due to the way in which information is gathered. On the one hand, free-space information is derived from robots' trajectories. And, on the other hand, we only consider occupancy information that comes from followed portions of walls (or obstacle edges). Our robots use infrared (IR) sensors to perform wall following. These sensors have a very limited range and do not supply the exact distance to a wall. Adverse lighting conditions may prevent robots from following walls because wall following requires specific sequences and combinations of IR readings. Thus, bright light may cause the robot to miss a wall, but never make it follow a non-existent wall. It is also common that robots do not detect dark objects. In such case, the robot will collide and try to abandon the problematic area. In both cases (i.e., bright light or dark object), there will not be any occupancy or free-space information, so that the corresponding area in the global map will remain unknown (but not misclassified). Although there is complete certainty about the presence of a wall or obstacle, its exact location is not known: partly because of odometry error, and partly due to the ignorance of the distance to the wall (which, in addition, is not constant, because robots do not follow the wall in a completely parallel manner). Nevertheless, odometry error is our main concern because the distance can be approximated by a mean value with an associated error, whereas the odometry error grows constantly.

Our approach consists in generating a global map represented on a bidimensional grid. Each cell contains two values: the degree of possibility and the degree of necessity of the presence of obstacles (as we will see, a cell can also have orientation and singular point labels). Initially, all the cells have a possibility value Π of 1 and a necessity value N of 0. These initial values correspond to a situation of total ignorance and are intuitively interpreted as "although it is completely possible that there is a wall, there is no certainty at all about it". Afterwards, detection information yields Necessity values greater than zero, corresponding to obstacles, whereas information about free space gives Possibility values smaller than one. As robots communicate their partial maps, the host processes them in order to infer the possibility and necessity values. They are added to the global map by combining these new values with the previous ones in the grid. Due to the robots' odometry error, the positions of the walls detected and followed have an associated error. This means that a portion of the wall may actually be in the surroundings of the cells where the robot thinks it is. This neighbourhood is approximated by an error rectangle centred on each detected position. This error rectangle is an approximation of the estimated odometry error and provides support for the distribution of possibility and necessity values.

Our results come from a simulation of the physical robots. Although the real robots have been used to measure the actual odometry error, their development is still in progress. Thus, we had to simulate the robot exploration to obtain input data for the host map generation process. The simulator has been developed, including the odometry errors of the real robots. Our experiments consider several autonomous robots exploring unknown indoor environments. Partial maps contain robots' trajectory segments that are used as evidence for the position of walls and free-space areas. This evidence is translated into possibility and necessity degrees. This assignment does not represent segments explicitly. Nevertheless, there is implicit information in the grid that allows us to treat adjacent cells as belonging to the same portion of a wall. We also describe how this implicit information can be used to refine the global map. On the one hand, wall information can be grouped in segments that can be extended under certain circumstances. And, on the other hand, wall information can be grouped into polygon-shaped obstacles that allow to plan paths toward less explored areas.

2 Problem statement: the mapping problem

The general problem

In robotics, solving the mapping problem means generating a model of a given environment. The solution to this problem can be addressed from different points of view depending on the characteristics of the a priori knowledge about the environment. In this way, if the environment has been augmented by adding landmarks, mapping means recognizing landmarks and establishing relations between them. On the contrary, if the system initially has no landmarks but an approximate map of the environment, the mapping process consists in matching previous knowledge with actual detections of landmarks in order to refine the map. Finally, when the environment is unknown and not augmented with landmarks, there is no direct way of distinguishing the relevant information and, therefore, every piece of information is included in the map.

Robot characteristics are also crucial in the mapping process. This is because imprecision associated with the obtained data differs substantially depending on the robot equipment. That is, robots can know with more or less accuracy their position depending on how they obtain their odometry information (encoders at the wheels for dead reckoning, compass for heading or GPS for global positioning). At a given position, robots can gather very different kinds of data depending on the sensors they are equipped with. Cameras, sonar sensors, infrared sensors and laser range finders are different possibilities. Cameras give global views, but present difficulties in distinguishing obstacles in an image. Sonar sensors gather more information than infrared sensors; they cover a bigger range, but their information is less reliable due to spurious reflections. Finally, laser range finders are very accurate, although, in most cases, their high cost prevents their use.

Our mapping problem

Among the possible settings of the mapping problem described above, we consider an office-like indoor unknown environment that is mapped using information coming form a troop of low-cost autonomous robots that explore the environment. The robots are equipped with wheel encoders to estimate their position and infrared sensors to follow detected obstacles. The uncertainty associated with the information that robots collect is treated with Possibilistic Logic. The reason for choosing possibility/necessity techniques instead of probability is our need for an initial assignment of values representing ignorance. Possibility theory allows a clear representation of ignorance, but probability does not. The latter often uses an initial value of 0.5 in case of ignorance, but this value actually represents occupancy equiprobability instead of ignorance. Furthermore, probabilistic techniques are reliable only if enough sensor data is available, the data is well distributed in the explored environment and this distribution can be easily obtained.

Environment

Our mapping process only considers environment features with rectilinear edges long enough to be followed by the robots. Small obstacles, such as legs of chairs or desks, are not represented because they generate spurious IR sensor readings that are considered as noise. This does not represent an important disadvantage because small obstacles can be easily avoided using a "non-informed reactivity". That is, information about a small obstacle is not crucial due to the fact that whether the robot avoids it by turning to the left or right will not imply a significant change in the global performance of the robot. Besides, if the contour of a non-rectilinear feature has been followed, it will be approximated by a line between the first and last positions where the wall following took place (these points are the only ones that the partial map stores).

Therefore, considering only relatively large environment features, we can assume that office-like environments are highly orthogonal (i.e., horizontal and vertical). This is due to the fact that, on the one hand, walls are usually connected by right angles; and on the other hand, human-made office objects such as bookshelves or drawers tend to have rectangular shapes.

In case a robot follows a wall or an edge obstacle with a trajectory not parallel to the object, the resulting detected segments will not be orthogonal. The orthogonal assumption presents the advantage that it can be used to correct these non-orthogonal segments. Nevertheless, half opened doors appear so often in office-like environments that the mapping process has been refined to consider exceptional non-orthogonal features. Thus, the host only orthogonalizes those segments that are almost vertical or horizontal and keeps the oblique segments as they are.

Characteristics of our mobile robots

Robots have been designed bearing in mind that the hardware had to be as simple as possible but, on the other hand, the robot had to show smart behaviour in order to navigate efficiently. These requirements resulted in a design that contains three different functional modules. First, the steering module that controls the motors in order to follow a trajectory. Second, the perception module, which acquires information about the environment by means of IR sensors. And third, the navigation module that generates the trajectory to be followed (see [14] for details).

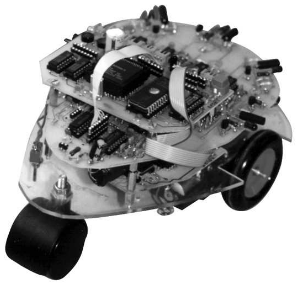


Fig. 1. Autonomous Mini-Robot.

