Planning Robot Motion Strategies for Efficient Model Construction

H. González-Baños, E. Mao, J.C. Latombe, T.M. Murali, and A. Efrat

Computer Science Department, Stanford University, Stanford, CA 94305, USA

Abstract

This paper considers the computation of motion strategies to efficiently build polygonal layouts of indoor environments using a mobile robot equipped with a range sensor. This problem requires repeatedly answering the following question while the model is being built: Where should the robot go to perform the next sensing operation? A next-best-view planner is proposed which selects the robot's next position that maximizes the expected amount of new space that will be visible to the sensor. The planner also takes into account matching requirements for reliable self-localization of the robot, as well as physical limitations of the sensor (range, incidence). The paper argues that polygonal layouts are a convenient intermediate model to perform other visual tasks.

1. Introduction

Automatic model construction is a core problem in mobile robotics [1, 2, 3]. After being introduced into an unknown environment, a robot, or a team of robots, must perform sensing operations at multiple locations and integrate the acquired data into a representation of the environment. Several types of models have been proposed, e.g., topological maps [4], polygonal layouts [1], occupancy grids [2], 3-D models [5], and feature-based maps [6]. Prior research has mainly focused on developing techniques to extract relevant data (e.g., edges, corners) from raw sensory data and to integrate collected data into a single, consistent model.

Here we consider the motion planning aspect of model construction, and we propose techniques to efficiently build a polygonal layout model of an indoor environment. We have tested these techniques both in simulation and with a mobile robot equipped with a laser range finder. The core of our system is an on-line next-best-view (NBV) planner that repeatedly answers the following question: *Where should the robot go to perform the next sensing operation?* To reduce the total duration of the exploration, the NBV planner computes the next sensing location by optimizing a utility criterion that is a function of both the amount of new information that is expected to become available at this location and the length of the motion to go there (given the partial model built so far). The planner also takes other important requirements introduced below into account.

To build accurate models, it is crucial that the robot localizes itself precisely relative to previous sensing locations (the so-called *simultaneous localization and map building* problem) [7, 8]. Dead-reckoning alone is often too imprecise and does not allow the merging of partial models built separately by multiple robots. Our system uses matching techniques to align and merge partial models into a single one. Such techniques require that the portion of the environment seen by the robot at each new sensing location has significant overlap with the portions of the environment seen at previous sensing locations [7].

Motion planning for model construction must also take into account the physical limitations of the sensor. The classical line-of-sight model – a point on an object is visible if the line segment connecting the sensor to this point does not intersect any object in the environment – is often too simplistic. Sensors have range limitations, both minimal and maximal, and object surfaces oriented at grazing angles relative to the line of sight may not be sensed properly. Our planner reads parameters specifying such limitations as inputs. In the same environment, but with different sensing parameters, it typically produces different motion strategies.

Several NBV techniques have been previously proposed [9, 10, 11, 12, 13, 14]. Several authors have considered the problem of building a 3-D model of a relatively small object using a precise range sensor moving around the object [10, 12, 13]. Typically, a model is first built by stitching together the data obtained from a few uniformly distributed viewpoints. The resulting model may include gaps caused by visual occlusions due to object's concavities, and an NBV technique is then used to select additional viewpoints that will provide the data needed to fill them. This approach does not seem efficient, if possible at all, for modeling unknown indoor environments. Like ours, the NBV algorithm proposed in [11] constructs a 2-D layout model of an indoor environment from data acquired by a sensor on a mobile robot. It assumes continuous sensing and computes at each step the direction of infinitesimal motion that produces the largest expected gradient of information. This technique, however, is too local and may yield very inefficient overall strategies. The need to take physical limitations of sensors into account was previously noted in [12, 13].

Section 2 gives additional motivation for our work. Section 3 presents the techniques used to construct a polygonal layout model independent of the NBV planner. Section 4 describes the algorithm of the NBV planner. Section 5 discusses the implementation of our system and its experimentation. Section 6 identifies areas requiring futher research.

