Robot Motion Planning for Model Building under Perception Constrains

Benjamin Tovar-López, Rafael Murrieta-Cid, Claudia Esteves-Jaramillo *

ITESM Campus Ciudad de México
Calle del Puente 222, Tlalpan 14380
Ciudad de México
México

Abstract. The goal of this work is to develop techniques that allow one or more robotic observers to operate with full or partial autonomy while accomplishing the task of model building. The planning algorithm operates using certain simple but flexible models of the observer sensor and actuator abilities. We provide techniques that allow us to implement these sensor models on top of the capabilities of the actual (and off-the-shelf) sensors we have. It is worth keeping the following points in mind regarding our goals:

- even with completely idealized sensing and mobility capabilities, the algorithmic task of model building is quite challenging.
- computational techniques can be used to approximate and implement these idealized sensors on top of actual sensors.
- the quality and success of the generated plans depend significantly on the observer capabilities; study of this dependency terms of high-level parameters describing the sensors (e.g., max. distance sensed, viewing frustum) is part of this work.

One characteristic concern of this study is the need to satisfy perception constraints while planning motions. We focus on the fundamental motion planning problem considering information provided by logic sensors. Some of the questions that this work tries to answer are: Which locations must be visited by a robot to efficiently map a building? How must a robot move to explore an environment? To answer these questions we propose randomized motion planning techniques which take into account both geometrical and image analysis computation.

1 Introduction

In this section we analyze the generation of motion strategies for building a map of an indoor environment using a mobile robot with range and video sensors. The main problem to solve is to choose where the robot should move to get the next perception . We describe a *planner* that selects the next position from a set generated with uniform probability. The selection of the next position is based on the maximization of an *utility function* inside a randomized motion planning frame. The evaluation of this function uses the concept of *robot information space*. Analyzing the space information, the utility function takes into account this *information space* and selects the next position using the following criteria:

- Trajectories where the robot may identify objects (landmarks) that can be used to navigate are preferred
- The robot's localization uncertainty should be minimized
- The number of sensing operations should be minimized
- Energy consumption should be minimized by exploring a minimum distance trajectory
- The sensor capabilities (max distance sensed, viewing frustum) should be taken into account to compute motion stategies
- Motions should be performed in such a way that the robot perceives non-explored regions

The implemented planner combines geometric calculations with an intensive use of information obtained by the perception algorithms, for instance scene recognition. The final result of the exploration is a multi-representational map consisting of polygons and landmarks, and including a *road-map* constructed from the trajectory followed by the robot.

* [betovar, rmurriet, cesteves]@campus.ccm.itesm.mx

Motivation Representing and understanding the environment from sensor reading is a fundamental task for robot navigation and exploration of an unknown environment. We believe that the approach proposed here is clearly useful for these task. Besides, we claim that this environment representation can also be a crucial step toward building robots that can automatically achieve visual tasks, such as finding and tracking a target. Unlike the simpler "Go from A to B" task, these tasks require reasoning about both motion and visibility obstructions. The representation proposed is a complete and reliable map for these tasks because it gives a way to easily compute visibility and robot localization which are a basic input to compute motion strategies based on sensor information.

- Finding an evasive target requires one or several robots to sweep the environment so that the targets does not eventually sneak into an area that has already been explored. The planner can use the representation proposed here to compute a robot motion such that, for any point p along this path, the section of the environment that has alredy been explored before reaching p is fully separated from the unexplored one by the region from the point p [13].
- In the target tracking task, the robot must visually track a target that may try to escape its field of view, for instance, by hiding behind an obstacle. The online planner can use the representation proposed here to decide how the robot should move [18, 11]. At each step, it computes the visibility region of the robot at several sample locations picked in a neighborhood of its current location, identifies the one that is most likely to contain the target, and commands the robot to move there.

2 Previous Work

Automatic model building is an important problem in mobile robotics [17, 6, 20, 24]. Several types of models have been proposed, e.g. topological maps [8], occupancy grids [6] which uses a 2D array to represent the environment. In this method, each cell is valued as free space, occupied space or unknown space. Grid-based building algorithms prove to be a very simple and quite useful model for obstacle avoidance and planning purposes [6, 14]. However, when the size of the environment is big it becomes hard to handle this type of models. 3-D models [22] or feature-based maps [2] and polygonal representation [17] have also been proposed. Feature-based models is another way to represent the environment by using geometrical primitives. The most popular geometrical primitive is the segment, which can be extracted from ultrasonic data [4], laser rangefinder data [10, 16], or vision data [1, 19].

