Planning Exploration Strategies for Simultaneous Localization and Mapping

Benjamín Tovar† Lourdes Muñoz‡ Rafael Murrieta-Cid‡
Moisés Alencastre‡ Raúl Monroy‡ Seth Hutchinson†

† University of Illinois
Urbana, IL 61801 USA

‡ Tec de Monterrey
Edo de México Campus

btovar@uiuc.edu lmunoz@itesm.mx rafael.murrieta@itesm.mx
malencastre@itesm.mx raulm@itesm.mx seth@uiuc.edu

Abstract

In this paper, we present techniques that allow one or multiple mobile robots to efficiently explore and model their environment. While much existing research in the area of Simultaneous Localization and Mapping (SLAM) focuses on issues related to uncertainty in sensor data, our work focuses on the problem of planning optimal exploration strategies. We develop a utility function that measures the quality of proposed sensing locations, give a randomized algorithm for selecting an optimal next sensing location, and provide methods for extracting features from sensor data and merging these into an incrementally constructed map.

We have also provide an efficient algorithm driven by our utility function. This algorithm is able to explore several steps ahead without incurring too high a computational cost. We have compared that exploration strategy with a totally greedy algorithm that optimizes our utility function with a one-step-look ahead.

The planning algorithms which have been developed operate using simple but flexible models of the robot sensors and actuator abilities. Techniques that allow implementation of these sensor models on top of the capabilities of actual sensors have been provided.

All of the proposed algorithms have been implemented either on real robots (for the case of individual robots) or in simulation (for the case of multiple robots), and experimental results are given.

1 Introduction

Autonomous robots must possess the ability to explore their environments, build representations of those environments, and then use those representations to navigate effectively in those environments. Collectively, these requirements define the problem of Simultaneous Localization and Mapping (SLAM) [5, 9, 24, 32], which has become a very active topic of research in the last decade.

To date, work on SLAM has focused primarily on issues related to uncertainty in sensing. Early research on SLAM [32] used a Kalman filtering approach to manage uncertainties that accumulate during robot motion, simultaneously providing an estimate of robot position and landmark locations. More recently, generalized Bayesian approaches have been proposed (see, e.g., [24]) for the SLAM problem, relaxing the restrictive conditions imposed by Kalman filtering methods.

In this paper, we address issues related to uncertainty in sensing and control, but our primary focus is on the problem of planning optimal exploration strategies. In particular, we develop a formalism for planning exploration strategies that optimize criteria such as information gain, uncertainty reduction, etc. We then present methods for extracting features from sensor data (predefined landmarks and geometric features such as corners), and fusing these features into a common, global map. The result is a robot that autonomously explores its environment, optimizing the exploration at each stage and merging newly acquired sensor data into its existing map. Preliminary versions of this work appeared in [35] and [36].

Our proposed utility function is constructed in such a way that it balances the desire to see as much of the as-yet-unseen environment as possible, while at the same time having enough overlap and landmark information with the already scanned part of the indoor environment to guarantee good map registration and robot localization. The exact form of this utility function (which is presented in Section 3) is fairly complex, which precludes solving the optimization problem in real time. Therefore, optimization of the utility function is achieved by a randomized sampling scheme.

Computer vision is used to recognize landmarks using a Bayesian approach. A laser range finder is used to find straight lines in the environment (using least squares fitting), and lines obtained in consecutive sensing operations are fused by minimizing a partial Hausdorff distance. The final result of the exploration is a multi-representational map consisting of polygons and landmarks, and including a roadmap (backtracking graph) constructed from the trajectory followed by the robot.

All of the proposed algorithms have been implemented. For the case of a single robot, the algorithms have been tested on a real robot, and experimental results are included in this paper. Algorithms for muti-robot map building have been implemented and tested via simulation.

The remainder of this paper is organized as follows. Previous research is discussed in section 2. The utility function that is used to evaluate candidate sensing locations is presented in section 3. In section 4 we briefly describe the landmarks and features that are used by our system. In section 5 we give the map building algorithm. A planner for multi-robot map building is presented in section 6. The robot architecture, experiments and results are

described in section 7. Conclusions and future work are given in section 9.

2 Related Research

Automatic model building is an important problem in mobile robotics [4, 6, 10, 34]. Three types of models have been mainly proposed: i) topological maps [7], ii) occupancy grids [10] and iii) feature-based maps [6, 21, 33].

Topological maps can be expressed as a graph, where the nodes represent places and the edge represent adjacency, or direct connectivity. Occupancy grids use a 2D array to represent the environment. There, each cell takes one of three values: free space, occupied space or unknown space. Grid-based algorithms have proved to be very simple and quite useful for obstacle avoidance and planning purposes [10]. However, when the size of the environment is large, these models become difficult to handle.

Feature-based maps [21] may portray a 2-D [6] or 3-D model [33]. They are another way to represent the environment by using geometric primitives. The most popular geometric primitive is the segment, which can be extracted from ultrasonic data [8], laser range-finder data [14], or vision data [3, 17].

Most previous research has focused on developing techniques to extract relevant information from raw data and to integrate the collected data into a single model; a robot motion strategy is, however, typically not developed. In this work, we deal mainly with this latter problem. An account of the field follows.

In [13,15] a map building motion planning strategy is presented. That research has shown that it is possible to find a function that reflects intuitively how the robot should explore the space. In a simple scheme, the evaluation function should assign a greater value to the position that best fits the compromise between possible elimination of unexplored space and travelled distance.

The approach presented in [22] proposes an exploration strategy for map building and localization. The exploration strategy makes use of a utility function that evaluates the next robot sensing location. This utility takes into account three elements: the information gain, the distance to sensing location (cost) and the utility of localizability based on a covariance matrix.

In [27] an algorithm for feature-based exploration of an unknown environment is proposed. In that approach the candidate next robot locations (goals) are associated with all the geometrical features of the environment. One major objective in [27] is to locally explore spare regions. To obtain that objective, the following procedure is proposed. First, for each goal generated, sample points are regularly scattered around it at a constant radius β. A circle of radius α centered on each sample is then drawn. The final size of the sampling set is the number of sample points for the goal at hand that satisfy these conditions: i) each point has a clear line of sight to the goal, and ii) it has no line sight to any other circle of radius α. The score $\eta \in [0, 1]$ for evaluating a goal location is set to be the ratio of the final and initial size of the sampling set.

The work presented in [31] deals with the problem of exploration and mapping of an

unknown environment by multiple robots. A probabilistic grid is used to represent the environment. The cells of the grid are of three types: Obstacle (probability of occupancy above a given threshold *po*), clear (probability below a threshold *pc*) and unknown (either never been sensed or probability between *po* and *pc*). The distance between the robot and the frontier between known and unknown environment is used as an exploration cost. The number of unknown cells that fall within the robot sensor range (for a possible next location) is used as the information gain of sending the robot to that location. The utility of a candidate sensing location is equal to its information gain minus its exploration cost. A central executive tries to maximize the total utility. To coordinate the robots (relate robots with sensing locations) the information gain is used.

Another robot motion exploration strategy is presented in [11]. There a metric for adaptive sensing that is defined in terms of the Fisher information is proposed. This metric is used to choose among discrete robot actions that given the current state would locally maximize the information gained in the next measurement. As a result of applying that algorithm the robot tends to explore selectively different objects in the environment.

Other approaches have been proposed that are related to our research, even though they are not directly intended for planning exploration strategies for SLAM.

In [12], software tools applied to multi-robot, distributed-robot and sensor network systems are proposed. That software is composed of two main elements: *Player and Stage*. *Player* is a robot device server that provides sensing and control algorithms. *Stage* is a multiple robot simulator that provides a population of simulated robots and sensors operating in a bitmapped environment. The main goal of the *Player/Stage* project is to provide Open Source software interface to support experimental research with multi-robot systems.

In [19], the CentiBOTS project is presented. In that project the authors envisage a system of a large number of robot able to explore, map, and surveil the interior of a building. One major contribution of that work is a distributed robot architecture, that is adaptive and fault tolerant.

3 Evaluating Candidate Sensing Locations

In this section we describe a utility function that can be used to evaluate the quality of a proposed next sensing location. We begin with a general discussion of the desirable attributes of such a function, and then present the specific utility function that we use.

A good utility function should prefer those positions from which the robot can recognize a landmark that can be used either to reduce uncertainty in position [29] or to perform navigation tasks [20, 25]. Here, we define a landmark as a predefined, recognizable object in the work space. We do not assume that landmarks can always be recognized without error by our computer vision system. Rather, we assume that recognition will be performed by a Bayes classifier, and that this classifier will provide both a classification and an error probability. Thus, our utility function should prefer landmarks with high probability of recognition to those with low probability of recognition.

In addition to landmark recognition, the robot's sensors are also used to extract features

(e.g. corners) that can be used to facilitate the fusion of data obtained from distinct views. Therefore, in addition to maintaining landmark visibility, the utility function should prefer sensing locations that maximize the number of visible and readily discernable features. In our work, we have used corners in the environment as features, and thus the utility function should prefer sensing locations from which a maximal number of corners are visible.

To explore the environment as quickly as possible, sensing locations that provide maximal views of unexplored areas should be preferred. Unfortunately, without an existing map of the environment, it is not possible for the robot to know where the unexplored areas lie. One way to approximate the size of the unexplored space is to use the size of the free edge near it. A free edge is defined as the border between regions of explored and unexplored space. Thus, our utility function will prefer sensing locations that lie near long free edges.

As SLAM research has shown, one of the principal difficulties in constructing maps of large environments is that odometry is typically imprecise, and that localization error grows as the robot moves. This complicates the process of fusing new sensor data with existing representations. A good utility function should therefore prefer trajectories that minimize localization uncertainty. We use a simple model of uncertainty. The robot position uncertainty grows in proportion to the square root of the distance traveled. In this uncertainty model we also include a term that penalizes rotation, reflecting a preference for straight line trajectories.

In the exploration process, at a given position, the robot may have more than one area to explore. Thus, some unexplored areas are postponed to be explored. To come back to those unexplored areas the robot travels a road-map built during exploration. Every node in the graph is a previous sensing location where the robot stopped and built the map. Only at those locations the robot is allowed to turn. In our experiments, we have found out that robot acceleration and decelerations will increase the robot position error. Thus, minimizing the number of stops to arrive at an unexplored area amounts to minimizing the robot position error. Our utility function is so that the robot moves back to an unexplored area travelling the road-map portion that requires the minimum number of stops—nodes—. Put another way, our utility function was designed to give a better score to the robot trajectory with the minimum number of stops.