2. Motivation

A 2-D layout is a limited model of an indoor environment, and constructing such a model is not the ultimate goal of our research. But it is nevertheless a crucial step toward building robots that we call *autonomous observers* [15, 16]. These are mobile robots equipped with visual sensors that can autonomously achieve visual tasks (in contrast to manipulation tasks), such as building a 3-D model of an environment or finding and tracking a target. Unlike the simpler "Go from A to B" task, these tasks require reasoning about *both* motion and visibility obstructions:

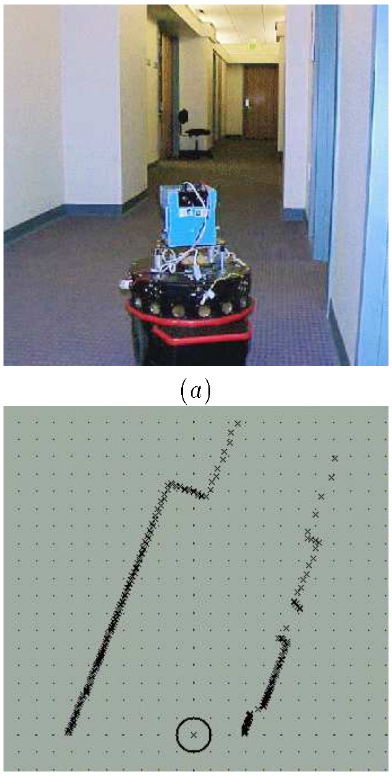
- In 3-D model construction, we use the 2-D layout to compute "good" locations where to perform 3-D sensing operations. These locations are selected so they collectively "see" the entire contour of the layout, or most of it (*art-gallery problem* [17]). Our algorithm samples the layout at random and computes the portion of contour visible from each sample. It uses a greedy set-cover technique to retain non-redundant samples [18].

- Finding an evasive target requires one or several robots to sweep the environment so that the target does not eventually sneak into an area that has already been explored. The planner uses the input layout to compute a robot's motion such that, for any point p along this path, the section of the environmemt that has already been explored before reaching p is fully separated from the unexplored one by the region visible from p [19].

- 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 uses a 2-D layout to decide how the robot should move. 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 [16, 20].

In each task, a frequently performed operation is to compute the visibility region of the robot at a given location. In a polygonal model with n edges, this is easily done in $O(n \log n)$ time by a classical line-sweep technique. Other representations, such as topological maps and occupancy grids, do not allow such computation or make it much more time consuming.

Planar layouts offer other advantages. They allow efficient collision-free path planning. They are more compact than most other models, thus facilitating wireless communication between onboard and offboard processors, and between cooperating robots, under limited bandwidth. They can be extended to include uncertainty information, e.g., in a way similar to tolerance representation in mechanical drafting. Finally, they are easy to understand by human users.



(a) (b)

Figure 1: Range sensing: (a) Scene, (b) Captured points

3. Construction of 2-D Layouts

Ideally, the 2-D layout of an indoor environment is the projection, into a horizontal plane, of the portions of objects in this environment that are obstacles to the robot's motion and/or may obstruct the field of view of its sensors. However, due to sensor limitations, the layout is often approximated as a cross-section through the environment at a given height. In the case of a multi-floor environment, the model may consist of several 2-D layouts connected by passageways.

We assume here that the robot is equipped with a polar range sensor measuring the distance between the robot's centerpoint and the objects in the environment along several rays regularly spaced in a 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 at height h in a coordinate system attached to the sensor. Figure 1 shows such points for a 180-dg field of view, with 0.5-dg spacing between every two consecutive rays. Our model of range sensing is specified by four parameters: field-of-view angle α, minimal range r, maximal range R, and maximal incidence angle η. The later is set to be the incidence angle above which range sensing is observed to be unreliable in practice.

The robot builds a polygonal layout by moving at successive sensing positions, which are chosen by the NBV planner described in Section 4. These positions could also be chosen by another software module or by a human user. We present below the steps performed at each sensing position q, independent of the NBV planner. We introduce an important novel concept which we call the robot's *safe region*.

Polyline generation Let L be the list of points acquired by the sensor at q. L is transformed into a collection p of polygonal lines called *polylines*. Our polyline extraction algorithm operates in two steps: (1) group data into clusters and (2) fit a polyline to each cluster. The goal of clustering is to group points that can be traced back to the same object surface. A sensor with infinite resolution would capture a curve $r(\theta)$, instead of a sequence of points. This curve would be discontinuous exactly where occlusions occur. For a real sensor, discontinuities are detected using thresholds selected according to the sensor's accuracy.

We fit the points in each cluster by a polyline so that no data point is farther away from the polyline than a given ϵ. We also try to minimize the number of vertices. The computation takes advantage of the fact the data points delivered by our polar sensor satisfy an ordering constraint along the noise-free θ-coordinate axis. We apply the transformation $u = \cos\theta/\sin\theta$, $v = 1/(r\sin\theta)$, and fit lines of the form $v = a+bu$ (which maps to $bx+ay = 1$ in cartesian (x, y)-space). Several well-known recursive algorithms exist to find polylines in (u, v)-space. We convert ϵ to the position-dependent error bound $e = \epsilon v\sqrt{(a^2+b^2)}$ in the (u, v)-space, in order to guarantee that, in the (x, y)-space, each data point is within ϵ from the computed polyline.