Most of these research has focused on developing techniques to extract relevant from raw data and to integrate the collected data into a single model; robot motion strategy is however not developed. In this work, we deal mainly with this problem. On [9] a map building motion planning strategy is presented. This work has shown that it is possible to find a function that reflects intuitively how the robot must explore the space. In a simple scheme, the evaluation function must assign a greater value to the position that best fits the compromise between possible elimination of unexplored space and energy consumption. One way to measure the size of the unexplored space is to measure the size of the free edge near to it. A free edge is defined as the border between regions of explored and unexplored space. Energy has been measured by the distance that the robot must travel.

This strategy is based on the computation of the next-best view and the use of randomized motion planning; the concept of the next-best view is, however, almost purely based on geometrical information, such as the visibility region from a robot position [15]. Uncertainty in robot localization and scene understanding are not taken into account. Our work tries to fill these gaps.

3 Our approach

The problem to solve in map building is to determine a *good* motion strategy that allows the exploration of the whole environment and to represent this new knowledge in such a way that not only the actual robot may deal with, but also the other mobile robots can work.

The definition of a *good* motion strategy depends on the desirable characteristics, such as minimum time, energy, uncertainty, representation complexity, etc. How to fit these requirements depends on the *explorer robot*, like sensors type and range, mechanical restrictions, etc. Until now most of the model building planners are based on pure geometrical calculations. These planners generate simple and manageable representations, but contain limited environmental information.

Our work extends previous approaches adding perceptual characteristics and the robot's position uncertainty [16]. The proposed criteria prefers those positions where the robot may recognize a *landmark* in order to perform landmark based navigation [12, 3]. A landmark is defined as a recognizable object in space. A simple definition of *place* is created where the robot's location uncertainty is minimized. The *place* corresponds to the influence area of a landmark (or set of landmarks), i.e., the area from which the robot may see these entities.

Regarding perceptual features, the evaluation function that selects the next-best view from the randomized generated positions, should prefer those positions where a perfectly recognizable object is found [23]. In this approach, it is proposed that the robot have online access to an object database and the identification is made by using some classification technique, for instance Bayesian rule. If the object probability of belonging to any class is low, then the object must be taken only as a visibility obstacle.

The results of perception algorithms are known until the robot is at the next position, given that perceptual information can be obtained only if the robot is able to perceive the scene. For this reason a two step evaluation function is proposed. The utility function T that we propose is given by the following expression:

$$T = (\mathrm{e}^{(l_v - s - s_v)}) \left(\frac{\mathrm{e}^{-|\theta|}}{\Gamma + \sqrt{s} + 1} \right) \left(\frac{1}{n} \sum_{j=1}^{n} p_j + \mathrm{N}_e \right) f_{min}(d_l) f_{min}(d_b)$$

Where:

s	Distance from the robot to the next possible position
s_v	Distance from the next possible position to the closest free edge
l_v	Length of the closest free edge
θ	Needed orientation to reach the next robot's configuration
Γ	Accumulated uncertainty
p_j	Object identification probability
n	Number of landmarks inside a visibility region
N_e	Number of corners and end-points inside a visibility region
f_{min}	Function of the minimum distance from an object or full edge
d_l	Minimum distance from a full edge
d_b	Minimum distance form an object

For f_{min} we choose a function like the shown in figure 1. Before a distance threshold t the function takes a low value, discriminating those positions near the objects. After the threshold the function takes the value of 1, to let the value of T reside on the other parameters.

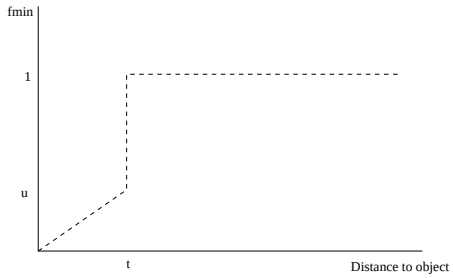

Fig. 1. f_{min} function

The first step for the utility function will evaluate the possible size of unexplored space, energy needed, distance to the nearest free edge, distance to nearest known walls and objects. All these parameters are inferred from the current robot position (by computing the visibility region [15]) which means that the robot does not have to go to every new possible position. The length of the free edges is used to estimated the size of the unexplored space. A very simple model of the uncertainty is used, whose growth is proportional to the square root of the distance to travel. In this uncertainty model we also include a term where the rotation cost is calculated. It is preferred that the robot travels straight than rotating.