Each robot is 21 cm long and 15 cm wide (see Fig. 1). It has three wheels, two of them are 5 cm steering wheels controlled by independent motors. The robots can reach a maximum speed up of 0.6 m/sec and they are equipped with the following sensors:

- Impulse generators at each wheel for odometry. They generate 5 encoder pulses per centimeter so that each pulse corresponds to a displacement of 2 mm.
- Five IR proximity sensors for obstacle detection placed at 0^o, $\pm45^o$ and $\pm90^o$. These provide two possible readings: 'near' and 'far', which correspond to 10 and 20 cm respectively.
- Safety micro switches for the detection of collision.
- One omnidirectional IR Emitter/Receiver sensor that detects the presence of other robots and transmits data.

Error analysis

In order to study the robots' odometry error, we have performed an analysis based on experimental data. This data has been obtained from the measurements of how real robots deviate when they move following straight line trajectories. We have considered the data from a robot covering 20 times a distance of 3 meters and 20 times twice that distance (i.e., 6 meters). With the obtained data, we have used the Kolmogorov normality test to verify that the experimental sample indeed follows a normal distribution both in the direction of the trajectory and in the direction perpendicular to the trajectory. The obtained distributions are $N(0, 1.7)$ and $N(0, 7.3)$ respectively. Furthermore, we have tested that both distributions are independent.

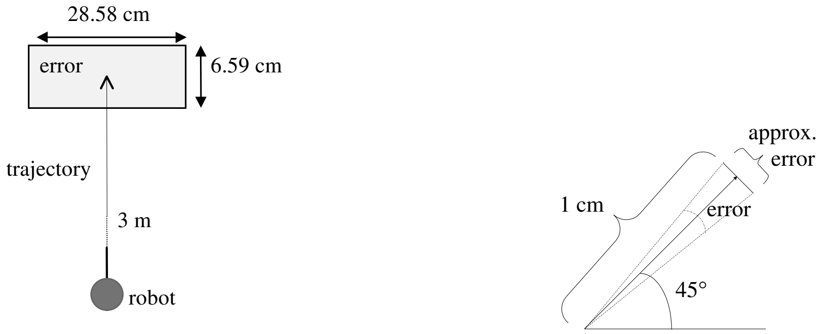


Fig. 2. Left: displacement error rectangle. Right: turning error approximation.

Based on the above normal distributions, we have determined the size of an error rectangle that reflects 95% of the sample coming from the 3-meter runs (in fact both samples present elliptical shapes). Fig. 2 shows the resulting rectangle associated to the robot final position after the 3-meter straight run. This rectangle size is 6.59 cm in the direction of the trajectory by 28.58 cm in the direction perpendicular to the trajectory. This respectively corresponds to errors of 0.022 cm and 0.095 cm per covered cm.

Applying the same study on the 6-meter run sample, we have obtained a rectangle that is twice the size of the previous one (by considering the confidence interval margin). And therefore, this allows us to conclude that the size of the error rectangle is linearly proportional to the covered distance.

Finally, two additional experiments test turning errors. First, the robot turns 45 degrees (+45° in one experiment and −45° in the other) and then, it makes again the 3 m displacement. Surprisingly, the sample deviation was not really affected by the turn, and the mean was very similar for both movements. This divergence corresponds to a deviation of approximately two degrees on the left (which is likely to originate in a built-in robot problem). In this manner, we conclude that the studied robot has a deviation but not a turning error that can be isolated from the displacement error. However, since this result cannot be generalized, when computing the odometry error we parametrise robot-turning errors in terms of a segment approximation in the direction perpendicular to the displacement. The turning error we use corresponds to an approximation of the error of turning ±45° and moving 1 cm.

Robot's architecture

Each robot implements two navigation strategies: random exploration and path following. Both strategies are based on the co-ordination among different elementary behaviours (following the Brooks' philosophy [4]. Basically, the representation of each strategy is a deterministic finite state automaton in which each state corresponds to an elementary behaviour. Fig. 3 shows the automata that perform the random exploration strategy and the path following strategy.

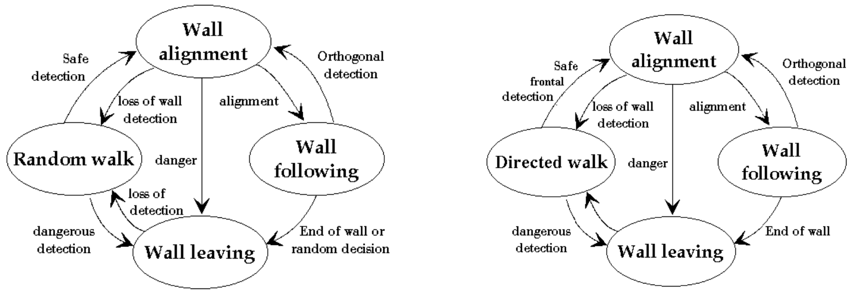


Fig. 3. Automata schemas. Left: random exploration strategy. Right: path following strategy

We will see in the next subsection that these basic behaviours use sensor readings as well as historical information about previously taken decisions to determine what are the actions to execute and when to switch to other behaviours under the conditions shown by labels on the arcs in Fig. 3. This "one behaviour active at a time" policy avoids the problem of combination of behavior outcomes that appears in those approaches activating more than one simple behaviour simultaneously [22]. Our solution is to have less simple behaviours in order to make the decisions that satisfy the goal for each strategy. Decisions are made using If-Then rules.

Elementary behaviours

The random exploration strategy co-ordinates elementary behaviours to cover free space by changing the robot's direction randomly and by following obstacles when detected. This randomness is modelled by two parameters that are individually assigned to each robot: a turning probability, that is, how often a robot changes its direction) and a left/right turning probability, that is, to which side the robot tends to turn more frequently. Different probability values produce different global behaviours. For instance, a robot with a high turning probability presents a global “nervous” behaviour, while one with a low value shows a “calm” behaviour. “Nervous” robots tend to cover small environment areas exhaustively. On the contrary, since “calm” robots turn less frequently, they have a tendency to explore larger areas.

Once the environment has been partially explored, the host computer can generate paths to unexplored areas. Such paths are represented as a sequence of positions. Robots can then apply the path following strategy, which consists of navigating through the environment reaching these path positions sequentially.

We briefly describe here each elementary behaviour (the behaviour switching conditions are shown in Fig. 3 and a more detailed description can be found in [16]):

- *Random walk*: this behaviour is active while the robot is in free-space. That is, it controls the movements of the robot when there are no sensor readings. Random turns and lengths of displacements are decided using the turning and left/right turning probabilities explained above.
- *Directed walk*: this behaviour is the default one in the path following strategy. Length and orientation of robot displacements are computed considering the next position to reach in the path and the current position. This behaviour is active while nothing blocks the robot’s way.
- *Wall alignment*: this behaviour tries to “align” the robot to what might be a wall. By aligning we mean having the robot parallel to the wall, or obstacle edge, by first facing the wall and next turning 90 degrees. A successful alignment is recognised by the robot when it has sensor information coming from sensors on one side without any front detection. Of course, this is not done instantaneously, but requires a series of small angular corrections and displacements. During random exploration, any sensor reading triggers the wall alignment behaviour. Under the path following strategy, everything works similarly, the only differences are: 1) the wall alignment behaviour tries to align the robot only to objects obstructing its way, 2) it turns in the direction of the smaller angle deviation with respect to the current path, and 3) a loss of sensor readings implies a switch to the directed walk behaviour.
- *Wall following*: once the robot is aligned to a wall, this behaviour computes the distance the robot will try to cover and controls its displacement by keeping it parallel to the wall .