Figure 2 shows the three polylines generated from the data points of Figure $1(b)$. A more complicated example, in a cluttered office environment, is displayed in Figure 3. The area in light grey in (b) is the robot's visibility region under the classical line-of-sight model. As explained below, this region is not guaranteed to be entirely safe for robot motion.

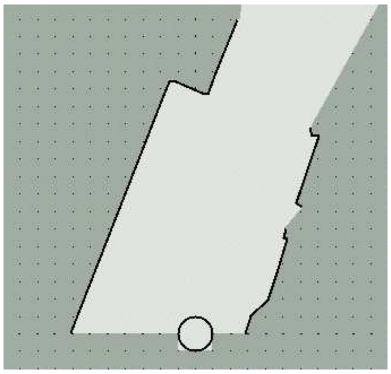
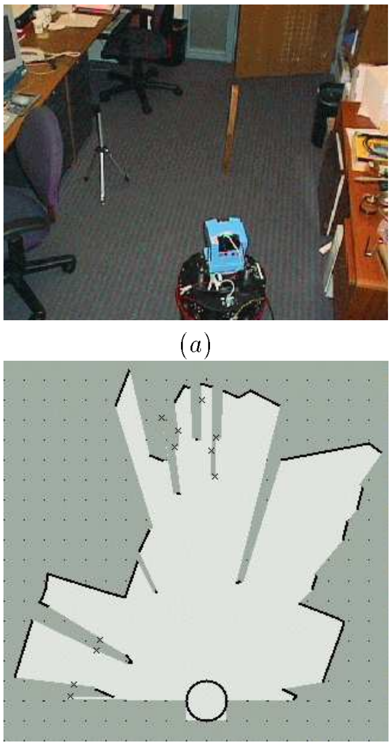


Figure 2: Polyline fit for the data of Figure $1(b)$

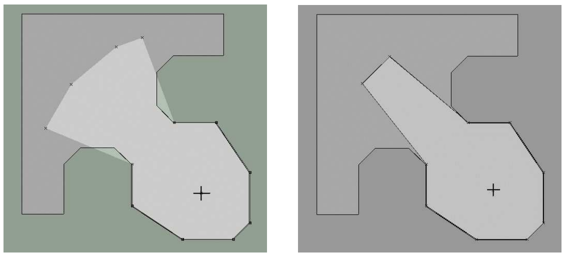


(a)

(b)

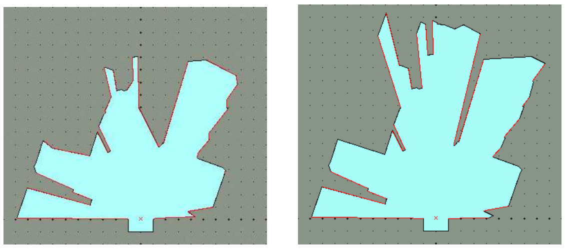
Figure 3: A more complicated example of polyline fit

Safe space computation A *safe region* f is then computed, in which the robot is guaranteed to move without collision. f is the visibility region of the robot's centerpoint computed by taking the range/incidence constraints on sensing into consideration. While the range restrictions (r and R) have an obvious impact on f, incidence (η) has a more subtle effect illustrated in Figure 4. The light-grey region in (a) is the visibility region computed with $r = 0$ and some given R, but ignoring the incidence constraint. It contains obstacles



(a) (b)

Figure 4: Impact of incidence constraint on safe region



(a) (b)

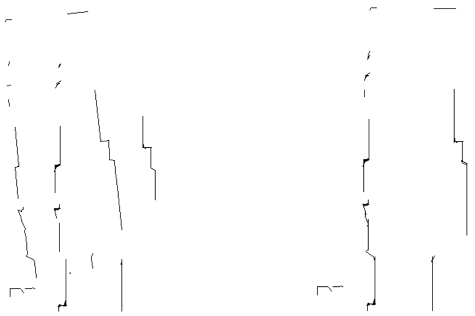
Figure 5: Computed safe regions: (a) $R = 275$ cm and $\eta = 50$ dg; (b) $R = 550$ cm and $\eta = 85$ dg

bounded by surfaces oriented at grazing angles relative to the sensor. Taking the incidence constraint specified by $\eta = 60$ dg into account yields the safe region in (b).