Power consumption is another problem that confronts autonomous mobile robots. To minimize power consumption, a good utility function should prefer trajectories that minimize the total distance that the robot must travel.

Our utility function integrates all of these features, preferring positions combining proximity to the robot, proximity to a free edge, small robot configuration uncertainty, high landmark identification probability and ability to see features like corners. Furthermore, positions near walls and objects will be discarded because many sensors become blind when the objects are very near.

We have chosen a utility function with a multiplicative form. A configuration with a very low value on at least one of these measures will be discarded even though it could have a very good value on the others. For instance, a position very close to a free edge (with great chance of discovering new space) must be discarded if the robot has no information to integrate this

i	Location where a sensing operation is done.
m	Total number of sensing operations.
q_i	Total number of robot stops to reach location i
lv_i	Length of the closest free edge at location i
s_i	Distance from the robot to the next possible location i
sv_i	Distance from the next possible location i to the closest free edge
j	Index for a robot configuration
θ_j	Orientation change to reach the next configuration j
p_k	Identification probability of landmark k at location i
k	Index for a given landmark
n_i	Number of landmarks inside a visibility region at location i
Ne_i	Number of corners inside robot visibility region at location i
$fmin_i$	A function that penalizes location i.
d_l	Minimum distance from a full edge

Table 1: Definitions of variables used in the utility function (1).

new area to the explored space. Similar forms of this utility function have been presented in [35,36]. *The main difference between our previously proposed utility functions and the one presented in this paper is that now our utility function can be used to measure the utility of a single robot location or a path associated to a sequence composed by several sensing locations.* The utility of a sequence of sensing locations will be simply the summation of the utility of each location. Consider the variables definitions given in table 1, then our utility function $\mathcal{T}_i$ is given by the following expression:

$$\mathcal{T}_i = \sum_{i=1}^{m} \left(\mathrm{e}^{(lv_i - sv_i)} \prod_{j=1}^{q_i} \left(\frac{\mathrm{e}^{-|\theta_j|}}{\sqrt{s_j}+1} \right) \left(\frac{1}{n_i} \sum_{k=1}^{n_i} p_k + Ne_i \right) fmin_i(d_l) \right). \tag{1}$$

The function $fmin_i$ is used to map the distance from an obstacle edge to a utility value. To minimize effects of occlusion, it is desirable to maximize the distance to obstacles. On the other hand, sensors such as sonars have a finite range. Therefore, beyond some threshold distance, no advantage is gained by moving further from an obstacle. For this reason, we have chosen for $fmin$ the mapping shown in Figure 1. When the distance to the nearest obstacle is less than a threshold distance t the function takes a low value, discriminating those positions near the objects. Once this threshold distance is exceeded, $fmin_i$ takes the value of 1, allowing the remaining parameters to influence the value of $\mathcal{T}$.

The parameters of the utility function are graphically described in the scheme presented in Figure 2.

Our utility function quantifies different a sensing location depending on the path that the robot has traveled to reach that location. We assume that the amount of the uncertainty will vary according to the controls applied to the robot. To better clarify our statements about the uncertainty induced by the types of paths traveled by the robot, we show below the score that our utility function gives to different robot paths.

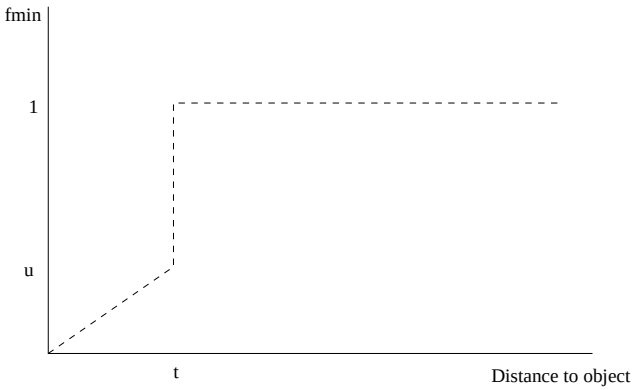


Figure 1: The function $fmin$, which penalizes sensing locations that lie near on obstacle.

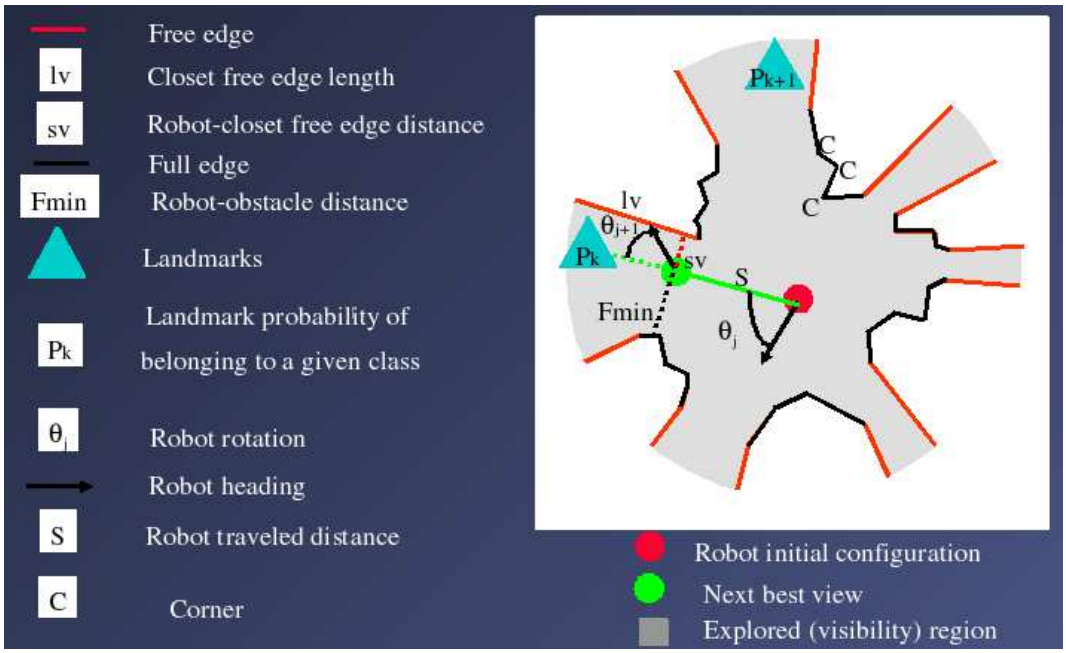
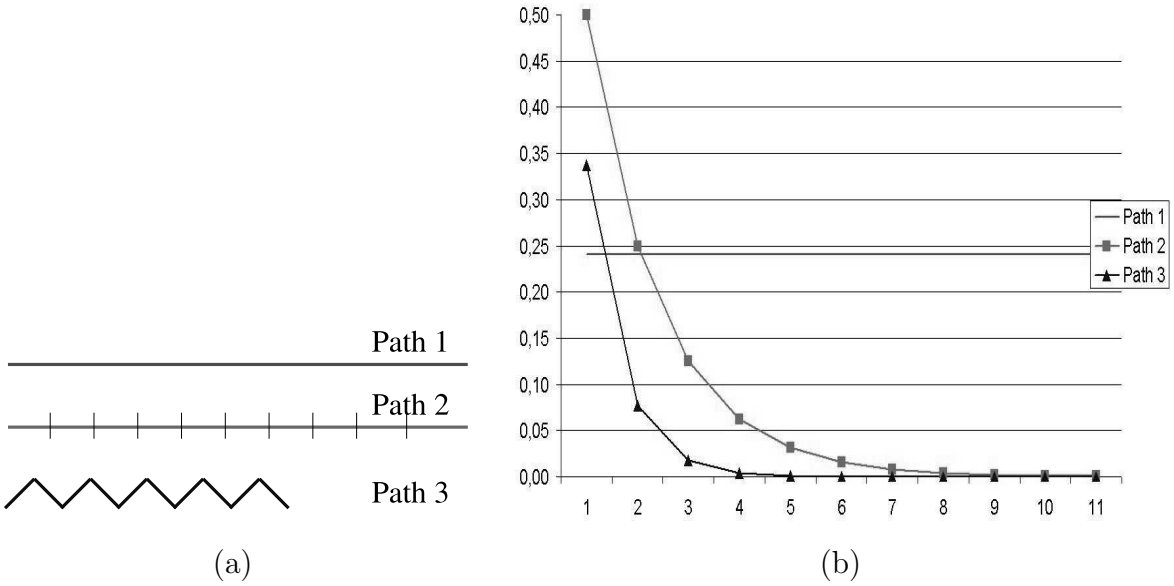


Figure 2: Parameters of the utility function



(a) (b)

Figure 3: (a) Three different paths (b) Score given by the objective function

Fig. 3 a) shows three different types of paths. The first path is a straight segment indicating that the robot goes from an initial to a final configuration in only one step. The second path is also a straight segment, but in this case the robot stops at the vertical black lines and then continues. Finally, in the third path the robot moves in zig-zag, changing its angle at each step. Note that to come back to unexplored areas through the road-map, a sensing location may be reached in more than one single step. The utility of each path is calculated using the second term of our objective function:

$$\prod_{j=1}^{m} \frac{e^{-|\theta_j|/\lambda}}{\sqrt{s_j}+1} \tag{2}$$

where:

- j is the the step index;
- m is the total number of steps;
- θ_j is the angle the robot must turn, given in radians;
- s_j is the distance the robot will travel in the current step; and
- λ is a constant.

We have plotted the utility values of each path. For path 1, there is only one single value calculated after the robot has reached the final configuration. For paths 2 and 3, there are several values, one for each time the robot stops. These plots are shown in Fig. 3 b). Note

that the more times the robot stops, the lower the utility value of the path. The lowest score is obtained when the robot stops several times and then turns to change its heading.

To our knowledge, *the effect of applying different types of controls to estimating the uncertainty in robot position has not been considered before. Integrating this information we better assess the utility of a robot path*

4 Landmarks and Features

In our work, we distinguish between landmarks and features. Landmarks can be uniquely identified by the robot, while features are geometric entities without specific identities. For example, corners are used as features, while artifacts such as posters can serve as landmarks. Below we briefly describe how landmarks are used by our system, and how line fitting and model matching are used to extract line segment features and to fuse these into a single representation.