In the second step, the robot *actually* moves to the m best evaluated positions and selects the best one taking into account perceptual information. In a very simple way, it is as if the robot would take a glimpse from the threshold of several doors, and explore the room that it likes the best. The robot will stop exploring when there are no free edges to be visited left.

The perceptual information is composed by the identification probability of the objects perceived by the robot and some particular features such as corners, that allows the matching between the areas already explored. The utility function is constructed in such a way that it integrates all these features. The next possible position that maximizes the utility function is kept. The function will prefer positions combining proximity to the robot, proximity to a free edge, small uncertainty value of robot position, high object identification probability and ability to see features like corners. Positions near walls and objects will be discarded because many sensors become blind when the objects are very near.

The function is composed by terms representing each one of these measures mentioned above. It is proposed that the utility function has a multiplicative form. A position with a very low value on at least one of these measures will be discarded even though it may 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 new area to the explored space.

The p_j is the identification probability for a landmark. A landmark is defined as a remarkable object. The landmark should have some properties that distinguish them from other objects, e.g.:

- **Discrimination**. A landmark should be easy to differentiate form other surrounding objects.
- **Accuracy**. A landmark must be useful to reduce the uncertainty of the robot position.

Landmarks in indoors environments are often structures such as corners, doors, columns, posters, etc. To compute the distance between a landmark and the robot by using a single camera, we suppose that the landmark size is known. Our landmarks are posters, doors, and columns to suppose that for a given environment the size of these objects is known a priori is not unrealistic. An object is labeled as a landmark if and only if its object identification probability is greater than a given threshold and has a remarkable shape (different from surrounding objects). The object shape can be obtained by using a segmentation algorithm as that presented in [19]. Object identification is obtained by comparing vector features with a database composed of different classes, issued from a learning process. The database is a function of the type of environment.

The object identification probability is calculated using Bayesian rule, which as defined as:

$$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{\sum_{i=1}^{n} P(X \mid C_i)P(C_i)}$$

Where $P(C_i)$ is the *a priori* probability that an object belongs to the class (C_i). $P(X \mid C_i)$ is the class conditional probability that the object is X, given that it belongs to class (C_i). $P(C_i \mid X)$ is the *a posteriori* conditional probability that the object's class membership is C_i, given that the object is X.

We have assumed equal *a priori* probabilities. In this case the computation of the *a posteriori* probability $P(C_i \mid X)$ can be simplified and its value depends solely on $P(X \mid C_i)$.

The value of $P(X \mid C_i)$ is estimated by using the k-nearest neighbor method. A sample X will be assigned to the class C_i whose k_{th} nearest neighbor to X is closest to X than to any other training class.

3.1 Line-fitting

Since we have a laser range finder as our sensor, it is necessary to recover the lines that forms the actual visibility region from the points that the laser gives.

We generate polylines with the laser data obtained as an ordered list (by angle) of polar coordinates (r, θ) where obstacles may be 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 radio between two consecutive points is similar. Then we take advantage of the error-free coordinate and apply the transformations $u = \cos(\theta)/\sin(\theta), v = 1/r\sin(\theta)$ as in [9]. We find fit lines by applying to each cluster a divide-and-conquer technique based on minimal squares and calculate the fit lines vertices as new (u, v) pairs.

Although, the minimal squares technique has the advantage of removing noisy measurements, it is not efficient in the number of lines it generates. For this reason a second divide-and-conquer style algorithm is necessary. This time we convert the new 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, and by this, eliminating the unnecessary ones. A cluster with a stand alone point or with very few points should be considered as a small object [9], 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.

3.2 Model-matching

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 and corners** of the polylines previously computed. First, the corners are used as input of our metric and second the alignment is improved by using all the original data, in this way computational running time is saved.