The safe region f is bounded by both the polylines in p and free curves connecting the polyline endpoints so that no unseen obstacle (at the sensor's height) lies in f's interior. Establishing the equations of these free curves is rather straighforward, but requires close attention to details and we omit it for lack of space. We approximate the free curves by polygonal lines to simplify subsequent computations. Hence, the safe regions is bounded by *solid* edges (derived from the polylines) and *free* edges (derived from the free curves). The pair $m = (p, f)$ is the *local model* constructed at q.

Figure 5 shows two safe spaces computed for the scene of Figure 3 with different values of the maximal range R and the incidence angle η (the minimal range r was set to 0 in both cases).

Model alignment Let $M = (P, F)$ be the partial layout model built prior to reaching q. P is the set of polylines of M and F the associated safe region. A best match is computed between the line segments in P and those in p, yielding an euclidean transform aligning P with p. Our algorithm is similar to a previous algorithm used to discover and align substructures shared by 3-D molecular structures [21]. It samples pairs of line segments from p at random. For each pair (u_1, u_2), it finds a pair of segments (v_1, v_2) in P with the same angle. The correspondence $u_1 \rightarrow v_1,\ u_2 \rightarrow v_2$ yields a transform $T(x, y, \theta)$ obtained by solving least-square equations.



(a) (b)

Figure 6: (a) Unaligned polylines; (b) Computed alignment

The algorithm then identifies the segments of p and P which match under this transform, and creates a new correspondence $u_1 \rightarrow v_1,\ u_2 \rightarrow v_2,\ \ldots,\ u_r \rightarrow v_r$, where the u_i's and v_i's are not necessarily distinct. It recalculates the transform based on this new correspondence and evaluates the quality of fit. The above steps are repeated for each pair of line segments sampled from p, and the transform with the best quality is retained. If all segments in p are approximately parallel, the algorithm uses endpoints and odometric sensing to estimate the missing parameter of the transform.

Figure 6(a) shows two sets of polylines before alignment. The computed alignment of these two sets is displayed in (b).

Model merging The selected transform is applied to the safe space F, and the new safe space F' is computed as the union of the transformed F and f. The solid edges of F' form the new set of polylines P'. To avoid edge fragmentation, consecutive solid (resp. free) edges in P' that are almost colinear are fused into a single edge. The new model $M' = (P', F')$ is represented in the coordinate system attached to the robot at its current position q. If F' contains no free edge, the 2-D layout is complete; otherwise, M' is passed to the next iteration of the mapping process. A weaker termination test — e.g., no free edge in F' is longer than a certain threshold — can be used instead, to handle environments that are too time consumming to fully map (which is almost always the case in practice).

Figure 7 displays four partial models. The robot is at some location where it rotates to four successive orientations spaced by 90 dg. The local model in (a) was built at the first orientation. The model in (b) was obtained by merging the model of (a) with the local model generated at the second orientation, and so on. The model in (d) combines the data collected at the four orientations. In addition to illustrating model merging, Figure 7 shows the artifice we use to "give" the sensor a 360-dg field of view. Note that the matching operation compensates for any small variation in the

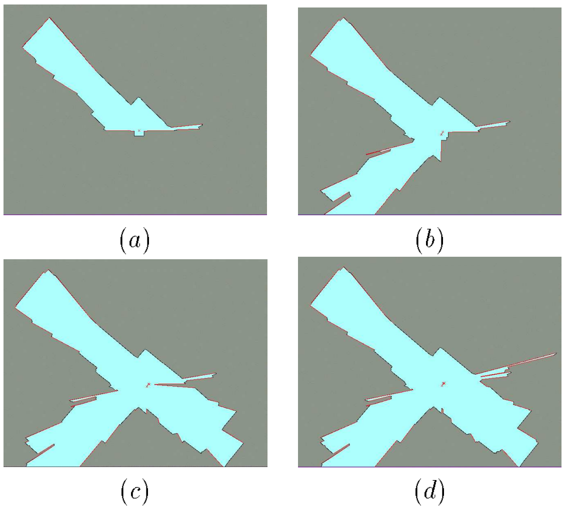


Figure 7: Merging four partial models at a given position.

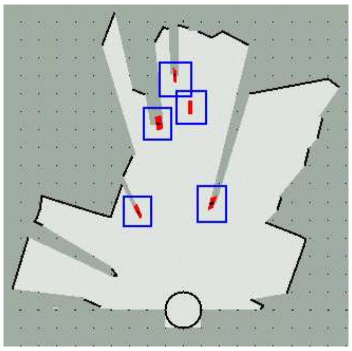


Figure 8: Small obstacles extracted from Figure 3(b).

robot's position as it rotates to a new orientation.