Landmark detection and identification are useful to both, localize the robot and merge different environment views. The merging becomes easier because these landmarks work as pivotal features to align these partial views [25]. Landmarks are also useful to determine the robot position relative to one or several of them [29].

Our utility function will prefer sensing locations from where corners or landmarks are visible, so making it easier for a registration procedure (in our case the Hausdorff distance, see section 4.3) to align the maps. To our knowledge *this is the first attempt where the ability to see landmarks or features is applied to estimate the utility of a sensing location.*

4.1 Landmarks

We assume that the robot is provided with a library of landmarks that can be recognized from sensor data or that it can build such a library as it explores its environment (note that this is not a requirement, as our utility function and the resulting exploration strategy can easily be modified to eliminate their use of landmarks). A good landmark is one that is easy to recognize and that provides good localization accuracy. Landmarks in indoor environments can be defined as structured elements such as posters, doors, or columns.

Recently, an indoor landmark detection method for robot navigation was proposed on [16]. The algorithm takes advantage of the geometry of indoor scenes and uses vanishing points to reduce the landmark detection complexity. This method is based on edge detection and texture measurement. The algorithm computes an edge segment image and applies two relaxation processes to match segments and segment pairs in order to detect landmarks. Our landmark detection software is still in its early stages. We plan to integrate it as a landmark extraction module in our robot architecture. We will used an algorithm similar to that proposed in [16].

Landmarks in an image can be identified using an hierarchical approach. A first step identifies the environment type, and a second step the Landmarks in the image [26].

The learning database is a function of the environment type. Techniques for image classification as a whole can be used as environment recognition method. In the first step the type of environment is determined (i.e, an office, a lab, etc), and in the second step an appropriated database (which corresponds to the environment type) is used; making it easier to label the elements in the image with a reduced number of classes.

Once an image region has been detected as a potential landmark, a Bayes classifier can be used for recognition of the landmark. In particular, a region in the image is labeled as a landmark if its probability of belonging to a given class is greater than a specified threshold and if it has distinct characteristics (color, shape, etc) from surrounding objects (regions) [25]. Furthermore, the classifier provides an estimate for p_k, the probability of correct classification of the landmark.

4.2 Extracting line segments to be stored

Since we use a laser range finder as our sensor, we recover lines that form the actual visibility region from the points that the laser gives. In particular, we generate polylines with the laser data obtained as an ordered list (by angle) of polar coordinates (r, θ) where obstacles are found. We suppose that θ is an error free coordinate. The line fitting is done in two steps. First we find clusters of points where the distance between two consecutive points is similar. Then we apply the transformations $u = \cos(\theta)/\sin(\theta), v = 1/r\sin(\theta)$ as in [13]. We fit lines to the laser data applying to each cluster a divide-and-conquer technique combined with a least squares method.

The least squares technique has the advantage of removing noisy measurements. However, it is not efficient in the number of lines it generates. For this reason we use a divide-and-conquer approach. We convert the generated vertices in the (u, v) space to a Cartesian space. Then, we apply a classical divide-and-conquer recursive technique to the vertices of each cluster to find the lines that fit the set of vertices. Thus, unnecessary vertices are eliminated. A cluster with a stand alone point or with very few points should be considered as a small object [13], a sensor error or the result of a small free space between two occlusions. In any case, those should not be taken into account when the divide-and-conquer algorithm is applied.

The lines generated are considered as full edges, while the line that may be formed between two consecutive clusters is considered as a free edge.

4.3 Fusing data from multiple views

The partial Hausdorff distance is used to find the best alignment between the previously explored region and the new one. The Hausdorff distance is computed on the original laser data of the polylines previously computed.

Given two sets of points P and Q, the Hausdorff distance is defined as (see [18]):

$$H(P, Q) = \max(h(P, Q), h(Q, P)) \tag{3}$$

where

$$h(P,Q) = \max_{p \in P} \min_{q \in Q} \|p - q\| \tag{4}$$

and $\|.\|$ is a norm for measuring the distance between two points p and q. The function $h(P,Q)$ (distance from set P to Q) is a measure of the degree in which each point in P is near to a point in Q. A small value of $h(P,Q)$ implies that every point in P is close to a point in Q. The Hausdorff distance is the maximum among $h(P,Q)$ and $h(Q,P)$. The Hausdorff distance measures the degree to which each point of P is near a point in Q and vice versa. By computing the Hausdorff distance in this way we are obtaining the most mismatched point between the two shapes compared, consequently, it is very sensitive to the presence of any outlying points. For this reason it is often appropriate to use a more general measure, which replaces the maximization operation with the calculation of the mean of the values. This measure (partial Hausdorff distance) is defined as:

$$h_k = M_{p \in P} \min_{q \in Q} \|p - q\| \tag{5}$$

where $M_{p \in P} f(p)$ denotes the statistical mean value of $f(p)$ over the set P.

One interesting property of the Hausdorff distance and the "partial Hausdorff distance" is the inherent asymmetry in the computation. The fact that every point of P (or subset of P) is near some point of Q says nothing about whether every point of Q (or subset of Q) is near some point of P. In other words, $h_{k1}(P,Q)$ and $h_{k2}(Q,P)$ can attain very different values. In fact each one of the two values give different information.
The term $h_{k1}(P,Q)$ is the unidirectional partial distance from the previously explored region to the current perception, and $h_{k2}(Q,P)$ is the unidirectional partial distance from the current perception to the previously explored region. The maximum of these two values defines the partial Hausdorff distance. The partial Hausdorff distance is function of a transformation composed by translation and rotation. The transformation that maximize the metric will determine the best alignment.

5 Map Building Algorithm

Algorithms 1 and 2 show our map building procedure. In Algorithm 2, $\mathcal{T}$ is given by Equation (1). As mentioned above, visibility is used to bias the sampling process. For a point robot, visibility is enough to guarantee a free path. If the robot is inside its visibility region, it is collision free. In real experiments, given that the actual robot occupies finite area, that is not enough to guarantee a free path. This area has to be subtracted along the boundary of the computed visibility region to determine if the robot can traverse through the sensed environment. The remaining region is guaranteed to be collision free. The samples are generated close to the free edges. Only the samples inside the observer visibility regions are considered. Thus, the generated samples will have a better chance to be useful. Since the samples are inside the visibility region of the robot, it is guaranteed that a straight line path between the current and next robot positions is collision free. Additionally, the samples

Algorithm 1 EXPLORATION($\mathcal{R}$,$\mathcal{W}$)

Input:
$\mathcal{R}$: Robot configuration
$\mathcal{W}$: Environment
Output:
$\mathcal{M}$ Environment map

P_n current robot perception
U_f List of free edge labels
$U_f \leftarrow 1$
while $U_f \neq \emptyset$ **do**
  $P_n \leftarrow$ TAKE_PERCEPTION($\mathcal{R}$, $\mathcal{W}$)
  $\mathcal{M}$.add(P_n)
  $U_f \leftarrow$ ID_UNXP_AREAS($\mathcal{M}$)
  if $U_f \neq \emptyset$ **then**
    $[x, y, \theta] \leftarrow$SELECT_NEW_CONF($U_f$, $\mathcal{R}$, $\mathcal{M}$)
    $\mathcal{R}$.move(x, y, θ)
  else
    Return $\mathcal{M}$, OK
    $U_f \leftarrow \emptyset$
  end if
end while

Algorithm 2 SELECT_NEW_CONF(U_f, $\mathcal{R}$, $\mathcal{M}$)

Input:
$\mathcal{R}$: Robot configuration
$\mathcal{M}$ Environment map
U_f List of free edge labels
Output:
q_{nbv} Robot configuration for the next best view
n number of samples per free edge

l_n list of robot configurations per free edge
L_q list of all candidate robot configurations
for $u_f \in U_f$ **do**
  $l_n \leftarrow$SAMPLES(n, u_f, $\mathcal{R}$, $\mathcal{M}$)
  $L_q = L_q \cup l_n$
end for
$q_{nbv} \leftarrow$MAXIMIZE($\mathcal{T}$, L_q)
Return q_{nbv}

generated close to the free edges will have a good chance of seeing unknown environment, because they are close to the border between the explored an unexplored environment. If the robot goes to those samples pointing toward the free edges it will perceive unexplored environment. Previously chosen samples are kept in the graph since they could be reused to reach other new samples in the future. The robot takes advantage of this by traveling in the backtracking graph. Sampling generation is illustrated in Figure 4.

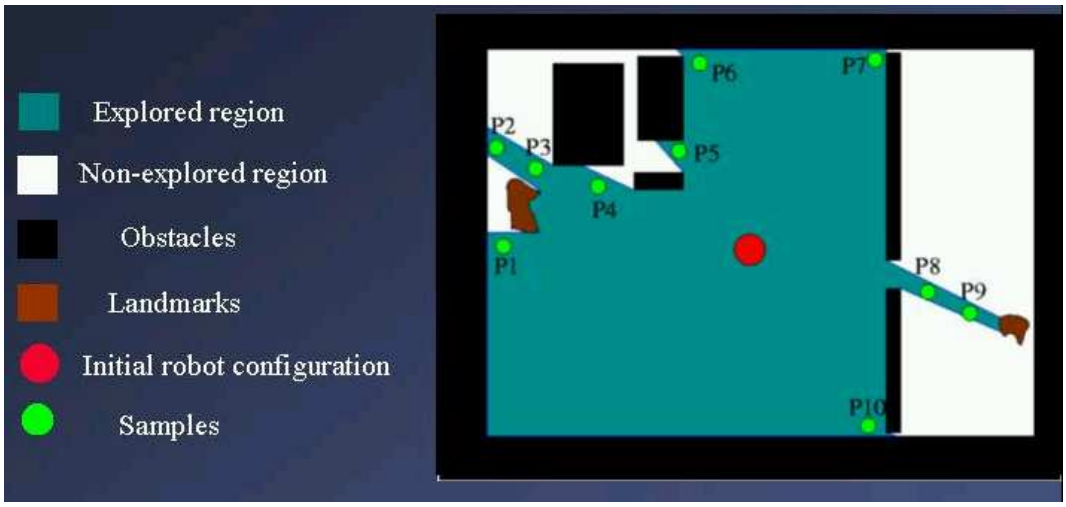


Figure 4: Sampling generation

5.1 Several steps ahead exploration: An efficient algorithm

Suppose that at some time during the exploration, the robot is at a certain position, $r_{ini} = (x_{ini}, y_{ini}, \theta_{ini})$, and there remain n frontiers (free edges) to be visited. After the sampling process, there will be several configurations for each free edge. The robot has two options to define the motion strategy: either it considers only one step ahead or considers more than one.