  In the case of random exploration, the wall following distance is computed using the turning probability. Thus, a “nervous” robot will follow a wall, on average, at a shorter distance than a “calm” robot (nevertheless, a minimum distance is guaranteed). In the case of path following, the behaviour tries to reach the end of the wall. First, the robot follows the

wall (or obstacle side) for a fixed distance, and if the end is not reached, it turns 180 degrees and doubles the distance.
- *Wall leaving*: During random exploration, the goal of this behaviour is to reorient the robot until there are no sensor readings and then switch to the random walk behaviour. Under the path following strategy, this behaviour reorients the robot until there are neither collision detections nor frontal sensor readings, and then switches to the directed walk behaviour. The goal of this behaviour is to redirect the robot until there are no sensor readings and then switch to the random walk behaviour.

There are still two more elementary behaviours concerning the communication process between pairs of robots:

- *Presence detection*: any behaviour will switch automatically to this behaviour when a robot detects the presence of another robot by means of the omnidirectional sensor. Once another robot is detected, the robot changes the frequency of the omnidirectional sensor, stops, and waits for the same frequency in the received signal from the partner to switch into the data transmission behaviour.
- *Data transmission*: this behaviour consists in the transmission of the information gathered so far, that is, the robot's partial map.

Navigation performance

At the lowest level of abstraction, the navigation is nothing more than a sequence of actions (turn, move, stop) executed in the environment. These actions are commands to the effectors. The sequence of actions is generated by the currently active elementary behaviour, which sets the sub-goal that must be executed and controls its success, interruption or adaptation. For example, when a wall-following elementary behaviour is active it sets a sub-goal (a displacement of a certain length and orientation) and controls its execution using sensors: wheel encoders give the travelled distance (this is the clue for the sub-goal success), any front detection cancels the behaviour, and too far or too close side sensor readings indicate that the orientation must be adjusted. Figure 4 shows an example of wall following execution that turns slightly in order to keep parallel to the oblique wall. The rules that generate deviations consider a fixed small angle (that is set to 7.5 degrees). These rules are:

```
If Behaviour=right-wall-following and 45right_sensor=no detection
Then rotate(-angle)
If Behaviour=right-wall-following
   and(45right_sensor = near or front_sensor=near)
Then  rotate(+angle)
```

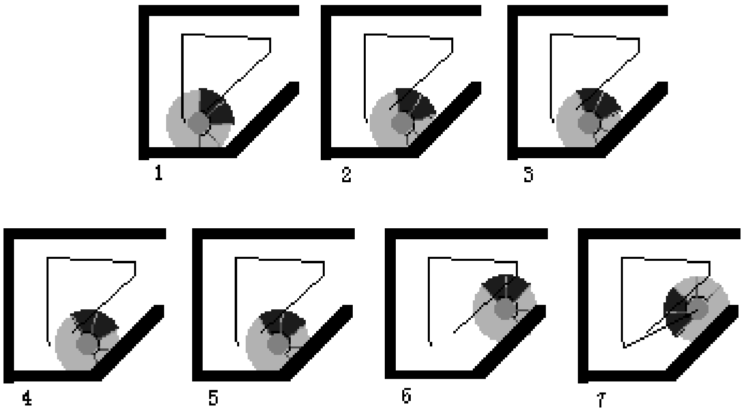


Fig. 4. Wall Following Example.

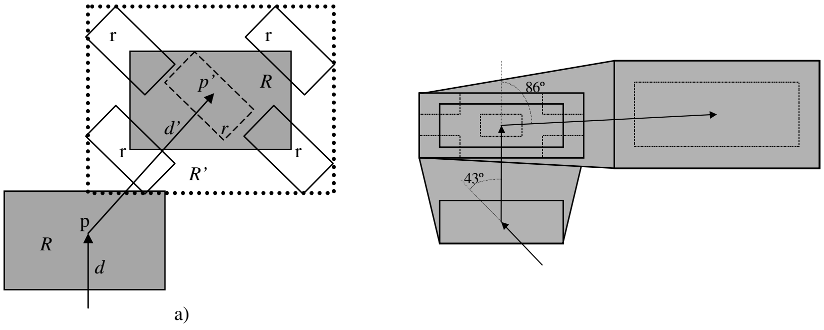
3 Map generation process

Error propagation

Each robot moves following a trajectory that is composed of straight lines and turning points. Therefore it is possible to describe a trajectory as a sequence of connected segments where the end point of a segment coincides with the initial point of the next. Considering the error rectangle that is associated with displacements and turns, we compute how this error grows along the trajectory. Fig. 5(a) illustrates how a rectangle error R gets expanded by a rectangle error r to obtain a new rectangle error R'. R and R' are the errors associated with the end points p and p' of two consecutive trajectory segments. The rectangle error r represents the resulting error from the displacement d' of the last segment. Its dimensions (length l and width w) are computed as follows:

$$\begin{aligned} l &= L + \cos(\alpha) \cdot (b + c) + \cos(90 - \alpha) \cdot a \\ w &= W + \cos(\alpha) \cdot a + \cos(90 - \alpha) \cdot (b + c) \\ a &= d' \cdot e_d,\ b = d' \cdot e_p,\ c = d' \cdot (e_t \cdot \alpha/45) \end{aligned}$$

Where L and W are the length and width of rectangle R, d' the distance travelled, α the last turned angle, and e_d, e_p, e_t the error values of the robot. As we have seen in the error analysis, these values are $e_d = 0.022$ cm per covered cm in the direction of displacement, $e_p = 0.095$ cm, which goes in the perpendicular direction, and e_t corresponding to the turning error.



(a) (b)

Fig. 5. Error propagation along the robot trajectory.

Obviously, all the intermediate points between the initial and final points of a segment also have an associated error proportional to the displacement done so far. Thus, consecutive rectangle errors define an area around the trajectory that contains all the positions where a robot could actually have been. Part (b) of Fig. 5 shows the error propagation after a 45 degrees right turn, a straight line, another right turn of 90 degrees and finally another straight line.

When following a wall, the robot tries to keep parallel to the wall at a certain distance d. Therefore, the segment that represents the detected wall is computed as a parallel segment on the side of the detection. Thus, the error associated with the wall is a translation of the trajectory error plus e_d: the error of the infrared sensors that detect the wall. IR sensors have a range of 20 cm and its error is ±5cm. Fig. 6 shows in dark grey the wall error that results from the trajectory error, which in turn appears in light grey. Since the robot practically always remains at the same distance from the wall, the error along the direction orthogonal to the wall is taken to be constant and equal to the mean of the initial and final segment errors.

Modelling the certainty

The space being explored by the robots is discretized by means of a grid. Cells in the grid represent a small area of the real environment. The size of this area depends on the granularity that the host uses to represent the environment. Our system considers square cells whose size is a parameter of the system (it is usually taken as 1 or 2 cm long). Grid cells contain two values: the degree of possibility and the degree of necessity of the presence of obstacles. Initially, all the cells have a possibility value Π of 1 and a necessity value N of 0. These initial values correspond to a situation of total ignorance and are intuitively interpreted as "although it is completely possible that there is a wall, there is no certainty at all about it".