Dealing with small obstacles and transient objects A horizontal cross-section through an indoor environment often intersects small objects, e.g., chair legs. Such objects are detected by a good range sensor and hence appear in the polyline p extracted at a sensing position q (see Figure 3). But, due to small errors in aligning polylines, such obstacles tend to be eliminated when the union of two safe spaces is computed. Other model merging techniques could be used, but modeling small obstacles by closed polygonal contours is known to be difficult and not very realistic. In many instances, a map is more useful when small obstacles are omitted, since the positions of such obstacles tend to change often over time. So, we proceed as follows. Small obstacles result into narrow "spikes" pointing into the safe space f obtained at q. These spikes can be automatically detected. The apex of each detected spike is a small isolated polyline, which we save in a separate *small-object map*. Hence, the final model consists of a main layout (polylines and safe space) and a secondary map (small-object map). Figure 8 shows the small obstacles (apexes enclosed into square boxes) detected in the scan of Figure 3; the small obstacles include an aluminium camera tripod, a narrow wooden bar, and a swivel chair.

Merging partial models by taking the union of their safe spaces has the advantage of eliminating transient objects. Comparing solid edges in successive partial models makes it possible to detect such objects and record them into a separate representation (in a way similar to small objects). However, this capability has not yet been implemented in our current system.

4. Next-Best View Algorithm

We now describe the online NBV planning algorithm to construct a polygonal layout of an environment. An initial partial layout M_0 is constructed from the data acquired by the range sensor at the location q_0 where the robot is introduced. The layout model is then expanded iteratively. At each iteration $k = 1, 2, \ldots$, the planner uses the model $M_{k-1} = (P_{k-1}, F_{k-1})$ constructed so far to select the next sensing position q_k.

q_k must be selected in the safe space F_{k-1} (after shrinking it by the radius of the robot). To decide whether a position q in F_{k-1} is a good candidate for q_k one must estimate how much new information about the environment may be obtained at q. This criterion leads to considering positions in F_{k-1} that may see large unexplored areas *through* the free edges of F_{k-1}. We generate such positions using a randomized algorithm similar to the art-gallery algorithm presented in [18]. We sample a number of positions in F_{k-1} at random, among which q_k will eventually be selected. For each sample position q, we compute the length of the solid edges in F_{k-1} that are visible from q under the range and incidence constraints defined by r, R, and η. We retain q only if this length is greater than some threshold. This filter ensures that the line matching algorithm will work reliably at the position q_k. We measure the goodness of every remaining position q by a function of both the area of the unexplored subset of the environment that may be visible from q "through" the free edges and the length of the shortest path inside F_{k-1} connecting q_{k-1} to q. We choose q_k to be the sample q that maximizes this function. Taking path length into account in the goodness function is important, as it prevents the robot from going back and forth between regions with similar potentials in term of visibility gain.

The amount of new space potentially visible at q is evaluated by casting a fixed number of equally spaced rays from q. Along each ray, we consider the segment between r and R from q. If this segment intersects a solid edge, we eliminate the portion beyond the edge

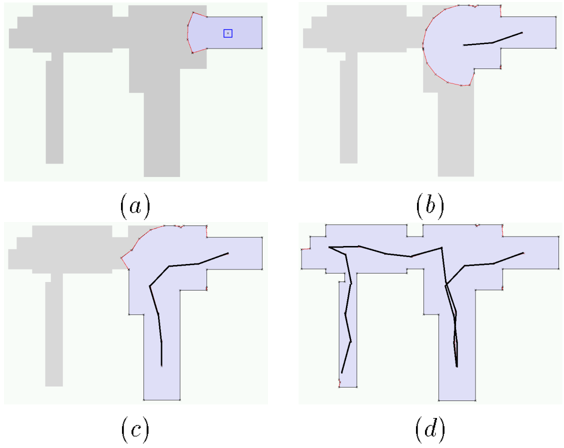


Figure 9: Example 1 of model construction in simulation

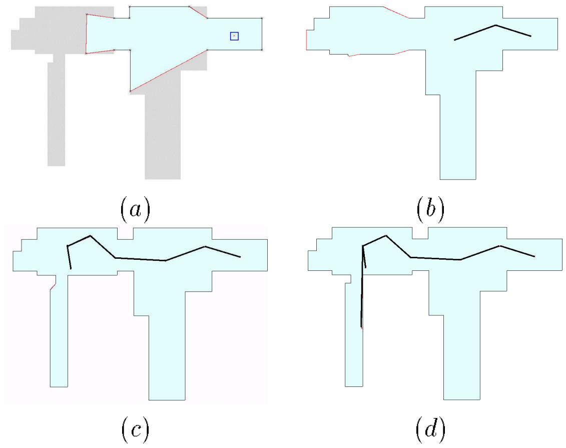


Figure 10: Example 2 of model construction in simulation

and we compute the length ℓ of the portions of the remaining segment which are outside F_{k-1}. We estimate the amount of new space visible from q as the sum of the ℓ's computed along all rays casted from q.