If it considers only one step ahead, the robot will move to the best evaluated configuration, according to the utility function. In this case, after the robot has reached the desired configuration, the robot will start the sampling process all over again, considering the remaining free edges. Otherwise, the robot will need to find a path for visiting the n free edges. In this case, both an ordering for visiting the free edges and a configuration for each free edge need to be established. This visiting order yields the maximal utility. We illustrate how our algorithm works for finding a visiting order through out an example.

Fig. 5 shows a situation where the robot has three new free edges to explore, denoted by A, B and C. We suppose there are two configurations for each of the free edges. For the free edge A the configurations are a_1 and a_2. In this case, the robot is at p_{ini}, so in the next step the robot can visit one of the six configurations from a_1 to c_2. Let us suppose that the

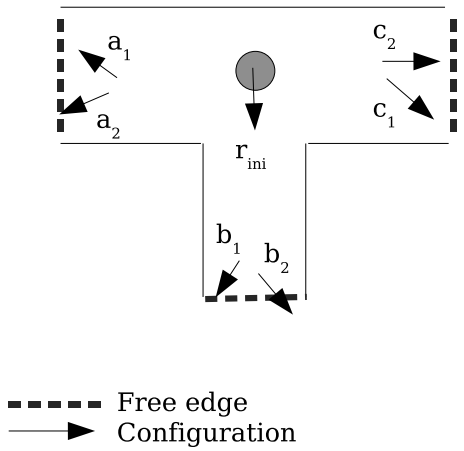


Figure 5: Robot in front of three free edges.

robot goes to b_1 in the first step. Then the robot can go to configurations a_1, a_2, c_1 and c_2 in the second step. Configuration b_2 is thus not an admissible configuration for this latter step, because the free edge B has been already explored.

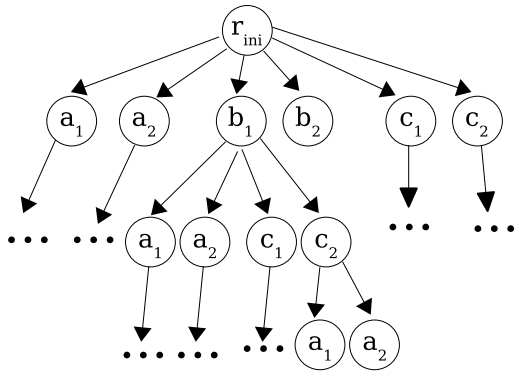


Figure 6: Three for searching a path.

We can model all possible paths that the robot may pursue to exploring the free edges using a tree. Each node represents one robot configuration. Each node's descendants represent the configurations that can be visited. Each arc represents a step in time. The tree for the running example is shown in Fig. 6. In this case, there are 3 levels pending from the root node. Note that in this case we have imposed one constraint for building the tree: the robot cannot return to a free edge that has been explored on a previous step. This means

that if a node labeled p_1 is at level 1 all the branches pending from this node will not have the node p_1 in the subsequent levels. Furthermore, the branches pending for node p_1 will not have nodes that represent configurations lying in the same free edge of p_1. The root node will have n levels pending from it according to the n free edges that will be visited.

This is not the only exploration strategy. There is also the possibility to allow the robot to return to a previously visited free edge for establishing an ordering. However we do not consider this case.

For building a tree like the one shown in Fig. 6, we need to know all the paths that can be possibly formed by permuting every robot configuration without repeating visited configurations. Our utility function is used to assign a cost to every edge. Once found, the cost of each path between two configuration in two different free edges is calculated using the utility function. Once the tree has been built, we can use a search algorithm for finding the configurations the robot must reach in order to visit all the free edges at minimum cost.

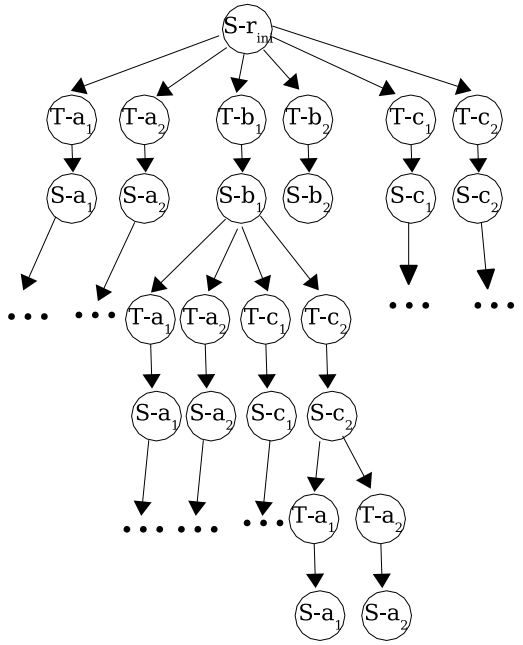


Figure 7: Searching tree considering the states of the robot.

We consider that the robot can be in one of two possible states: sensing in the configuration previously reached and traveling to the next configuration. Going back to our running example, suppose that the robot has selected the following path for visiting all free edges: r_{ini}, b_1, a_2, c_1. In this case the robot will have the following sequence of states during the execution of the path: sensing at r_{ini}, then traveling to b_1, then sensing at b_1, then traveling to a_2, then sensing at a_2, then traveling to c_1 and finally sensing at c_1. We can see here that for each node in the path, we have two states for the robot, except for the root node, in which there is only a sensing state. Considering this, we can split each node in the tree in two nodes each one for the corresponding state (traveling, sensing) and we will obtain a tree

like the one shown in Fig. 7. The notation used in Fig. 7 is the following: $S - a_1$ means sensing at a_1 and $T - b_1$ means traveling to b_1. The cost calculated with the utility function will be assigned to the edges going from a sensing state to a traveling state, and the weight for an edge going from a traveling state to a sensing state will be zero. So we are assuming that sensing yields no cost, which is not always the case.

The representation of the states in the tree may seem unnecessary, but this scheme is useful for extending our approach to the multi-robot case as will be shown below (see section 6.1).

5.1.1 Branch reduction heuristic

Suppose that at some time, the robot has n free edges to visit and there are exactly m configurations for each free edge. There will be $n \cdot m$ nodes pending from the root node. For the second level of the tree, there will be $(n-1) \cdot m$ children for each node, because the robot has already visited a free edge. So at level two the robot has to choose from the $n-1$ unvisited free edges. For the third level of the tree, there will be $(n-2) \cdot m$ children for each node, and so on and so forth.

We can see here, that the tree grows exponentially according to n and m, so as n and m are larger, the search for the optimal path is intractable. Note that the whole tree has n levels, this is the maximum depth. In this case, we can reduce the search space pruning the tree using a branch and bound algorithm [1].

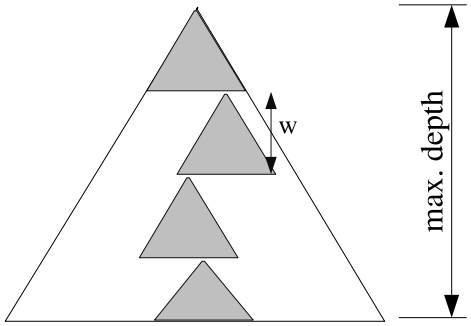


Figure 8: The big triangle represents the whole tree until max. depth, the gray part represent the reduced tree.

The idea with the branch and bound algorithm is to build the tree to a level $w < n$. At this point, we select the leaf with the minimum cost and continue expanding the tree only from that leaf on. This idea is represented in Fig. 8.

6 Multi-Robot Map Building

We have developed, implemented and simulated a planner for multi-robot map building. Our approach consists in a centralized planner. The position and current visibility from every

robot is assumed to be known by the planner.

We denote by $V(q_k)$ the visibility polygon of a robot at configuration q_k. A robot is free to move in the interior of its visibility polygon, so long as it does not collide with an obstacle. We denote by $\mathcal{F}(q_k)$ the visibility polygon reduced by the robot radius, i.e., $\mathcal{F}(q_k)$ is a safe region for navigation that is visible from configuration q_k. We define V_{tot} as the total visibility region for the ensemble of robots, and $\mathcal{F}_{tot}$ as the total visible region region in which any robot can move, i.e.,

$$V_{\text{tot}} = \bigcup_k V(q_k) \qquad \text{and} \qquad \mathcal{F}_{\text{tot}} = \bigcup_k \mathcal{F}(q_k).$$

In our multi-robot map building strategy a robot at configuration q_k has as a priority to visit those free edges that lie within $\mathcal{F}(q_k)$. Associating each robot with its visibility region reduces the complexity of the problem. If there are no free edges visible from q_k, then the robot is free to explore free edges that lie in $\mathcal{F}_{\text{tot}}$. In this case, a sampling-based evaluation determines which free edge will be visited by the robot. In all the cases, the decision is taken by evaluating Equation (1). Since Equation (1) estimates the size of the unexplored space by using the size of the free edges, exploring the unknown space is equivalent to sending a robot to visit a free edge.

6.1 Multi-robot several steps ahead exploration

The proposed exploration strategy described in section 5.1 has been extended to the multi-robot case. We have a search tree to find a plan for all robots. This plan consists in finding which robot should visit a free edge i considering it is at configuration x_i, y_i, θ_i at some time. We consider that only one robot is moving at any time while the rest are sensing. The state of the multi-robot system is defined as a vector having the states of each robot. For example one possible state of the system can be: (Robot 1 traveling, Robot 2 sensing, Robot 3 sensing). We also consider that once a free edge has been explored by one robot, none of the others will go to that edge.

Fig. 9 shows an example of a search tree for two robots with three free edges. We suppose that there is only one configuration for each free edge, so there are in total three configurations, denoted a_1, b_1 and c_1. As an example of the notation used for describing the state of the system consider $R_1 - T - a_1$, $R_2 - S - c_1$. They respectively mean that Robot 1 is traveling to configuration a_1 on free edge A, and that Robot 2 is sensing at configuration c_1.