Given two sets of points P and Q, the Hausdorff distance is defined as (see [7])

$$H(P,Q) = \max(h(P,Q), h(Q,P))$$

where

$$h(P,Q) = \max_{p \in P} \min_{q \in Q} \|p - q\| \tag{1}$$

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 $h(Q,P)$. Thus 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 rank order measure, which replaces the maximization operation with a rank operation. This measure (partial distance) is defined as:

$$h_k = K^{th}_{p \in P} \min_{q \in Q} \|p - q\| \tag{2}$$

Where $K^{th}_{p \in P} f(p)$ denotes the K^{-th} ranked value of $f(p)$ over the set P. That is, if we consider the points in P to be in sequence ordered by their values $f(p_1) \leq \ldots \leq f(p_n)$, the K^{-th} element in this sequence, $f(p_k)$, is the K^{-th} ranked value. For example, the n_{-th} ranked value is the maximum (the largest element in the sequence), and the $n/2{-}th$ ranked value is the median.

One interesting property of the Hausdorff distance and the "partial distance" is the asymmetry inherent 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. Where $P = M_t$ is the model and $Q = I_t$ is the model or region of the model given an t possible translation and rotation. The maximum of these two values defines the partial Hausdorff distance. Of course, 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.

4 Experimental Approach and Robot Architecture

We are using a Pioneer mobile robot 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:

- A frame server
- A sick laser server
- A line fitting module
- A model matching module
- A landmark identification server
- A motion planner
- A motion controller and system coordinator

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

The frame server module grabs the RGB images from the hardware and separates the RGB component of the image. The 3 resulting images are stored in shared memory and accessible by other observer system modules (i.e. landmark server).

To improve maximal range and maximal cone angle we are using a controllable pan, tilt and zoom camera. This camera is able to execute [-100, 100] deg. pan action, [-25, 25] deg. tilt action and active zoom ($f = 5.4mm$ to $64.8mm$). Our implementation at present only uses the pan action. We are currently incorporating tilt and zoom actions. The motion in the camera is computed by a dedicated controller rather than to be computed by the planner. This camera motion, however, is taken into account by the planner by considering the total field of view as the addition of the normal field of view determined by the camera parameters (40 deg.) plus the motion of the camera ([-100 100] deg.).

To localize the robot we use natural landmarks. Several works have dealt with the use of landmarks in robot navigation [21, 5, 12, 23, 16]. The landmark detection module that we are currently developing is mainly based on the work presented in [19, 24]. The idea behind this approach is to provide the positions of the landmark as an input map to the observer. Each landmark induces a landmark region from which the landmark is visible for the robot. The robot localizes itself by detecting landmarks and computing its relative position respecting the landmarks. Afterwards, it compares this reading with the odometric history and updates its position. It is important to mention that there is not localization respect to a global frame. The robot is instead localized relatively to the landmarks, we think this approach will give us a more reliable performance.

A Sick sensor server handles communications through the on-board PC. It allows the connecting clients to assume they read data from an ideal sensor and offers bath transmissions of multiple scans or request, choice among 2 speed modes: 1, 5 scans/sec, scan averaging using sensor electronics and operations in continuous mode. The sick laser is working as a polar range sensor measuring the distance between the robot's centroid and the objects in the environment along several rays regularly spaced in horizontal plane at height h above the floor. The sensor driver converts these measurements into a list of point coordinates representing a cross-section of the environment a height h in a coordinate system attached to the sensor. Based on these data our line fitting module is charged to computed polylines which are necessary for the map building process. Our line fitting technique is fast enough to be performed on line at any of the two speeds modes, it is included in the Sick sensor server capabilities, which reduces the amount of data transmitted to clients. The model matching module is responsible for aligning new data with previous model using robot's odometry for pre-alignment before calling the model alignment module. The technique used to perform the model alignment is described in more detail in section 3.2.

The global planning algorithm that computes an appropriate milestone location for the robot to go towards is described in section 3. At the same time, the robot architecture includes a motion controller for the robot whose function is to reach the goal provided by the planner.

In order to improve the performance of the whole system, we will execute the planner program in a computer off of the robot. This allow us to increase the execution speed of the programs by splitting the task among two processors. The motion controller, vision programs are running on-board the robot. In fact vision programs have to run on-board because sending images through the network is very time-consuming.

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. The results are shown from Figures 2 through 6. In these figures landmarks are represented with dark discs, the robot with a light square and the roadmap with lines. The robot is placed anywhere inside the map, and begins exploring. As the robot moves across the map it takes every

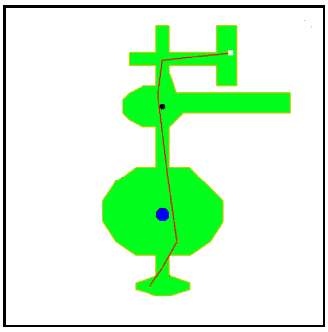

Fig. 2.