Figure 9 shows partial models generated at several iterations (0, 2, 6, and 19) during a run of the planner on simulated data, and the path followed by the robot. The layout model was completed in 19 iterations. Because path length is taken into account in the goodness function, the robot fully explores the bottom-right corridor before moving to the left corridor. Figure 10 shows another series of snapshots for the same environment, the same initial position, but greater R and η. The motion strategy is simpler with only 7 iterations required.

5. Implementation and Experimentation

Robot platform Our robot is a Nomadic SuperScout wheeled platform equipped with a Sick laser range sensor (Figures 1(a) and 3(a)). This sensor uses a time-of-flight technique to measure distances. The robot's onboard processor (Pentium 233 MMX) acquires a 360-point 180-dg scan in 32 ms through a SeaLevel 500Kbs PCI serial card. At each sensing location, a 360-dg view is obtained by taking 4 scans (Figure 7).

The onboard processor is connected to the local-area network via 2 Mbs radio-ethernet. The NBV planner and the navigation monitor run on an offboard Pentium II 450 MHz Dell computer. The software is written in C++ and uses geometric functions available in the LEDA-3.8 library.

System architecture The software consists of several modules executing specialized functions and communicating using TCP/IP socket connections under a client/server protocol.

A *Sick sensor server* handles communication through the SeaLevel card. It allows the connecting clients to assume they read data from a device resembling an ideal sensor and offers the following capabilities:

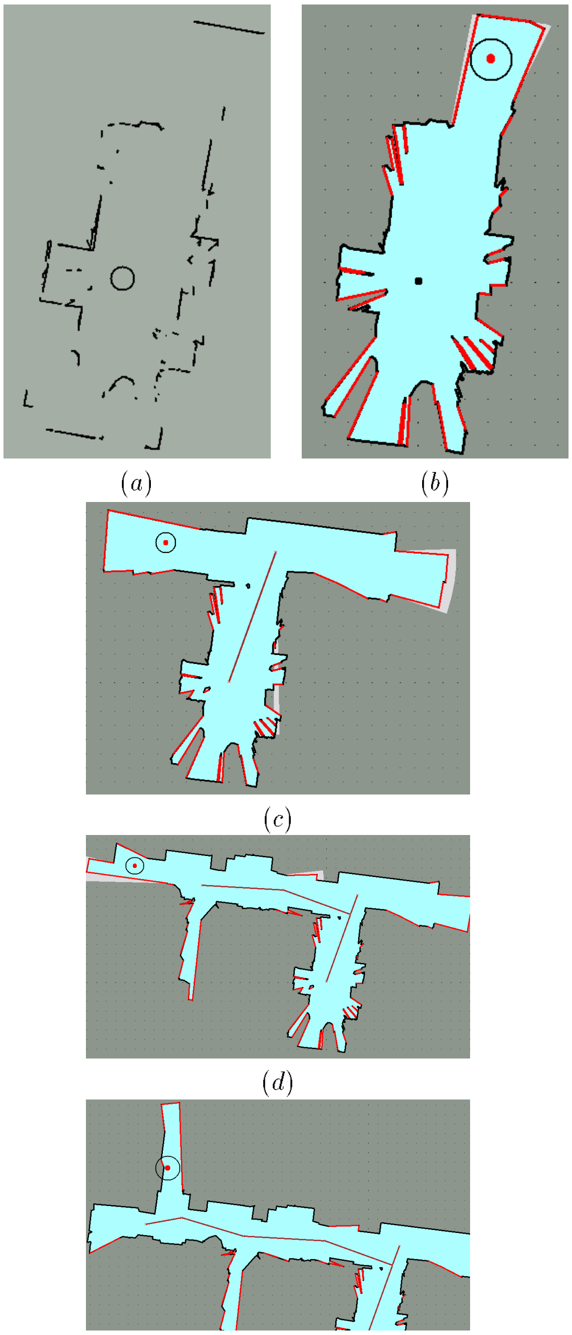
- choice among 3 speed modes: 1, 5, and 30 scans/sec,
- batch transmission of multiple scans on request,
- scan averaging using sensor electronics,
- operation in continuous mode,
- real-time polyline fitting with 2.5-cm error bound.