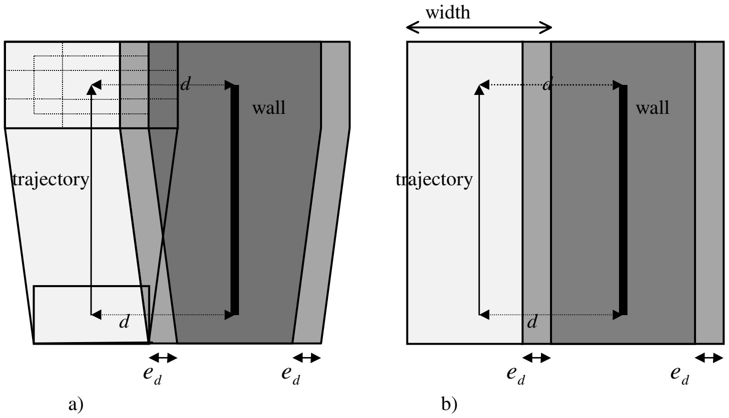


Fig. 6. Computation of a wall segment and its error based on the robot's trajectory.

As robots transmit the information gathered during their exploration, the possibility and necessity grid values are modified in a way that depends on the presence, or absence, of obstacles. The information gathered by each robot is called a partial map and contains its trajectory in global co-ordinates. Trajectories are sequences of segments that are specified as lists of turning positions. Each segment has associated information that specifies if the trajectory segment corresponds to a wall following segment, in which direction the wall has been followed, and if it ends (wall ends and corners are referred to as singular points).

Taking into account the errors associated with the segments, we use each robot's trajectory and detections to determine the areas in the grid for which the possibility and necessity values will be updated. Fig 7 shows two examples of cell areas: in a) related to a communicated position, or in b) a discretized segment.

Modelling the certainty of followed walls When an error rectangle is associated with a position that belongs to a followed wall, the occupancy certainty degree is updated in every cell that becomes partially or totally covered by the error rectangle associated to that position. Information about followed walls updates the necessity value in the corresponding cells and represents the degree of certainty about the presence of an obstacle in each position.

Necessity values decrease linearly with the magnitude of the error and remain positive ($N(wall) = \alpha > 0$) in the cells inside the error rectangle, but obtain the value 0 in the cells outside the limits of the rectangle. These values have been established with the aim of reflecting that, having detected some obstacle, the necessity that there is a wall cannot be zero any longer, but

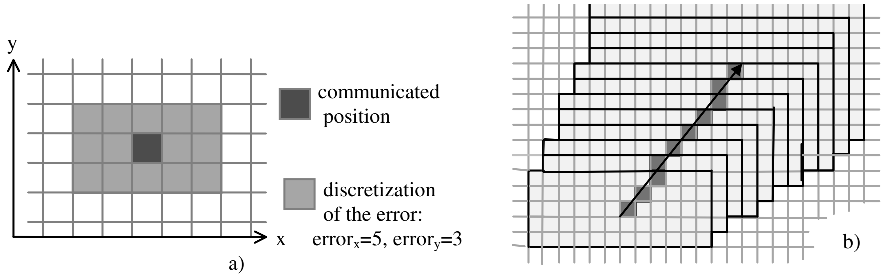


Fig. 7. a) Grid representation of a position and its associated error; b) Discretization of a segment and its error.

can only be positive since a positive value denotes some certainty about the occupancy of the space. However, this occupancy certainty degree decreases when the distance to the central cell of the error rectangle increases. Fig. 8(a) shows this case. Notice that the possibility value is constantly equal to 1 in all the cells covered by the error rectangle. This is due to the following axiom of Possibility Theory: $N(A) > 0 \Rightarrow \Pi(A) = 1$.

As we have already mentioned in the introduction, our Possibility and Necessity values are particular cases of Belief and Plausibility ones. We can easily see how our assigned values $N(wall) = \alpha > 0$ and $\Pi(wall) = 1$ can be considered as $Belief(wall)$ and $Plausibility(wall)$ corresponding to the following basic probability assignment (b.p.a.):

Let the frame of discernment $\Omega = \{wall, \neg wall\}$, with mass:

$m : P(\Omega) \to [0, 1]$, $m(\emptyset) = 0$, $m(wall) = \alpha$, $m(\{\neg wall\}) = 0$, $m(\Omega) = 1 - \alpha$.

Therefore, we obtain:

$$Bel(wall) = \sum_{A \subseteq \{wall\}} m(A) \ = \ m(\{wall\}) + m(\emptyset) = \alpha$$

$$Pl(wall) = \sum_{A \cap \{wall\} \neq 0} m(A) \ = \ m(\{wall\}) + m(\Omega) = 1$$

Modelling the certainty of free space Robot trajectory segments provide us with information about free space. In terms of possibility and necessity values, free space information corresponds to $\Pi(\neg wall) = 1$ and $N(\neg wall) > 0$. These values entail that $\Pi(wall) < 1$ and $N(wall) = 0$ according to the following axiom of Possibility Theory: $N(A) = 1 - \Pi(\neg A)$.

In the same way as necessity values of obstacles, $N(wall)$ decrease when approaching the edges of the error rectangle. Necessity values of the negation of obstacles $N(\neg wall)$ also decrease proportionally to the distance to the

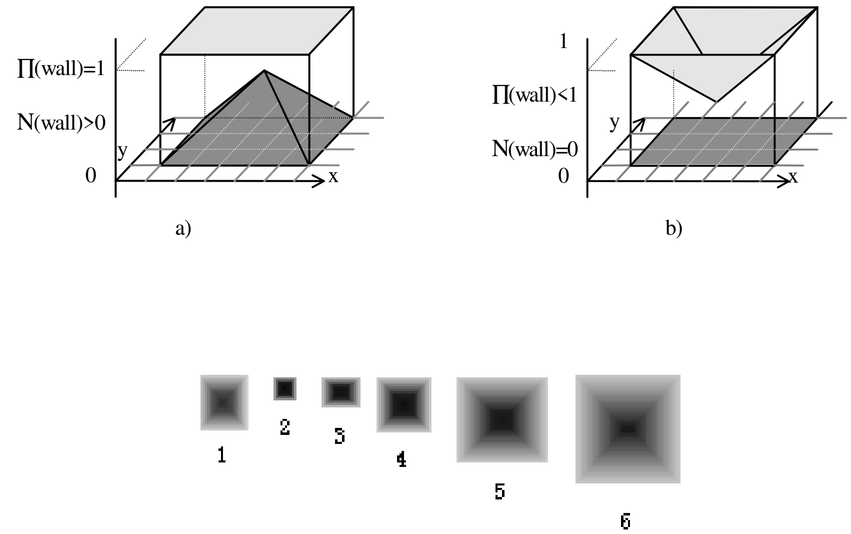


Fig. 8. *Π* and *N* value assignment from: a) Wall, and b) Free space information. From 1 to 6: examples of the representation in the host of a Necessity pyramid (1) and several Possibility pyramids (2..6).

central cell of the error rectangle. Therefore, the possibility value of obstacles $\Pi(wall)$ increases in the same proportion until it reaches the value 1 at the cells outside the limits of the error rectangle. Obviously, we have $N(wall) = 0$ for all the cells covered by the error rectangle. Fig. 8(b) shows this case.