The root node represents the initial state where all robots are sensing at their corresponding initial configuration. After that, there are six different nodes in the first level: Either robot can move to one of a_1, b_1 or c_1 while the other one is sensing. Suppose that the selected next state is Robot 1 traveling to b_1 while Robot 2 remains sensing. After this move, there are two possible states: Robot 2 traveling to a_1 or Robot 2 traveling to c_1. In any case, Robot 1 will be sensing at b_1 [1]. As only one robot moves at a time, the cost assigned to any

[1]Note that in Fig. 9, we have only developed the first two nodes of the second level

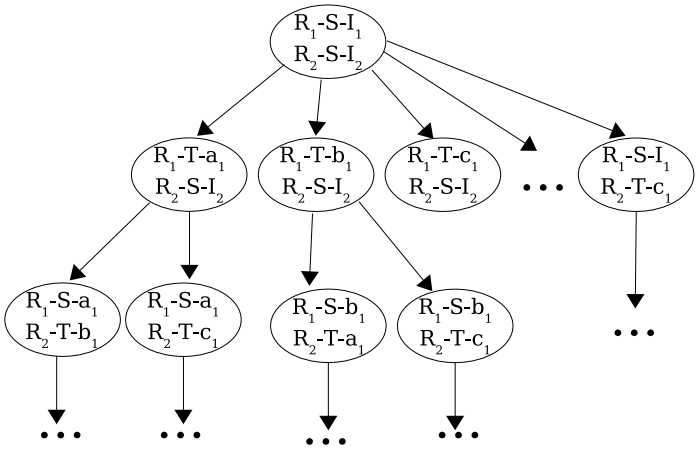


Figure 9: Search tree for two robots having two free edges and only one configuration for each free edge.

one arc corresponds to the cost associated with the moving robot. For example if the system is at state $(R_1 - S - a_1, R_2 - T - b_1)$ and if the next state is $(R_1 - T - c_1, R_2 - S - b_1)$ the cost for going from one state to the other is the cost associated with moving the Robot 1 from configuration a_1 to configuration c_1.

Once the tree has been defined, we can apply any search algorithm and also we can apply the branch and bound algorithm to reduce the search space.

7 Robot Architecture and Experiments

We are using a Pioneer mobile robot (See Figure 10) with an on-board PC 400 MHz processor. It is equipped with a Sony EVI-30 CCD moving camera for landmark identification. The robot is also equipped with a Sick laser range sensor. This sensor uses a time-of-flight technique to measure distances.

The software consists of several modules executing specialized functions and communicating using TCP/IP socket communications under a client/server protocol. The main modules in our robot architecture are: Frame server, sick laser server, line fitting module, model matching module, landmark identification server, motion planner, motion controller, and system coordinator.

We are currently developing and integrating the robot architecture necessary to perform our whole approach in a real robot. Up to now we have totally developed the frame server, motion planner, line fitting, sick laser server, model matching, motion controller and system coordinator modules. We are working on the landmark identification module.

A computer simulation of this planner has been done. The software is written in C++ and uses geometric functions available in the LEDA 4.2 library. The simulation shows that this approach produces good results for the model building task.

Figure 10: Indoor mobile robot and sensors

In our simulation landmarks are represented with dark disks, the robot with a light square and the road-map with lines. The robot is placed anywhere inside the map, and begins exploring. As the robot moves across the map it takes every visibility area from the positions selected by the utility function to construct the model incrementally. The road map is constructed at the same time. The final map is constituted by polygons (which represent walls or obstacles), landmarks, and a road-map, constituted by a graph. When the robot ends exploring an area, it is capable to go back since it remembers past unexplored areas. This *backtracking* is based on navigation across the backtracking graph.

Figure 11 shows how the utility function works: in Figure 11 the robot has to take a decision between going to a large free edge, which means seeing as much of the as-yet-unseen

(a) (b)

Figure 11: (a) Utility function selection of next view (b) Robot going to the landmark

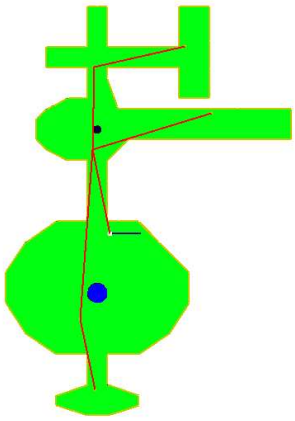


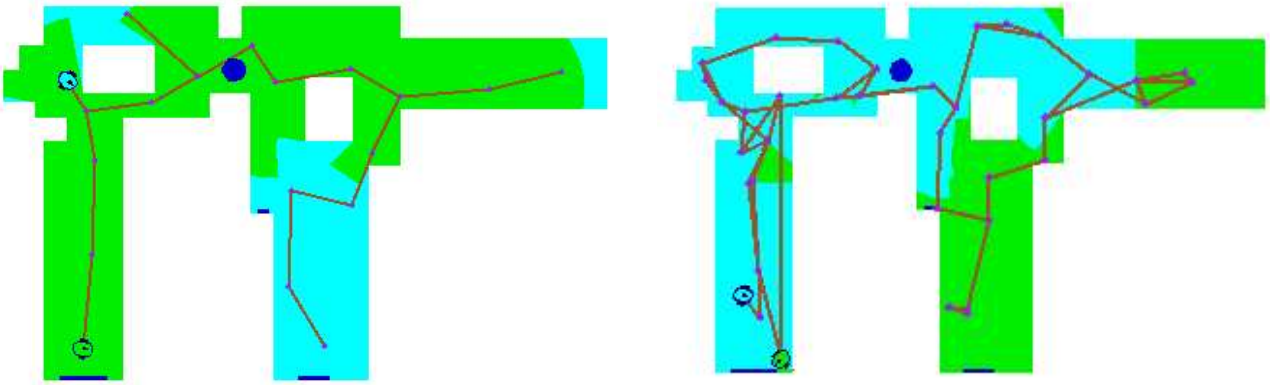
Figure 12: Exploration result using genetic algorithms to optimize utility function selection

environment as possible or going to a landmark to re-localize itself. In our simulation, the robot chose to improve its localization by going to the landmark (Figure 11(b)) and then go back and explore the unknown environment. In these figures the current visibility region is showed by a dotted line semi-circle.

Genetic algorithms have been also used as an optimization method in probabilistic based motion planning methods [2, 23]. An implementation of the utility function here proposed have been done using *genetic algorithms*. The genetic algorithm uses the Vasconcelos deterministic model for individuals crossing and the parameters like population size, crossing and mutation probabilities are self-adapting. Results are shown in Figure 12. The red lines indicates the robot path, the blue objects are landmarks. This implementation gives as result fewer sensing operations and rotations. However, this implementation takes considerable more computational running time (several hours).

Figure 13 shows multi-robot map building. The colors in the map are used to associate a part of the map to the robot that has explored it. It can be seen that the map is uniformly distributed among the robots in the environment. The landmarks are shown in the figure as a blue disk (a column) or blue segments on the walls which represent posters. In Figure 13(a) the environment is explored using non-limited range sensor and a 360 degree visibility capability. It can be seen that with such conditions, the number of created milestones for sensing operations is smaller and the robot trajectory much simpler and shorter than in Figure 13(b) where a limited range sensor and a visibility of 180 degrees was chosen.

Figure 14 shows a simulation with 8 robots in environment composed of a central hall and several narrow corridors. It can be seen that every robot has explored a different corridor.



(a) (b)

Figure 13: Multi-robot map building: (a) case of an omni-directional and infinite range sensor (b) case of 180 degrees field of view and limited range

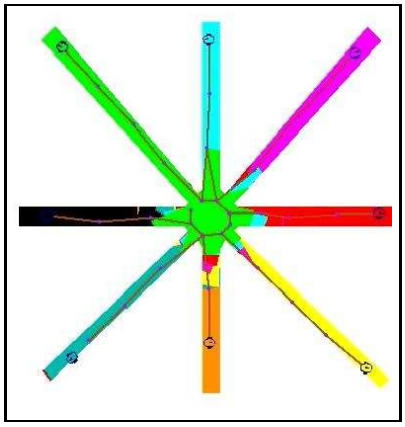


Figure 14: multi-robot map building

7.1 One single step ahead vs. several steps ahead optimization

Properties	**one robot/one single step**	**one robot/several steps**
path length	517.0367	637.2925
number of stops	51	58
number of sensing locations	20	18
total turn angle (degrees)	1406.735	879.578
Properties	**two robots/one single step**	**two robots/several steps**
path length	260.71756	461.5974
number of stops	27	57
number of sensing locations	15	20
total turn angle (degrees)	1323.924	1903.179

Table 2: One single step ahead vs. several steps ahead optimization

In this section, we compare a one single step optimization scheme vs. our partial greedy algorithm, capable of exploring several steps ahead. We consider two scenarios: One single robot and two robots exploration. The parameters used to make the comparison are path length, number of robot turns, number of robot stops and number of sensing locations. In all these simulations the robots have limited range sensor and an omnidirectional sensing.

The robots paths and the perimeter of the current visibility regions are shown in all figures. Figures 15 a), b) and c) show the evolution of an exploration simulation with one robot and an optimization schema with only one single step ahead. Conversely, figures 15 d), e) and f) show a simulation with one robot an an optimization schema considering several steps ahead. Similarly, figures 16 a), b) and c) and figures 16 d), e) and f) respectively show an experiment with two robots and an optimization schema with only one single step ahead and with several steps.

The robots paths are qualitatively more complex when more than one step are used in the optimization. Table 2 shows quantitative results in terms of path properties and number of sensing locations. In general, the results are better with a totally greedy exploration. That is, the path length is shorter, the total angle turned by the robot, the number of robot stops and the number of sensing location are smaller. In a several steps ahead optimization, the robots make a long term plan. That plan returns the ordering to visit all the current free edges. The same strategy is repeated until all free edges are explored. This planning process yields unnecessary motions.

As soon as new free edges appear (which have not been considered in the original plan), the robot will need to come back to locations near to other locations already visited. The interpretation of these results is that making long term plans with partial and dynamic information will result in a waste of resources.