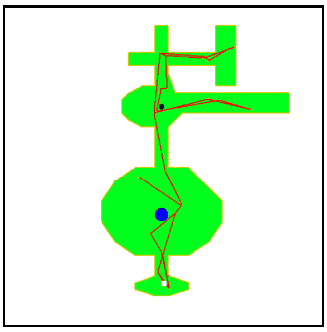

Fig. 3.

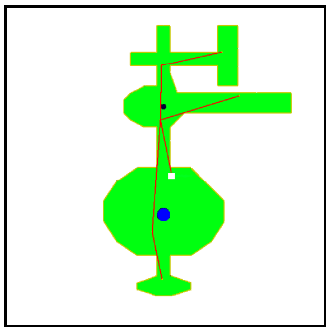

Fig. 4.

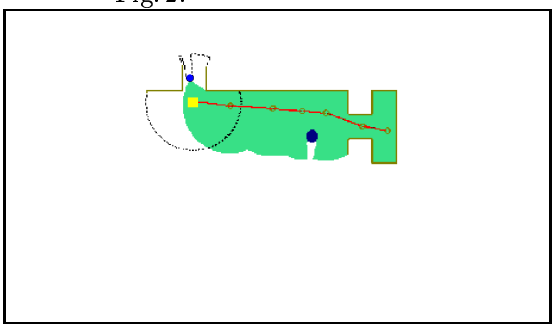

Fig. 5.

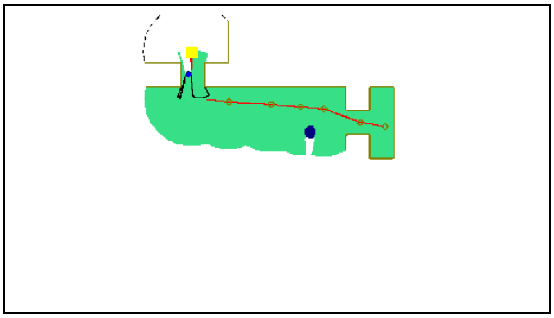

Fig. 6.

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 graph. Figure 2 shows how an environment is explored using ilimited range sensor and a 360 deg. 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 3 where a limited range sensor and a visibility of 180 deg. was chosen. Figure 4 shows an implementation of the metric here proposed with **genetic algorithms** which gives as result fewer sensing operations and rotations. However, this implementation takes more computational running time. The genetic algorithm uses the Vasconcelos deterministic model for individuals crossing and the parameters like population size, crossing and mutation probabilities are self-adapting. Figures 5 and 6 show how the metric works: in figure 5 the robot has to take a decision between going to a large free edge, which means seeing as much of the as-yet-unseen environment as possible or going to a landmark to relocalize itself. In our simulation, the robot chose to improve its localization by going to the landmark (figure 6) and then go back and explore the unknown environment. In these figures the current visibility region is showed by a dotted line semicircle. Figure 11 shows the simulation of our planner in an environment where there are posters (landmarks on the walls) and landmarks in the middle of the rooms, as well as obstacles (white square in the environment).

In figures 7 and 9 an example is shown of the data we receive from the laser. Figures 8 and 10 show the corresponding lines formed after the line-fitting process.

From the 2D planner an extension to 3D has been made considering constant height. We take information from the 2D planner such as the robot position and its orientation together with position of obstacles (probable landmarks) and free and full edges (walls). In the 3D case the cameras are placed in such a way that they allow us to get a better knowledge of the behavior of the proposed metric. Figure 12 shows four views of the 3D planner. The first view (top-left) is a top view from the already explored environment, the second (top-right) is a front view from the robot, in the third (bottom-left) the camara is placed where the robot is looking at and the last one (bottom-right) is a back view from the robot.

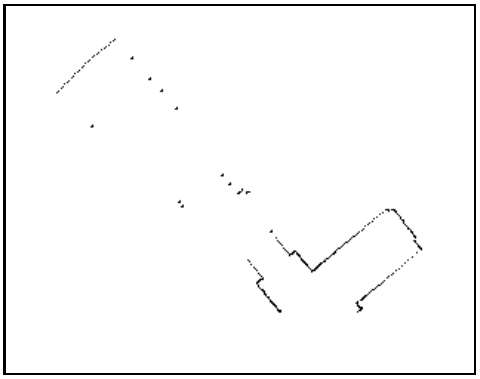

Fig. 7.