Value assignment

The height of the pyramids in Fig. 8 is determined by the size of the error rectangle. The underlying idea is to establish a linear error-height relation such that, a null error implies the maximum allowed value of height (i.e., one). An error equal to the maximum allowed error, K, implies a zero height since the information is no longer reliable. This maximum allowed error threshold, measured in grid units, assigns a limit to the error. K is established experimentally and it is twice the one that forces the robot to return from its exploration. Big errors imply data irrelevancy and high risk of not being able of returning to the host, even when using the same trajectory without loops.

Let us represent the error rectangle by the tuple $\{x_c, y_c, e_x, e_y\}$, where x_c and y_c are the coordinates of the central cell of the rectangle and e_x and e_y are, respectively, half the length of the base and half the length of the height of the error rectangle measured in number of grid cells. Following this, the height of the pyramid is given by:

$$\text{height} = 1 - \frac{\max(e_x, e_y)}{K} \qquad (1)$$

The necessity value of having a wall at cell (i, j) is given by:

$$N_{ij}(wall) = \text{height} \cdot \min(1 - \frac{|i - x_c|}{e_x}, 1 - \frac{|j - y_c|}{e_y}) \qquad (2)$$

The possibility value $\Pi_{ij}(wall)$ is obtained by $1 - N_{ij}(\neg wall)$ using 1 and 2.

Combination of values

In the previous subsection we have seen how necessity and possibility values are propagated from a central cell towards its surrounding cells. Each central cell represents a point of the discretization of wall segments or free-space trajectory segments. As a consequence of this discretization, central cells are adjacent along the trajectory segment and therefore, value propagation is done over cells that may already have been updated by a previous propagation. In other words, we are propagating overlapping pyramids, and this implies that new values must be the result of a combination between old and new pyramids.

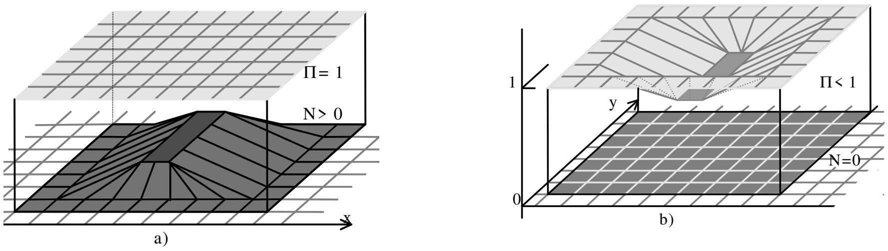


Fig. 9. a) Wall representation. b) Representation derived from trajectory segments.

In the case of wall segments, they are discretized, and orthogonalised if they are nearly horizontal or nearly vertical. The values derived from wall evidence are necessities increasing from 0 and which combined using the *max* operation (Fig. 11). In the case of trajectories, possibility values decrease from 1 and are combined by means of the *min* operator. Fig. 10 graphically shows the results of such combination.

When the same portion of a wall has been followed more than once (either by the same or different robots), we can consider the resulting information as being independent due to the random exploration they perform.

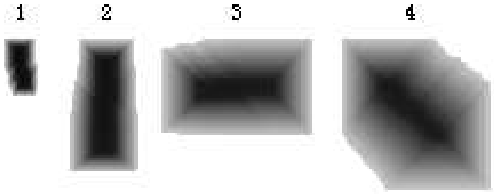


Fig. 10. Examples of possibility pyramids derived from trajectory segments combined at the host.

The necessities are combined by means of the probabilistic sum, that is $S(x, y) = x + y - xy$, in order to reinforce the certainty about the location of the wall. Fig. 11 shows a top view of this situation. Part 6 of Fig. 11 corresponds to wall information in the global map. The host represents each cell by a pixel (a point in the figure) and assigns colours whose darkness grows proportionally to the necessity value. In part 7 of Fig. 11 we can see an auxiliary grid that represents an overlapping portion of the same wall. The values in the auxiliary grid are combined with the values in the global map so that the global map is updated (see Fig. 11 part 10)).

To summarise, the process is as follows:

```
Add_new_partial_map_into_global_map(partial_map,max_error)
{ Repeat while(non-empty(partial_map))
```

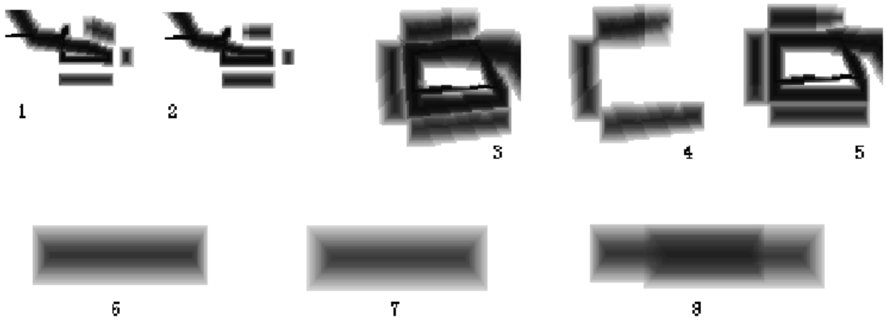


Fig. 11. Examples of necessity pyramids for a wall representation combined at the host. 1,3,4: Non-orthogonalised walls and trajectories; 2,5: Orthogonalisation of 1 and 3; Wall reinforcement: 8 results from the combination of 6 and 7; The upper and left walls in 3, 4, and 5 have been also combined. Singular points correspond to light extremes.

```
  {  segment=read_segment(partial_map)
     error=max_error(segment)
     If (error<max_error)
     {  d_seg=discretize(segment)
        If (segment.wall=='yes')
        {  d_seg=orthogonalize(d_seg)
           d_wall=compute_wall_segment(d_seg)
           add_wall_and_trajectory_to_global_map(d_wall,d_seg)
        }
        Else
          add_trajectory_to_global_map(d_seg)
     }
     Else
         partial_map=empty
  }
}

Add_wall_and_trajectory_to_global_map(d_wall,d_seg)
{ orthogonalize(d_seg,d_wall)
  dim=compute_wall_dimensions(d_wall)
  orient=compute_wall_orientation(d_wall)
  aux_grid=create_auxiliary_map(dim)
  Repeat(for each discrete position 'p' in d_seg)
  { add_possibility_values_to_global_map(p,min)
    wall_p=compute_the_corresponding_wall_position(p)
    sing_pt=check_if_wall_p_is_singular_point(wall_p)
    add_necessity_values_to_aux_grid(wall_p,orient,sing_pt,max)
  }
  add_auxiliary_map_into_global_map(aux_grid,probabil_sum)
}
```

Following the interpretation of the Possibility/Necessity assignments as Belief/ Plausibility values, we can now justify the use of the two different combination rules described above. On one hand, we have already seen that we apply the probabilistic sum when combining independent wall detections in the same cell, and this operation is nothing but Dempster's rule for simple support masses [6]:

$$\begin{aligned}
Bel_1(wall) &= \alpha_1 \\
Bel_2(wall) &= \alpha_2 \\
Bel_{1,2}(wall) &= (m_1 \oplus m_2)(wall) \\
&= \alpha_1 + \alpha_2 - \alpha_1 \cdot \alpha_2,
\end{aligned}$$

with $m(\Omega) = (1 - \alpha_1) \cdot (\alpha_1 - \alpha_2)$.