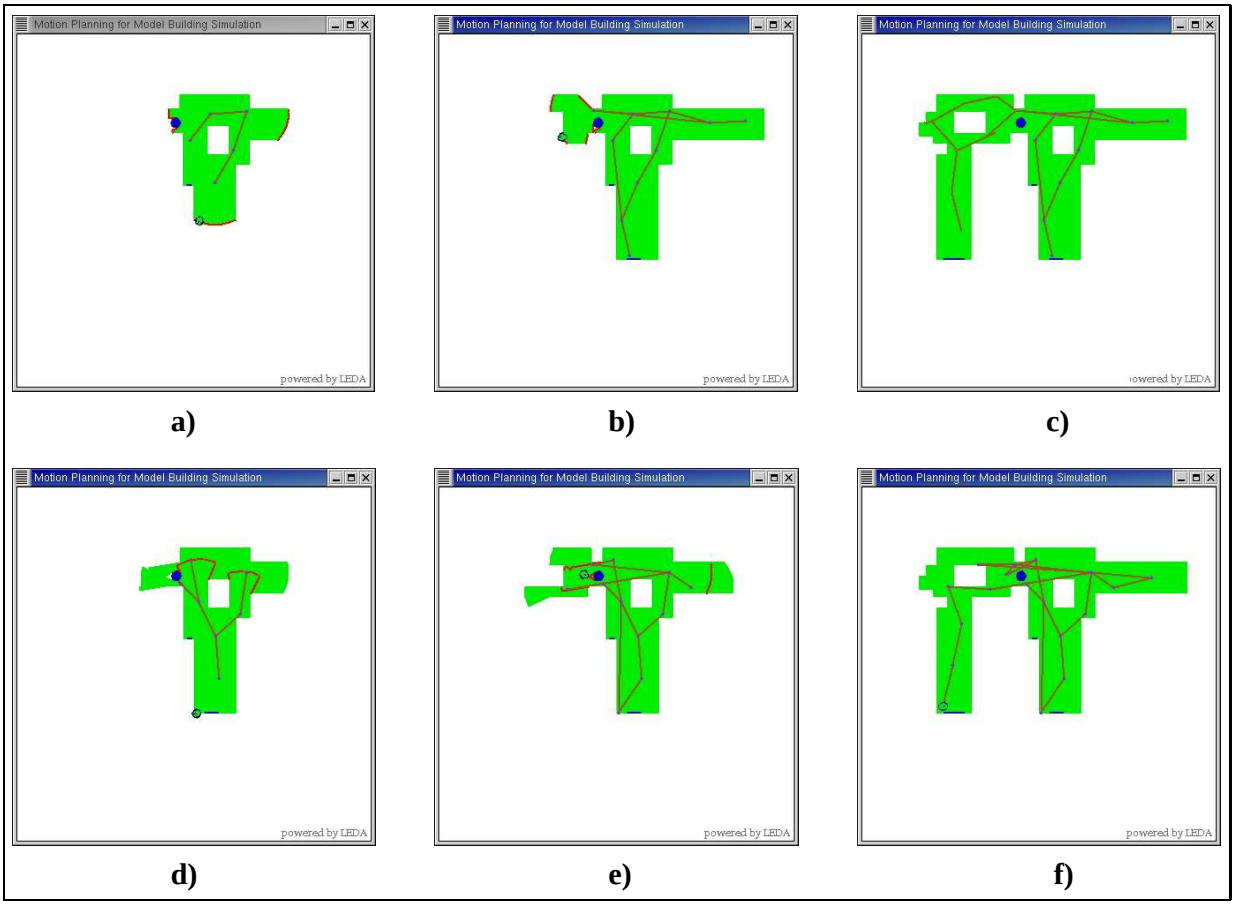


Figure 15: one robot exploration

7.2 Experiments with real robots

Figure 17 shows an experiment on the real robot. The robot has chosen to go to the free edge in the front left because it represents the next best view according to the Equation (1). Note that the robot has not chosen to turn around to see the free edge behind it, because this configuration has a low value according to the Equation (1). Actually, this configuration is bad because it is not possible to perform model matching and therefore deal with robot localization. Figure 18 shows the environment map at time t and the robot location at time t and $t+1$. Figure 18 shows the environment map and robot location at time $t+2$. In both cases, the visibility robot region is shown in yellow, the free edges in red and the full edges in blue. Figure 19 shows the laser data at time $t+2$ and Figure 20 shows the robot going to the next best view.

Currently our global architecture is not complete yet. For the experiments performed on the robot, we are using only data obtained from the laser. The camera is not used.

Figure 21(a) shows a picture of a more complex map building experiment. Figure 21(b) shows the map built at time t, the robots visibility region is shown in yellow, the free edges

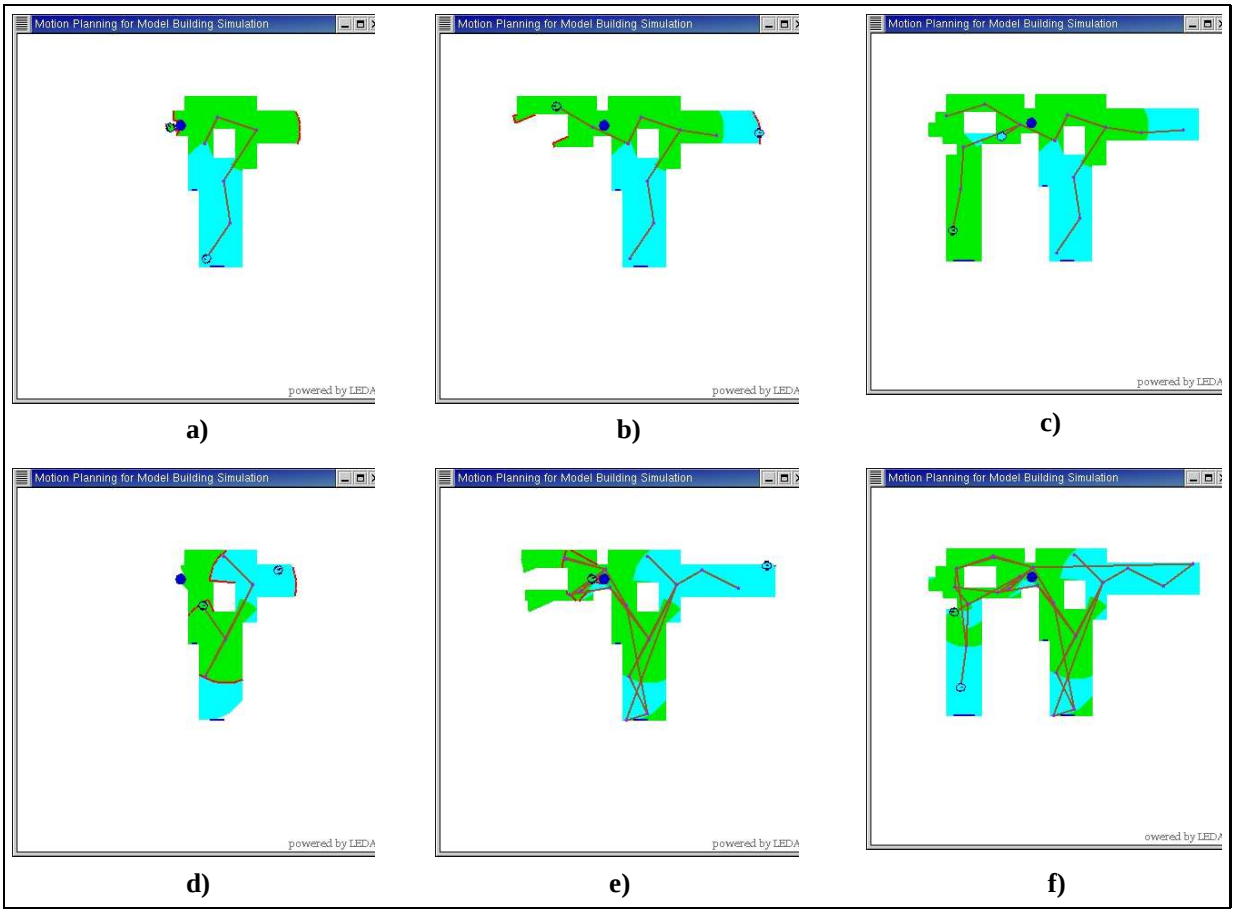


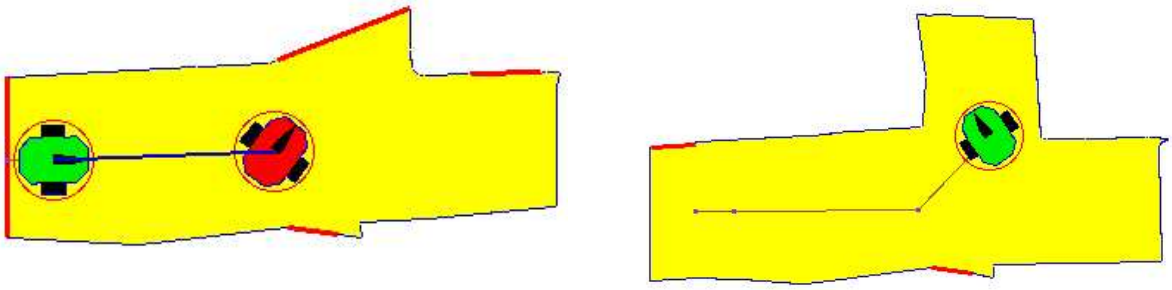
Figure 16: two robots exploration

in red, and the full edges in blue. In this figure it is also shown the robots location a time t (blue disk) and the next position where the robot has to move at time $t+1$ (red disk). This location has been computed using Equation (1), but taking into account only the parameters related to the laser data.

Figure 22(a) shows the laser data at time t and $t+1$ without registration. Laser data obtained at time t are in blue and those obtained at time $t+1$ in red. Figure 22(b) shows the map matching obtained by using Equation (4). The transformation (rotation and translation) related to the matching is used to improve robot localization. Figure 23(a) shows the polygonal model of the environment and the last two robot locations. The road-map used by the robot is shown in the figure by using brown dotted lines. When the robot gets back to a previous location (traveling through the backtracking graph) a localization procedure is executed at each graph node using the model matching result. Figure 23(b) shows the laser final map.

Our experiments were conducted in environments with clear polygonal shapes. This is still of interest since we are not proposing new algorithms to deal with complex noisy data.