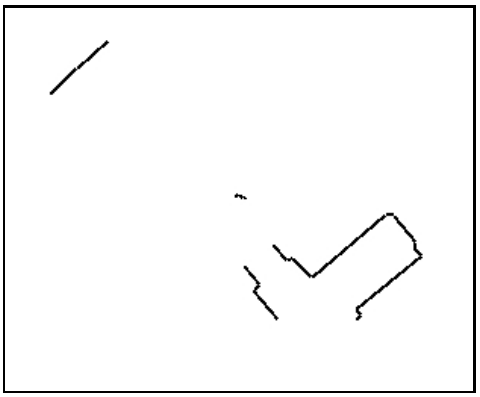

Fig. 8.

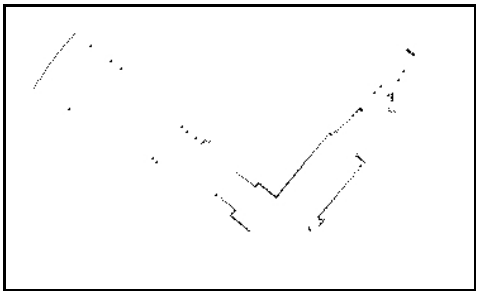

Fig. 9. Laser data

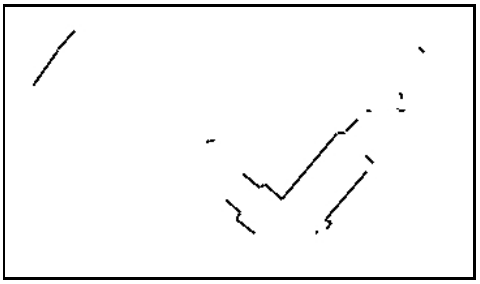

Fig. 10. Line-fitting process result

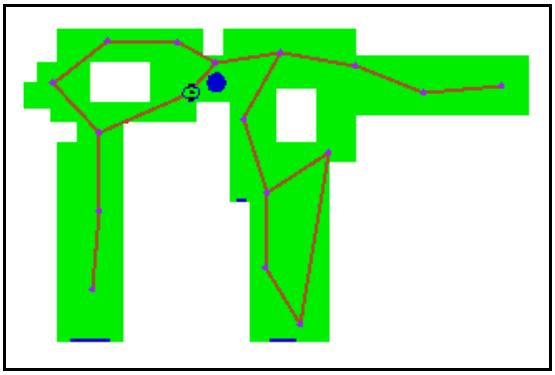

Fig. 11. Another example

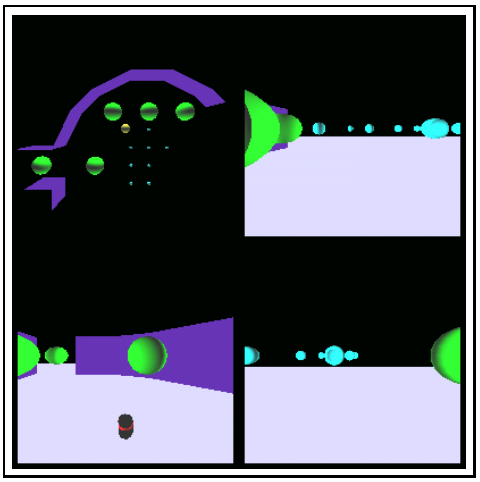

Fig. 12. Three Dimensional View

5 Conclusion and Future research

A planner that selects the next position of the robot based on maximizing the utility function is proposed. The evaluation of this function uses the concept of robot information space which combines geometrical information with an intensive usage of the results obtained from perceptual algorithms. The crux of our method is a randomized 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 at the same time 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.

As future research, first, we will complete our global architecture and perform intensive experimentation. A fundamental limitation of our system is the lack to take into account loops in the environment. Imagine, that the robot has performed a long trajectory such that it comes back to the same position and no landmark has been perceived, then the system have to be able to identified that this place has already been visited. We plan to keep track of the successive local models and transforms to make possible the periodic optimization of a global matching criteria. We also plan to extend this approach to handle multiple robots with relatively minor changes. We think that the representation of the environment proposed here is really useful to several robotics task specially visibility-based ones such as target tracking and target finding. However, a 3D model of the environment is a inter-medium goal to other tasks (object manipulation, assembling, etc). We belive that the model proposed can be used to select "good" locations where to perform 3D sensing operations. This 3D modeling is one possible future research.