On the other hand, we also combine values coming from a single wall segment detection, and since in that case we are considering non-independent pieces of evidence, Dempster's rule is not suitable for evidence combination. Instead, we have used a max-combination, a cautious operation whose results are still within the evidence theory framework. Indeed, max-combination is in accordance with the so-called "combination of compatible Belief functions" [5] that makes sense when interpreting Bel/Pl values as bounds of the probability measures consistent with them. Namely, let

$$F_i = \{P | Bel_i(A) \leq P(A) \leq Pl_i(A)\} \quad i = 1, 2,$$

be the family of such probabilities [6]. Then, their natural combination can be taken as the intersection:

$$\begin{aligned} F_{1,2} &= F_1 \cap F_2 \\ &= \{P \mid \max(Bel_i(A), Bel_2(A)) \leq P(A) \leq \min(Pl_i(A), Pl_2(A))\}. \end{aligned}$$

In general, $\inf_{P \in F_1 \cap F_2} P(A)$ and $\sup_{P \in F_1 \cap F_2} P(A)$ are not a pair of Belief and Plausibility values [5]. However, in our particular case, this combination leads to a proper belief function. Indeed, the function Bel^P is defined as

$$\begin{aligned} Bel^P_{1,2}(wall) &= \inf_{P \in F_1 \cap F_2} P(A) \\ &= \max(Bel_1(wall), Bel_2(wall)) = \max(\alpha_1, \alpha_2) \\ Bel^P_{1,2}(\emptyset) &= m(\{\neg wall\}) = 0 \\ Bel^P_{1,2}(\Omega) &= 1\,. \end{aligned}$$

This is a belief function whose corresponding mass assignments are:

$$\begin{aligned} m(\{wall\}) &= \max(\alpha_1, \alpha_2) \\ m(\emptyset) &= m(\{\neg wall\}) \\ m(\Omega) &= 1 - \max(\alpha_1, \alpha_2)\,. \end{aligned}$$

Moreover, in this particular case, this max-combination is also in accordance with the new combination operation proposed in [24].

Local propagation

The method used to update wall information in the global map (that is, the combination operation we have just seen) takes advantage of the fact that wall information is given as segments that come from a single wall following. This allows us to reinforce overlapping, but independent, wall segments. Nevertheless, this presents the disadvantage of forcing the use of an auxiliary grid map. This implies that the updating process can not be considered as being purely local: it is local at the segment level but not at the cell level. If we want to remain purely local, the computation of the necessity value of a cell must be done by considering only those cells that surround it.

We propose here an alternative method to include wall information. This method is purely local. Nevertheless, with the information stored in the cells it is not possible to distinguish whether two necessity values are independent or not. Or in other words, the algorithm cannot know if new propagated values that arrive to a given cell come from the same wall following or not. Therefore, there is no way of discerning when two necessity values should be combined using the maximum operator or applying the probabilistic sum. This restriction forces local propagation to use only one operation to combine necessity values. We have chosen the most conservative one: the maximum operator. Being conservative reflects the fact that we prefer not to reinforce independent wall information than to reinforce non-independent wall information. We compute the height —i.e., necessity value— of the central cell of the error rectangle, at position (i, j), with (1) as

$$n^t_{i,j}(wall|\langle i, j, e_x, e_y\rangle) = 1 - \frac{\max(e_x, e_y)}{K}.$$

Next, we need to define the propagation of this measure from any cell (n, m) within the error rectangle to its four neighbours. The propagation follows the following inequalities:

$$\begin{aligned}
n^t_{n-1,m}(wall|\langle i, j, e_x, e_y\rangle) &\geq n^t_{n,m}(wall) \cdot \max(0, 1 - \frac{|(n-1) - i|}{e_x}) \\
n^t_{n+1,m}(wall|\langle i, j, e_x, e_y\rangle) &\geq n^t_{n,m}(wall) \cdot \max(0, 1 - \frac{|(n+1) - i|}{e_x}) \\
n^t_{n,m-1}(wall|\langle i, j, e_x, e_y\rangle) &\geq n^t_{n,m}(wall) \cdot \max(0, 1 - \frac{|(m-1) - j|}{e_y}) \\
n^t_{n,m+1}(wall|\langle i, j, e_x, e_y\rangle) &\geq n^t_{n,m}(wall) \cdot \max(0, 1 - \frac{|(m+1) - j|}{e_y}).
\end{aligned}$$

We use inequalities because we want to keep the maximum of the different propagated values. For instance, if a propagation is made from cell (n, m) to cell $(n-1, m)$, then the propagated value that $(n-1, m)$ receives is smaller than the necessity at (n, m). Although the inequalities allow to compute afterwards a new propagation back from $(n-1, m)$ to (n, m), the $\geq$ constraint prevents (n, m) to change its necessity value because the propagated value will be smaller than the one at (n, m). The propagation finishes when the values become stable in all cells.

Finally, the N measure is updated as follows:

$$N^t_{n,m}(wall|\langle i, j, e_x, e_y\rangle) = \max(N^{t-1}_{n,m}, n^t_{n,m}(wall|\langle i, j, e_x, e_y\rangle))$$

Once again, free-space can be computed with the previous inequalities from $N^t_{n,m}(\neg wall|\langle i, j, e_x, e_y\rangle)$, which is translated into possibility values $\Pi^t_{n,m}(wall|\langle i, j, e_x, e_y\rangle)$.

Figure 12 depicts the results of combining wall information with the *max* operator only. This figure includes three different combinations of pairs of walls. With the aim of illustrating the difference between using and not using reinforcement, these walls are the same ones that appeared in figure Fig. 11. Since the results without reinforcement are poorer, reinforcement will be used in the sequel.

Fig. 12. Local necessity combination using the max operation. The walls are the same ones as in Fig. 11.

4 Implementation: map usage

Wall extension

As we have already said, robots move pseudo-randomly in free space. When a robot detects a wall, or obstacle, it follows it along a random distance (that depends on the turning probability of the robot) and marks the trajectory segment so that the host can compute the corresponding wall segment. Following a wall along a random distance implies that very often the robots leave the wall they are following before reaching its end. This is done in order to increase the number of discovered features and to avoid robots going around an obstacle once it is detected. In general, random distances force robots to leave walls before reaching their end in order to look for other detections. If the robot reaches an end of an obstacle, this end point is labelled as *singular point* to represent the fact that the robot found a discontinuity in the shape of the object (i.e., a corner or an open door).

Both wall and trajectory segments are discretized into grid cell positions, and for each position, we have seen that necessity and possibility values are assigned to the corresponding cells in the grid. When the position corresponds to a singular point, the cells are marked accordingly. In this manner, although the segments are not explicitly represented as such in the grid, the host can treat neighbour cells with positive necessity values and equal orientation labels as belonging to the same piece of wall or object. Obviously, if this part of wall or object has cells containing singular point labels, it means that there is a discontinuity in its shape. (It could correspond to a corner, an open door, etc.) In the sequel, we will refer to these cell groups as wall segments because it is for this kind of environment features that the extension we propose makes more sense. Following the grouping cells idea, we can think of the global map as an implicit representation of wall segments and trajectories.