Figure 17: Experiment on a real robot

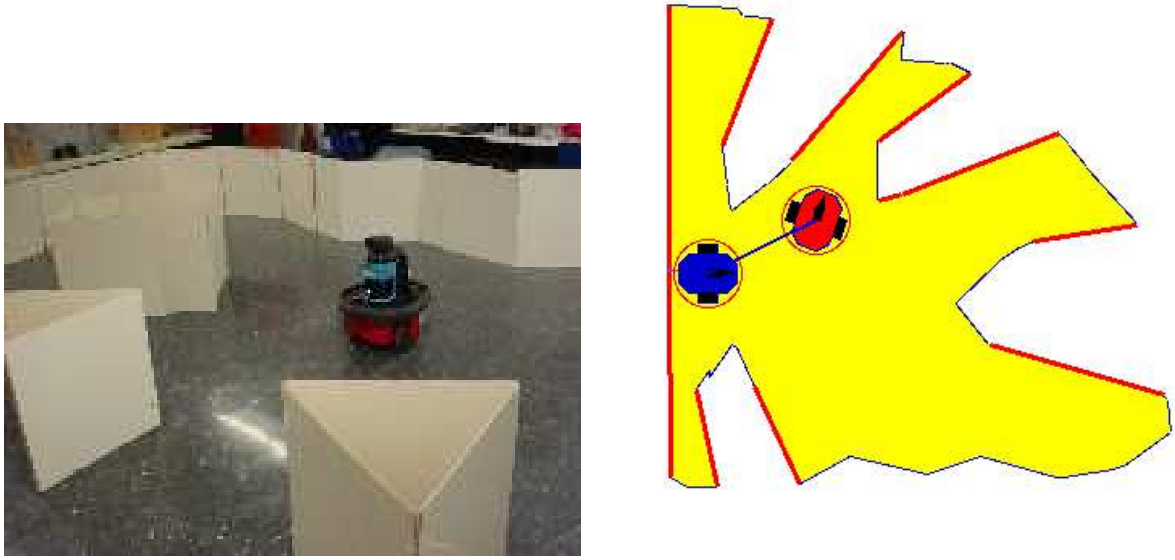


(a) (b)

Figure 18: (a) Environment map at time t, (b) Environment map at time $t + 2$

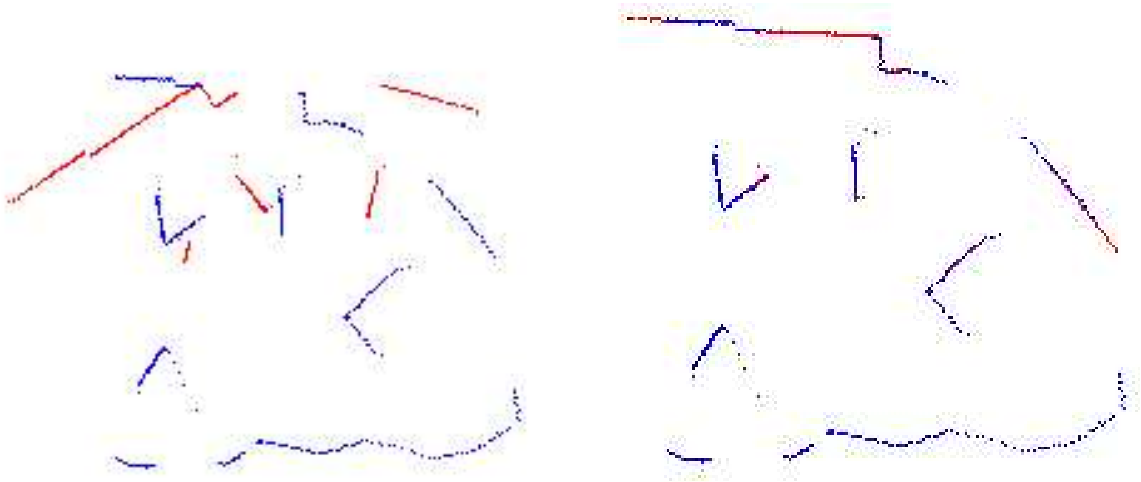
Figure 19: Laser data

Figure 20: Robot going to the next best view



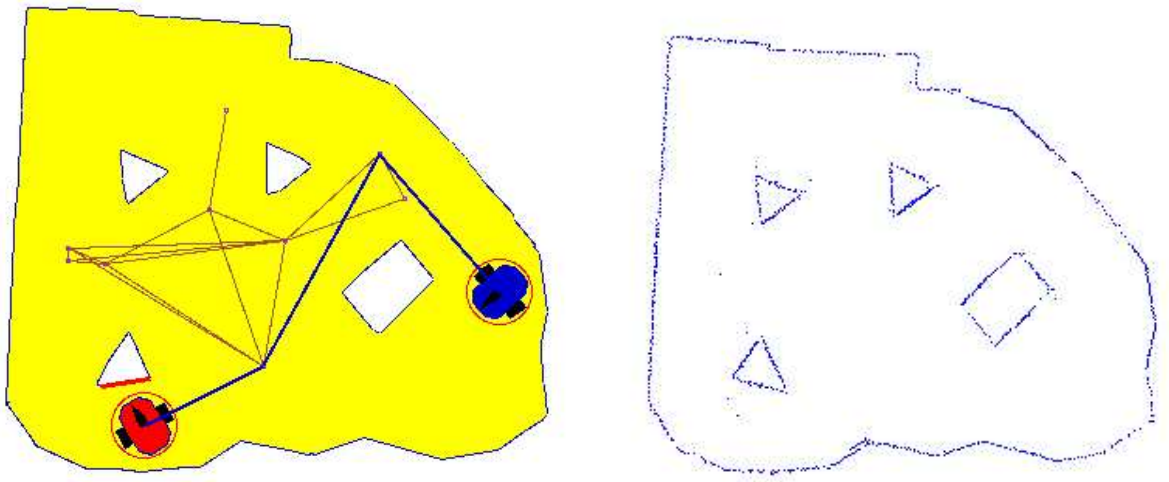
(a) (b)

Figure 21: (a) Experiment with a mobile robot, (b) The robot is going to explore a free edge



(a) (b)

Figure 22: (a) Laser data, (b) Model matching



(a) (b)

Figure 23: (a) The multi-representational map, (b) The final model

We have focused on the problem of planning optimal exploration strategies for SLAM. We believe that our experiments show that our proposed algorithms go beyond unrealistic theoretical assumptions. More sophisticated SLAM techniques can deal with more complex data. For instance, for large and complicate noisy environments a global optimization data matching would produce more precise maps. In general, we believe that SLAM algorithms and algorithms devoted to output exploration strategies solve two distinct and complementary aspects of the model building problem. In this vain, we claim that our algorithm makes it easier for an feature-based slam algorithm to integrate the information collected during navigation into the most accurate possible map.

8 Comparing our Approach with other Works about Exploration and Mapping

In this section, we compare our approach with other works devoted to map building and exploration.

8.1 González-Baños et al's. Approach

The exploration strategy presented in [15] is based on the computation of the next best view and the use of randomized motion planning. Our approach follows this research line. In [15] the concept of the next best view is based mainly on two factors: Estimated area of unexplored environment and distance traveled for the robot to reach a new sensing location. Visibility type methods [28, 30] are used to estimated information gain.

There are three main improvements of our work over the one presented in [15]. First, we consider the case of multi-robot exploration

Second, we have taken into account uncertainty (cost) due to control errors. This is contrast with [15], where the cost of a next sensing location is based only on the distance traveled by the robot. Experimentally, we have found out that uncertainty due to control errors (at least for our robots) increases faster when the robot rotates than when it translates. Location errors also increase faster when the robot changes its velocity. Thus, minimizing the number of robot stops to arrive at an unexplored area amounts to minimizing the robot position error. Our utility function reflects these facts

Third, overlapping between a local perception (local map) and the current global map may be necessary (but not sufficient) to merge a local map with the global one. In [15] the amount of overlapping between a local map and the global one is measured using the size of the common perimeter between them. However, measuring overlapping as an estimator of the ability of merging maps may not be enough. For instance, in corridors bounded by parallel featureless walls, a matching procedure only corrects positioning errors in the direction perpendicular to the walls. Our approach will try to avoid sensing locations, where the environment has that unwanted property. Our utility function will prefer sensing location from where corners or landmarks are visible making easier for a registration procedure (in our case the Hausdorff distance) to align the maps.

8.2 Makarenko et al's Approach

The work of [22] has proposed concepts that are similar to ours. We underline that both approaches have been proposed independently. Even though similar, these approaches are also different in several senses. First, we use a feature based modeling, as opposed to a grid. In our approach obstacles are modeled as polygons, while in [22], the environment is represented with a grid. We used original laser data to align local maps with the global one. The best transformation according to our metric (the Hausdorff distance) is used to correct the robot position and merge the maps. But, at the end the stored map is composed of polylines obtained through line fitting of the original laser points. Thus, the memory required to store the map is drastically reduced in comparison with grid based maps. It is well known that when the size of the environment is large, grid based models become difficult to handle. Additionally, feature based map are more suitable for path planing and free space visualization. We believe that our proposed environment representation is more useful to several robotics tasks, specially visibility-based ones such as target tracking and target finding.

Second, in our approach the use of landmarks is proposed to better localize the robot and merge partial maps. We also use landmarks as a pivotal feature to align partial maps. For any given candidate robot destination, the more landmarks the robot is able to sense in it, the more rewards it will get. We distinguish between landmarks and features. Landmarks can be uniquely identified by the robot, while features are geometric entities without specific identities. In general a landmark should be more useful than a feature to solve the data association problem. By comparison, in [22] the use of landmarks is not integrated to evaluate the utility of a next sensing location.

Third, our utility function is much better at balancing opposite factors. This is because it has multiplicative form. The utility function of [22], however, is a summation. Thus in our approach a robot configuration with a very low value on at least one of factors will be discarded regardless of having a very good value on the others.

Fourth, the approach presented in [22] does not consider muti-robot exploration.

8.3 Simmons et al's Approach

Similar to the work presented in [31] our approach also defines a utility value over candidate sensing locations. For the information gain we take into account two factors: i) the size of the frontier between explored and unexplored environment and ii) the distance between the obstacles and the robot. By comparison, in [31] the information gain is based on a probabilistic estimation of the frontier size, but does not consider the distance between the robot and the obstacles. This is a limitation as many sensors become blind when the objects are very near to them. Thus, in this respect, our approach surpasses Simmons et al's since it captures the information gain better. Same as [13,15], in [31] the cost associated to a sensing location is defined as the distance that the robot would travel to reach that location. As argued before, we believe that our utility function provides a more realistic estimation of this cost. In [31] a central executive tries to maximize the total utility minimizing the overlapping

between the areas that will be explored. In our approach the robot team coordination is similar. Each robot is assigned to one unexplored area, but only one robot moves at each time to avoid possible collision among moving robots.

8.4 Newman et al's Approach

The approach presented in [27] has some similarities with ours. In both works visibility computation is used to estimate the pertinence of a next robot sensing location and both approaches use feature-based maps.