Since our fitting technique is fast enough to be performed on-line at any of the 3 speed modes, it is included in the server capabilites, which reduces the amount of data transmitted to clients.

A *navigation monitor* allows a user to supervise the exploration process. The user may query the NBV module for the next position and/or the most recent environment model, or decide to select the next position manually. He may also operate the robot and laser in continuous mode, receiving scan updates every 0.5 sec. The navigation module is also reponsible for aligning new data with the previous model using robot's odometry for pre-alignment before calling the model alignment function. The computed transform is sent to the NBV module with each new scan.

An *NBV module* computes the next position given the current model of the world. The model is updated whenever a new scan is received.

Layout construction Figure 11 shows a sequence of partial layouts built by the intergrated robot system in our laboratory. The robot is initially placed in a messy office with many small obstacles (chair and table legs, and cables). The polylines extracted at this initial location are shown in (a) and the safe region is displayed in (b) along with the next sensing position computed by the NBV planner. The safe region is bounded by many free edges forming spikes, but our candidate evaluation function automatically detects that little additional space can be seen through such free edges. Consequently, the NBV planner reliably selects the next sensing position near the exit door of the office. Figures (c)-(e) show the safe region after the robot has reached the second, fourth and sixth sensing positions, respectively. At that stage, the layout consists of the initial office, a second office (incompletely mapped), and two perpendicular corridors.



(a) (b) (c) (d) (e)

Figure 11: Experiment with the integrated system

6. Current and Future Research

The experiments with our system, both in simulated and real environments, show that the NBV planner can considerably reduce the length of a robot motion and the number of sensing operations to obtain a polygonal layout model of an indoor environment. This claim is difficult to quantify, as this would require extensive comparison with strategies produced by other means (e.g., trained human operators). But in none of our experiments could we imagine a strategy significantly better than the one produced by the planner. Our system also demonstrates that current sensing technology makes it possible to construct good-quality polygonal layouts. As we have argued in Section 2, this representation has significant advantages over other representations.

The most obvious limitation of our system is that it only builds a cross-section of the environment at fixed height. Therefore, important obstacles may not be sensed. One way to improve sensing is to emit laser rays at multiple heights. The techniques described in this paper would remain applicable with very little changes.

A more fundamental limitation of the system is its lack of error-recovery capabilities. Any serious error in extracting polylines from data points or in matching polylines can result in an inacceptable model. For that reason, our basic algorithms are very conservative. For instance, to avoid generating incorrect polylines, we often discard many points delivered by the sensor. This may lead the robot to sense the same part of an environment several times, hence resulting into a longer motion path. The model alignment function still remains under the control of the user, who calls it back whenever the computed alignment is not accurate enough.

Matching transforms are computed locally to align the current partial model with the local model generated at the robot's current location. Once a transform has been computed it is never revised. In the future, we plan to keep track of the successive local models and transforms to make possible the periodic optimization of a global matching criterion, especially after the robot has completed a long "loop". We expect that in large environments such global optimization will produce more precise layouts. In corridors bounded by parallel featureless walls, line matching only allows offsetting positioning errors in the direction orthogonal to the walls. Odometry is then used, but imprecision in the direction parallel to the walls grows bigger with distance. Global optimization should also help in dealing with this difficulty.

Finally, we expect our system to handle multiple robots with relatively minor changes. If N robots are available and they know their relative positions, a single model can be generated and a central NBV planner can compute N positions that, together, can potentially see a large area through the free edges of the current safe region. If the robots do not know their relative positions, they can act independently (distributed planning) until their partial models have enough overlap to make alignment possible, and then switch to centralized planning.

Acknowledgments

This work was funded by DARPA/Army contract DAAE07-98-L027, ARO MURI grant DAAH04-96-1-007, and NSF grant IIS-9619625. A. Efrat was also supported by a Rotschild Fellowship. L. Guibas, C.-Y. Lee, D. Lin, R. Murrieta, and S. Yao helped developing the ideas presented in this paper and/or implementing the robot system. T.M. Murali is now an employee of Compaq Computer Corp.

References

[1] Chatila, R. and J.P. Laumond (1985). Position Referencing and Consistent World Modeling for Mobile Robots. *Proc. IEEE Int. Conf. Rob. and Aut.*, 138–143.

[2] Elfes, A. (1987). Sonar-Based Real World Mapping and Navigation. *IEEE J. Rob. and Aut.*, RA-3(3):249–265.