The obtained map is as reliable as the information error allows, but it is also relatively limited. Actually, we can ensure that wall segments without singular points correspond to longer walls, and therefore, we can extend them. However, there is not enough information in the map to know the magnitude of the extension. In the absence of knowledge about real wall lengths, we use some criteria to limit the extension of wall segments. First, since trajectories represent free space, they are used as extension bounds. And secondly, it seems reasonable to stop extending a wall segment when it meets either another detected wall segment or another segment extension. In the latter case, if the wall segments have the same direction, then they can be considered as being part of the same wall. On the contrary, if they have perpendicular directions, the obvious interpretation is to consider that they form a corner, so that their junction can be labelled as a *hypothetical singular point*.

Extension is nothing but a conjecture—there are no evidences to support it. As a consequence, to be conservative, the host only extends orthogonal wall segments (i.e., vertical or horizontal but not oblique ones). This is because, on the one hand, it is more likely that orthogonal wall segments actually correspond to real walls, and since walls are in average longer than the distance they have been followed. On the other hand, oblique features usually correspond to doors or other less usual objects. Thus, considering that it is more difficult to predict the real shape of oblique features, it is safer not to extend them.

Extension is done locally by propagating low constant certainty values of occupation for those cells in the segment extremes. These low necessity values are set to 0.1 in order to reflect that they are just assumptions and do not correspond to actual robot detections.

As we have already said, segments are not explicitly represented in the grid. Therefore, the extension algorithm must search through the grid for those cells that belong to the extremes of a group of cells that can be considered as a wall. These cells are characterised by two conditions. On the one hand, they must correspond to a discontinuity in the necessity values. And, on the other hand, the discontinuity must be in the direction of the wall segment. In algorithmic terms, this means that this cell must have a positive necessity value, and one of its neighbours in the direction of its orientation must have a zero necessity value. In this manner, if for example there is one cell with positive necessity and horizontal orientation, then either the cell on its right or the one on its left must have zero necessity value (up and down respectively for vertical segments). Each time a cell with one neighbour with zero necessity value is localised, it is necessary to check if it can be extended by one cell (that is, if the stopping conditions do not apply). If this is the case, the neighbour is marked for extension (with an “extendable” mark), otherwise the cell is marked as “non-extendable”. Once all cells have been marked, the extension algorithm assigns 0.1 necessity values to cells with "extendable" marks and the "extendable" marks are removed.

Wall extension has two main advantages: first, it increases the coverage of the real environment, and second, planning over the resulting maps gives more conservative and safer paths than those obtained considering just detected features. As we will see in the following subsection, these paths are computed from a visibility graph of the free space and they guide robots towards less explored areas. In Sect. 5 we will see how the need for reactivity is reduced when robots follow paths computed from expanded maps.

Path planning

The robot troop implements a distributed solution to the map building problem and allows an increased coverage of the environment. Although a total coverage cannot be guaranteed, we try to complete the map as much as possible. This is done in two steps: first, the host computes the shortest path to uncovered areas; and second, this path is used as a guide to new robot missions to explore those areas. The host analyses the map with the aim of extracting those features that are useful for planning. In fact, this analysis consists of a schematization method, which transforms the certainty grid map representation into a graph.

There are several well known schematization methods in the literature. We have chosen the "Visibility graph" method [21]. The obtained visibility graph consists of nodes representing vertices of obstacles that are linked by arcs whenever they are "visible" or accessible from each other. That is, whenever there is a straight line trajectory through free space joining the obstacles' vertices. Next, planning a minimum distance trajectory between any two cells is easily achieved by a minimal path searching algorithm. We have used the A^* algorithm with the Euclidean distance as heuristic function (it is an admissible and consistent function that guarantees the algorithm's completeness). This method generates the shortest path between two positions so that the resulting path is a sequence of rectilinear segments linking the vertices of those obstacles that obstruct the ideal direct trajectory. For safety reasons, we represent vertices of obstacles by a point in the free space area next to them. Consequently, we can no longer guarantee optimality in the length of the path between any two positions, but the planned paths are safer and nearly optimal.

Path following strategy

Paths are planned at the host so that they connect two positions selected by the user of the host application. The user is also in charge of assigning the resulting paths to certain robots. The behaviour-based strategy that robots apply to follow the planned paths has been already explained in Sect. 2. Basically, one behaviour tries to follow the given path and the remaining ones implement the reactivity necessary to deal with unexpected obstacles, i.e., those that were not previously detected and obstruct the path.

5 Experimental results

The global performance of a group of real robots has been simulated. This simulator includes random errors in robot movements and allows us to illustrate the whole process of exploration, map generation, map expansion, path planning and path following. In order to give a general idea of our results, we reproduce here some screens over two different environments.

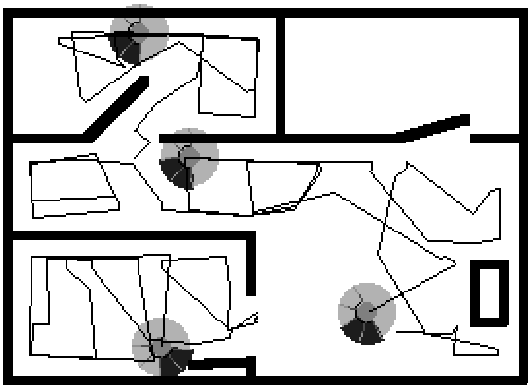


Fig. 13. Four robots explore an unknown environment.

We have used the environment in Fig. 13 to test how the exploration covers an unknown environment (which is 5.3 m. long by 3.8 m. wide and has a 29.4 m. of walls). Fig. 13 shows 5 robot exploration trajectories: they appear in their final positions and their initial positions are distributed along the environment for a better coverage ([3]). The host builds the global map in an incremental way, that is, it adds the information of a partial map each time a robot delivers it. Therefore, we can measure how the global map covers the environment by computing the percentage of detected walls with respect to the total amount of walls in the environment. The data in Table 1 shows the environment coverage each time one more robot delivers its exploration information to the host:

The column labelled "Detection" shows the coverage obtained by considering the portions of wall segments that have been actually followed. Images a), c), e), and g) in Fig. 14 show how the global map is incrementally generated (it contains both wall and trajectory information). The next column in the table indicates the percentage of correctly expanded wall segments.