In our approach sensor parameters (range and field of view) are input to the planner. The planner returns a path that depends on those parameters. Thus, a plan for a robot with stronger sensor capabilities will result in a smaller number of milestones for sensing operations and shorter trajectories than a plan for a robot with weaker sensor capabilities. This adaptation capability is not presented in [27]. Furthermore, Newman et al do not consider neither the distance nor the control errors as a cost to be integrated to the utility of a sensing location. Our work does take into account that cost.

In contrast, in [27], it is assumed that the location of a feature is uncertain and represented by a set of probability distribution functions. Associating uncertainty with feature locations seems to better capture real world situations. This is not considered in our approach and has been left as part of our future research.

8.5 Feder et al's Approach

The work presented in [11] proposes a metric for adaptive sensing that is defined in terms of Fisher information. That approach does not consider neither path planning nor obstacle avoidance. In our work a road map is built as the exploration progress. When the robot ends exploring an area, it is capable to go back since it remembers where the unexplored areas are. This *backtracking* is based on navigation across the road-map. We use the robot visibility polygon reduced by the robot radius $\mathcal{F}(q_k)$ to avoid robot collisions. $\mathcal{F}(q_k)$ is a safe region for navigation that is visible from configuration q_k.

9 Conclusions and Future research

In this paper, we presented a motion planning approach for building a map of the environment. Our motion planning algorithms are all based on sampling.

From the path planning point of view, the originality of our work comes from the fact that the robot goal has to be determined at every single iteration of the algorithm. Unlike classical motion planning techniques, ours does not assume to know the exact robot position.

A planner that selects the next robot position is proposed. It works by maximizing a novel utility function. This function was especially designed to combine geometric information with an intensive usage of the results obtained from perceptual algorithms. The crux of our method is a sampling-based motion planner algorithm that, given a partial map of the

environment, selects where to move the robot next. We balance the desire to see as much of the as-yet-unseen environment as possible, while having enough overlap and landmark information with the scanned part of the building to guarantee good registration and robot localization. The final result of the exploration is a multi-representational map constituted by polygons, landmarks and a road-map.

Our approach improves existing methods in that the robot plans motions in such a way that its uncertainty localization is minimized. At the same time, the motion strategy takes into account that the robot must discover unexplored environment regions minimizing energy consumption. The proposed robot motion strategy generates a fast and reliable map building.

In the SLAM problem the main goal is to integrate the information collected during navigation into the most accurate possible map. In our work we want to provide a robot path through sensing locations. These location have been chosen to provide both the best possible sensor inputs and the minimal cost (both given maximal utility) to reach them in terms of energy and induced uncertainty.

Our algorithm does take into account the chosen map representation and sensor capabilities. Thus, our motion strategy will prefer sensing location with large overlap between the partial and the global maps and from where corners or landmarks are visible making easier for a registration procedure to align the local map with the global one.

The quality and success of the generated paths depend significantly on the sensing robot capabilities. Studying the plan's dependency on the high level parameters describing the sensors (e.g., max. distance sensed, field of view) is an important part of our work.

Additionally, the uncertainty in the robot location will depend on the controls applied to the robot. Some path properties such as: path length, number of robot turns and number of robot stops will directly influence the magnitude of uncertainty in the robot location. Our algorithm will chose the robot path that minimizes unwanted types of controls.

We have proposed an efficient algorithm driven by our utility function. This algorithm is able to explore several steps ahead without incurring too high a computational cost. We have compared that exploration strategy with a totally greedy algorithm that optimizes a cost function with a one-step-look ahead. In general, the results are better with a totally greedy exploration. The interpretation of these results is that making long term plans with partial and dynamic information may often result in a waste of resources.

In general our algorithms will output a set of sensing location and robot path to reach them that makes it easier for an feature-based slam algorithm to integrate the information collected during navigation into the most accurate possible map.

In summary, our work fills in some gaps between exact geometrical approaches and approaches that consider uncertainty by taking advantage of perceptual information - from data registration up to scene understanding - to reduce the robot position uncertainty.

Multi-robot coordination algorithms were presented as well. The proposed algorithms have been implemented and experiments on real robots are included. The quality of the plans mainly depends on the number of generated samples and the robot sensing capabilities.

While significant, our work leaves room for improvements. Further work should consider

a more sophisticated algorithm for coordinating a team of robots to explore the environment, specially one where more than one robot moves simultaneously. Further work should also consider that a feature location is uncertain and should be represented with a set of probability distribution functions, as in [27]. We want to complete our global robot architecture including a video camera to extract landmarks from visual data. We want also to make multi-robot map building experimentation. We are planning to use our 2D model to help selecting "good" locations where to perform 3D sensing operations to construct a 3D model of the environment.

Acknowledgments

The authors thank Jean Claude Latombe for his contribution to the ideas presented in this paper. The authors also want to thank Héctor González Baños for his suggestions on the implementation of our algorithms, and Claudia Esteves for his help in the development of the robotic system. This research was partially funded by CONACyT project J34670-A and by the ITESM Campus Ciudad de México, México.

References

[1] A. Aho, J. Hopcropft, and D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983.

[2] J. Ahuactzin, E. Mazer, and P. Bessire. Using genetic algorithms for robot motion planning. In *In proc European Conference on Artificial Intelligence*, 1992.

[3] N. Ayache and O. Faugeras. Building registration and fusing noisy visual maps. *Journal of Robotics Research*, 7(6):45–65, 1988.

[4] H. Bulata and M. Devy. Incremental construction of landmark-based and topological model of indoor environments by a mobile robot. In *IEEE Int. Conf. on Robotics and Automation*, 1996.

[5] J. Castellanos, J. Montiel, J. Neira, and J. Tardós. The SPmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15(5):948–953, 1999.

[6] R. Chatila and J.-P. Laumond. Position referencing and consistent world modeling for mobile. In *IEEE Int. Conf. on Robotics and Automation*, 1985.

[7] H. Choset and J. Burdick. Sensor based motion planning: The hierarchical generalized voronoi diagram. In J. Laumond and M. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*. Springer Verlag, 1997.

[8] J.-L. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *IEEE Int. Conf. on Robotics and Automation*, 1989.

[9] G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localisation and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 2001.

[10] A. Elfes. Sonar-based real world mapping and navigation. *IEEE Transactions on Robotics and Automation*, 3(3):249–264, 1987.

[11] H. Feder, J. Leonard, and C. Smith. Adaptive mobile robot navigation and mapping. *The International Journal of Robotics Research*, 18:650–668, 1999.

[12] B. Gerkey, R. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distibuted sensor systems. In *Int. Conf. in Advanced Robotics*, 2003.

[13] H. Gonzalez, E. Mao, J.-C. Latombe, T. Murali, and A. Efrat. Planning robot motion strategies for efficient model construction. In *Robotics Research - The 9th Int. Symp*, 1999.

[14] J. Gonzalez, A. Reina, and A. Ollero. Map building for a mobile robot equipped with a 2d laser rangefinder. In *IEEE Int. Conf. on Robotics and Automation*, 1994.

[15] H. H. González-Baños and J.-C. Latombe. Navigation strategies for exploring indoor environments. *IJRR*, 21(10/11):829–848, Oct–Nov 2002.

[16] J. Hayet, M. Devy, and F. Lerasle. Visual landmarks detection and recognition for mobile robot navigation. In *Int. Conf. on Computer Vision and Pattern Recognition*, 2003.

[17] S. Hutchinson. Exploiting visual constraints in robot motion planning. In *IEEE Int. Conf. on Robotics and Automation*, 1991.

[18] D. Huttenlocher, G. Klanderman, and J. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863, 1993.

[19] K. Konolige, C. Ortiz, R. Vincent, A. Agno, M. Eriksen, B. Limketkai, M. Lewis, E. Briesemeister, L. Ruspini, D. Fox, J. Ko, B. Steward, and L. Guibas. Centibots: Large scale robot teams. In *International Workshop on Multi-Robot Systems*, 2003.

[20] A. Lazanas and J.-C. Latombe. Landmark-based robot navigation. *Algorithmica*, 13:472–501, 1995.

[21] W. Lee, B. Kuipers, R. Froom, and D. Pierce. The semantic hierarchy in robot learning. In *IEEE Int. Conf. on Robotics and Automation*, 1993.

[22] A. Makarenko, B. Williams, F. Bourgault, and H. F. Durrant-Whyte. An experiment in integrated exploration. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2002.

[23] E. Mazer, J. Ahuactzin, and P. Bessire. The ariane's clew algorithm. *Journal of Robotics Research*, 9:295–316, 1998.

[24] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2002.

[25] R. Murrieta-Cid, C. Parra, and M. Devy. Visual navigation in natural environments: From range and color data to a landmark-based model. *Journal Autonomous Robots*, 13(2):143–168, 2002.

[26] R. Murrieta-Cid, C. Parra, M. Devy, B. Tovar, and C. Esteves. Building multi-level models: From landscapes to landmarks. In *IEEE Int. Conf. on Robotics and Automation*, 2002.

[27] P. Newman, M. Bosse, and J. Leonard. Autonomous feature-based exploration. In *IEEE Int. Conf. on Robotics and Automation*, 2003.

[28] J. O'Rourke. *Visibility*. CRC Press, Inc., 1997.

[29] C. Parra, R. Murrieta-Cid, M. Devy, and M. Briot. 3-d modelling and robot localization from visual and range data in natural scenes. In H. Christensen, editor, *First Int. Conf on Vision Systems*. Springer Verlag, 1999.

[30] T. Shermer. Recent results in art galleries. *Proc. of the IEEE*, 80(9), 1992.

[31] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes. Coordination for multi-robot exploration and mapping. In *AAAI National Conference on Artificial Intelligence*, 2000.

[32] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*. Springer Verlag, 1990.

[33] S. Teller. Automated urban model acquisition: Project rationale and status. In *DARPA Image Understanding Workshop*, 1998.

[34] S. Thrun, W. Burgard, and D. Fox. Probabilistic mapping of an environment by a mobile robot. In *IEEE Int. Conf. on Robotics and Automation*, 1998.

[35] B. Tovar, R. Murrieta-Cid, and C. Esteves. Robot motion planning for model building under perception constraints. In *International Symposium on Intelligent Robotic Systems*, 2001.

[36] B. Tovar, R. Murrieta-Cid, and C. Esteves. Robot motion planning for map building. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2002.