[3] Thrun, S., D. Fox, and W. Burgard (1998). Probabilistic Mapping of an Environment by a Mobile Robot. *Proc. IEEE Int. Conf. Rob. and Aut.*, Leuven, Belgium.

[4] Choset, H. and J. Burdick (1997). 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), 47–61.

[5] Teller, S. (1998). Automated Urban Model Acquisition: Project Rationale and Status. *Proc. 1998 DARPA Image Understanding Workshop*.

[6] Kuipers, B., R. Froom, W.K. Lee, and D. Pierce (1993). The Semantic Hierarchy in Robot Learning. In *Robot Learning*, J. Connell and S.Mahadevan (eds.), Kluwer Acad. Pub.

[7] Moutarlier, P. and R. Chatila (1989). Stochastic Multisensory Data Fusion for Mobile Robot Location and Environment Modeling. *Proc. 5th Int. Symp. on Rob. Res.*, H. Miura and S. Arimoto (eds.), 85–94.

[8] Leonard, J.J. and H.F. Durrant-Whyte (1991). Simultaneous Map Building and Localization for an Autonomous Mobile Robot. *Proc. IEEE Int. Conf. on Intelligent Robot Syst.*.

[9] Banta, J.E., Y. Zhien, X.Z. Wang, G. Zhang, M.T. Smith, and M. Abidi (1995). A Best-Next-View Algorithm for Three-Dimensional Scene Reconstruction Using Range Images. *Proc. SPIE*, Vol. 2588, 418–429.

[10] Curless, B. and Levoy, M., A Volumetric Method for Building Complex Models from Range Images. *Proc. ACM SIGGRAPH 96*.

[11] Kakusho, K., T. Kitahashi, K. Kondo, and J.C. Latombe (1995). Continuous Purposive Sensing and Motion for 2-D Map Building. *Proc. IEEE Int. Conf. of Syst., Man and Cyb.*, Vancouver (BC), 1472–1477.

[12] Maver, J. and R. Bajcsy (1993). Occlusions as a Guide for Planning the Next View. *IEEE Tr. on Patt. Anal. and Mach. Intell.*, **15**(5), 417–433.

[13] Pito, R. (1995). *A Solution to the Next Best View Problem for Automated CAD Model Acquisition of Free-Form Objects Using Range Cameras*, TR 95-23, GRASP Lab., U. of Pennsylvania.

[14] Wixson, L. (1994). Viewpoint Selection for Visual Search. In *Proc. IEEE Conf. on Computer Vision and Pattern recognition*, 800–805.

[15] González-Baños, H.H., L.J. Guibas, J.C. Latombe, S.M. LaValle, D. Lin, R. Motwani, and C. Tomasi (1997). Motion Planning with Visibility Constraints: Building Autonomous Observers. In *Robotics Res.*, Y. Shirai and S. Hirose (eds.), Springer, New York (NY), 95–101.

[16] González-Baños, H., J.L. Gordillo, D. Lin, J.C. Latombe, A. Sarmiento, C. Tomasi (1999). The Autonomous Observer: A Tool for Remote Experimentation in Robotics. *Proc. SPIE Conf. on Telemanipulator and Telepresence Techn.*, Boston (MA).

[17] O'Rourke, J. (1997). Visibility. In *Handbook of Discrete and Comp. Geometry*, J.E. Goodman and J. O'Rourke (eds.), CRC Press, Boca Raton (FL), 467–479.

[18] González-Baños, H.H. and J.C. Latombe (1998). Planning Robot Motions for Range-Image Acquisition and Automatic 3-D Model Construction. In *Proc. AAAI Fall Symp.*, AAAI Press, Menlo Park (CA).

[19] Guibas, L.J., J.C. Latombe, S.M. LaValle, D. Lin, and R. Motwani (1997). Visibility-based pursuit-evasion in a polygonal environment. *Proc. 5th Workshop on Algorithms and Data Structures (WADS'97)*, Springer, New York (NY), 17–30.

[20] LaValle, S.M., H.H. Gonzalez-Bãnos, C. Becker, and J.C. Latombe (1997). Motion strategies for maintaining visibility of a moving target. *Proc. IEEE Int. Conf. on Rob. and Aut.*.

[21] Finn, P.W., L.E. Kavraki, J.C. Latombe, R. Motwani, C. Shelton, S. Venkatasubramanian, and A. Yao (1998). RAPID: Randomized Pharmacophore Identification for Drug Design. *J. of Comp. Geometry: Theory and Applic.*, 10:263-272.