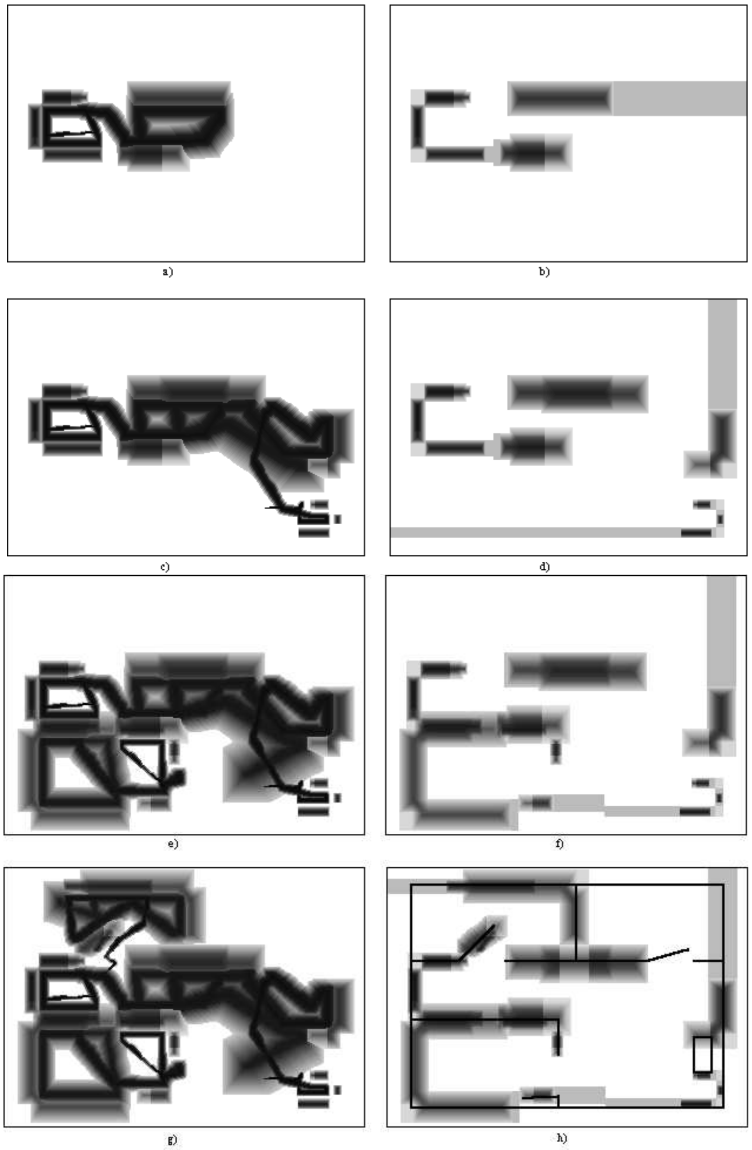


Fig. 14. Incremental global map generation: a),c),e),g) global maps after the ordered addition of partial map information (Portions of trajectories whose error exceeds a maximum error are not considered); b),d),f),h) corresponding extensions (h contains the real environment in order to evaluate the results). Wall detections in medium grey, extensions in light grey, trajectories in dark grey and singular points in very light grey (darker scales mean more certainty).

Table 1. Environment coverage considering an increasing number of robots

# robot maps	Detection	Expansion	Incorrect expansion
1	16.2%	22.4%	1.4%
2	30.1%	55.1%	0%
3	45.7%	57.9%	3.4%
4	57.1%	71.4%	3.4

Images b), d), f), and h) in Fig. 14 show the detected walls and their expansion. As we have already said, expansion is not a completely reliable process, and the rightmost column gives the error. Obviously, the more information the host has about the environment, the less chances the expansion process has of making wrong assumptions about the environment. Unfortunately, the percentages are strongly dependent on the robots' trajectories and the environment's topology. Thus, these figures can only illustrate the improvement without giving general results. In fact, the environment in Fig. 13 already shows an example of this dependency. The reason is that we have been considering all walls in the environment, including the ones in the upper right room. We should not take into account these walls because the robots cannot cross the almost closed door of this room, and therefore, walls inside become unreachable.

Using a different environment, the following results compare the planning of paths in expanded and non-expanded maps (In this case 45.3% of the environment has been detected, 70.1% correctly expanded and 6.4% incorrectly expanded.). Fig. 15 a) shows planned paths in the non-expanded map and Fig. 15 b) the paths in the expanded map. Paths are labelled with the same number to emphasise that they have the same initial and goal position. Nevertheless, since they consider different maps, they yield different trajectories.

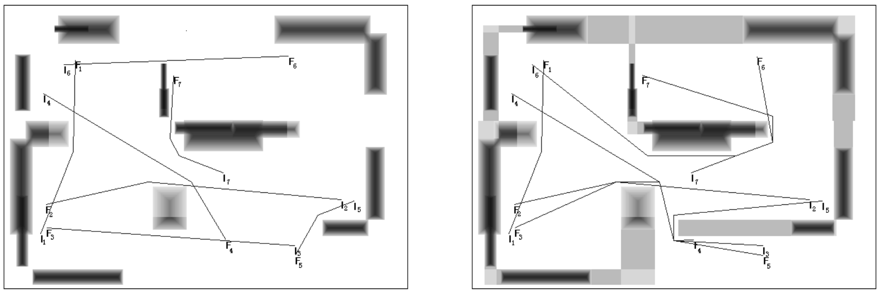


Fig. 15. Paths form a) A non-expanded map and b) An expanded map.

The images in Fig. 16, 17, and 18 depict the trajectories actually performed by robots following paths planned over the non-extended and the extended map respectively. Small circles in the trajectories represent the planned path positions. Obviously, in areas with the same information both perform equally well (see paths number 1 and 2). When one map is more accurate than the other, incorrect expanded information yields to longer paths than necessary, while non-detected walls may result in the need to engage in extensive reactive manouvers.

Sometimes, an incorrect expansion closes narrow gaps that require the use of reactive capabilities, like real-time obstacle avoidance, to get through. Therefore an initially shorter path in the non-expanded map can result in an actual trajectory whose length is in fact comparable to the length of a trajectory from the expanded map (this is the case of paths number 3). Of course, this is not always the case: path number 4 in the expanded map avoids a non-existing piece of wall causing an extra displacement. The worse case of avoiding a non-existent wall is path number 5 in the expanded map: it turns around the corner where a different robot previously went. Although this tendency of going over other robots' trajectories (i.e., already explored free space) can be a disadvantage, it is also the source of more conservative and safer paths.

Finally, when expansion is correct, the resulting paths are more informed and reactive manouvers are less often needed (see path number 6). Reactivity is less efficient because it causes the robot to decide how to avoid an obstacle without knowing its shape. This can yield the problem that appears in path number 7 in the non-expanded map. Here, reactivity makes the robot take so many wrong decisions that the robot stops before reaching the target. In fact, the robot stops because the rectangle error associated with its position grows so much that it includes the target coordinates (although the robot is still quite far from it).

6 Discussion

The work presented here describes an approach to the problem of building maps of unknown structured environments by means of small autonomous robots. The unavoidable imprecision of the sensors induces uncertainty in the information gathered by the robots. We have used Possibility Theory to treat the uncertainty associated with the position of the obstacles as well as of the free space. In our maps we represent robots' trajectories (which represent free-space) and wall following information (that reflect occupancy).

The use of Possibility Theory has two main advantages: it allows us to represent the initial total ignorance about the environment to be explored; and it provides operators that combine values in a very intuitive way. These advantages fairly compensate the minor disadvantage, which is the lack of a unique assignment of possibility and necessity distributions. We have assigned

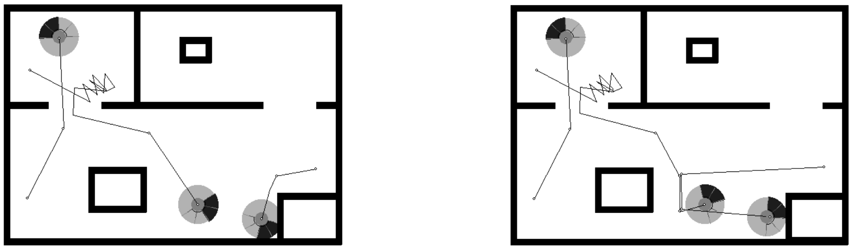


Fig. 16. Resulting trajectories when trying to follow the planned paths to the goal positions 1, 4, and 5. Left: on the non-expanded map. Right: on the expanded map.