Acknowledges: The authors thank Hector González Baños for his contribution to the development of the ideas presented in this paper. This work was funded by CONACyT project J34670-A and by the ITESM Campus Ciudad de México

References

1. N. Ayache and O.D. Faugeras. Building, registrating, and fusing noisy visual maps. *IEEE Journal of Robotics Research*, 7(6):45–65, 1988.
2. W.K. Lee B. Kuipers, R. Froom and D. Pierce. The semantic hierarchy in robot learning. In *IEEE Int. Conf. on Robotics and Automation*, 1993.
3. C. Becker, J. Salas K. Tokusei and J.C. Latombe. Reliable navigation using landmarks. In *IEEE Int. Conf. on Robotics and Automation*, 1995.
4. J.L. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *IEEE Int. Conf. on Robotics and Automation*, 1989.
5. D.J. Kriegmen E. Triendl and T.O. Binford. Stereo vision and navigation in buildings for mobile robots. *IEEE Trans. on Robotics and Automation*, 5(6):1722–1727, 1991.
6. A. Elfes. Sonar-based real world mapping and navigation. *IEEE Journal on Robotics and Automation*, 3(3):249–264, 1987.
7. D.P. Huttenlocher et. al. Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863, September 1993.
8. H. Choset and J. Burdick *Sensor based Motion Planning: The Hierarchical Generalized Voronoi Diagram.* In Algorithms for Robotic Motion and Manipulation, J.P. Laumond and M. Overmars (eds.), A K Peters, Wellesley (MA), 46-61, 1997.
9. H.H. Gonzalez, E. Mao, J.C Latombe and T.M. Murali. Planning robot motion strategies for efficient model construction. In *Robotics Research - The 9th Int. Symp*, 1999.
10. 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.
11. S. Lavalle, H. H. González-Banos, C. Becker, and J. C. Latombe. Motion strategies for maintaining visibility of a moving target. In *Motion Strategies for Maintaining Visibility of a Moving Target*, volume 1, pages 731–736, april 1997.
12. A. Lazanas and J.C. Latombe. Landmark-based robot navigation. *Algorithmica*, 13:472–501, 1995.
13. L.J Guibas, J.C Latombe, S.M LaValle, D. Lin and R. Motwani. Visibility-based pursuit-evasion in a polygonal environment. In *5th Workshop on Algorithms and Data Structures*, 1997.
14. L. Matthies and E. Elfes. Integration of sonar and stereo range data using a grid-based representation. In *IEEE Int. Conf. on Robotics and Automation*, 1988.
15. J. O'Rourke. *Visibility.* Handbook of Discrete and Computational Geometry, 467-479, J.E. Goodman and J. O'Rourke, 1997.
16. 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 *Accepted to International Conference on Vision Systems (ICVS)*, 1999.
17. R. Chatila and J.P. Laumond Position referencing and consistent world modeling for mobile robots. In *IEEE Int. Conf. on Robotics and Automation*, 1985.
18. R. Murrieta-Cid, B. Tovar and C. Esteves Robot motion planning under visibility constraints. In *Workshop Advances in Percepcion and Robotics (AAPR'2000)*, Guanajuato, México, October 2000.
19. R. Murrieta-Cid, M. Briot and N. Vandapel. Landmark identification and tracking in natural environment. In *International Conference on Intelligent Robots and Systems IROS'98*, Victoria, Canada, September 1998.
20. S. Thrun, W. Burgard and D. Fox. Probabilistic mapping of an environmet by a mobile robot. In *IEEE Int. Conf. on Robotics and Automation*, 1998.

21. S.Hutchinson. Exploiting visual constraints in robot motion planning. In *IEEE Int. Conf. on Robotics and Automation*, 1991.
22. S. Teller. Automated urban model acquisition: Project rationale and status. In *DARPA Image Understanding Workshop*, 1998.
23. T.S. Levitt, D.M. Chelberg, D.T. Lawton and P.C. Nelson. Qualitative navigation. In *DARPA Image Understanding Workshop*, 1987.
24. 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.

This article was processed using
the TEX macro package with SIRS2001 style