Navigation Strategies for Exploring Indoor Environments

Héctor H. González-Baños$^{(1)}$ Jean-Claude Latombe$^{(2)}$

(1) Honda R&D, Americas; 800 California St. Suite 300; Mountain View, CA 94041

(2) Department of Computer Science; Stanford University; Stanford, CA 94305

e-mail:{hhg,latombe}@robotics.stanford.edu

Abstract

This paper investigates safe and efficient map-building strategies for a mobile robot with imperfect control and sensing. In the implementation, a robot equipped with a range sensor builds a polygonal map (layout) of a previously unknown indoor environment. The robot explores the environment and builds the map concurrently by patching together the local models acquired by the sensor into a global map. A well-studied and related problem is the Simultaneous Localization and Mapping (SLAM) problem, where the goal is to integrate the information collected during navigation into the most accurate map possible. However, SLAM does not address the sensor-placement portion of the map-building task. That is, given the map built so far, where should the robot go next? This is the main question addressed in this paper. Concretely, an algorithm is proposed to guide the robot through a series of "good" positions, where "good" refers to the expected amount and quality of the information that will be revealed at each new location. This is similar to the Next-Best View (NBV) problem studied in Computer Vision and Graphics. However, in mobile robotics the problem is complicated by several issues, two of which are particularly crucial. One is to achieve safe navigation despite an incomplete knowledge of the environment and sensor limitations (e.g., in range and incidence). The other is the need to ensure sufficient overlap between each new local model and the current map, in order to allow registration of successive views under positioning uncertainties inherent to mobile robots. To address both issues in a coherent framework, the paper introduces the concept of a safe region, defined as the largest region that is guaranteed to be free of obstacles given the sensor readings made so far. The construction of a safe region takes sensor limitations into account. The paper also describes an NBV algorithm that uses the safe-region concept to select the next robot position at each step. The new position is chosen within the safe region in order to maximize the expected gain of information under the constraint that the local model at this new position must have a minimal overlap with the current global map. In the future, NBV and SLAM algorithms should reinforce each other. While a SLAM algorithm builds a map by making the best use of the available sensory data, an NBV algorithm like the one proposed here guides the navigation of the robot through positions selected to provide the best sensory inputs.

1 Introduction

Automatic model building is a fundamental task in mobile robotics [3, 5, 6, 32]. The basic problem is the following: After being deployed 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. However, in practice, this problem turns out to be a difficult one. First, one must choose an adequate representation of the environment – e.g., topological maps [7], polygonal layouts [6], occupancy grids [12], 3-D models [31], or feature-based maps [18]. Second, this representation must be extracted from imperfect sensor readings [9, 21, 30]. Finally, to be truly automatic, the robot must decide on its own the necessary motions to construct the model [14, 16].

Past research has mainly focused on extracting relevant features (e.g., edges, corners) from raw sensor data, and on integrating these into a consistent model of the environment. The latter operation is related to the *simultaneous localization and mapping* (SLAM) problem [10, 11, 17, 20, 24]. SLAM techniques seek to integrate the information collected by the robot during navigation into the most accurate map possible [4]. However, SLAM does not address the sensor-placement problem within a map-building task. That is, given the map known so far, where should the robot move next?

Computing a sequence of sensing positions based on data acquired at previous locations is referred to as the *next-best-view* (NBV) problem. While a SLAM algorithm builds a map by making the best possible use of the available sensory data, an NBV algorithm guides the navigation of the robot through positions chosen to provide the best possible sensory inputs. The NBV problem has attracted considerable attention in the past, both from the Computer Vision and Graphics communities [1, 8, 21, 27, 34]). But most proposed techniques do not apply well to mobile robots. There are at least two reasons for this:

1. A robot must avoid colliding with uncharted obstacles. Existing NBV techniques do not address safe navigation constraints because they were designed for systems that build a model of a relatively small object using a sensor moving around the object. Collisions are not an issue for sensors that operate outside the convex hull of the scene of interest. In contrast, in mobile robotics, the sensor navigates *within* the scene's convex hull. Hence, safe navigation must be taken into account when computing the robot's next-best-view.

2. Due to errors in odometry (e.g., wheel slippage), a mobile robot must be able to localize itself with respect to the partially-built map. Robot localization involves aligning (or registering) new images with the current map. A number of alignment techniques exist, which all require a minimum overlap between each new image and portions of the environment seen by the robot at previous locations [24]. An NBV technique for a mobile robot should take this requirement into account when computing the next view. This issue is partially addressed in [26].

The next-best-view is a variant of the sensor placement problem. Previous work in this subject study the placement of one or several sensors to best achieve a certain task [2], usually under the assumption that the workspace is known. The NBV is related to this general problem, but the computation of a new sensor position is done on-line as the map is constructed. The computation of sensing positions for exploration, surveillance, inspection or tracking are in the end extensions to the art-gallery problem [15, 29, 33].

In this paper, we introduce the concept of a *safe region*, which is the largest region guaranteed to be safe given the history of sensor readings. Using this notion, we propose an algorithm that iteratively builds a map by executing union operations over successive safe regions, and use this same map for planning safe motions. Safe regions are also used to anticipate the overlap between future images and the current, partially-built map and to check that this overlap satisfies the requirement of the alignment algorithm. Finally, they enable our NBV algorithm to estimate the gain of information for each new position candidate and select the position that is the most likely to reveal large unexplored areas of the environment. Thus, the safe-region concept provides a single coherent framework to incorporate safe-navigation and image-alignment considerations into the computation of the next-best view.

This article is divided in two parts. Part I presents the notion of a safe region (Section 2), and the complexity of computing this region from sensor data (Section 3). The general form of our NBV algorithm is described in Section 4.

In Part II, we describe an experimental system (a Nomadic SuperScout robot equipped with a Sick laser range finder) implementing our map-building algorithm. Operations not covered in Part I are presented in Section 5. Details of the specific NBV procedure embedded in our system are described in Section 6. The system architecture and experimental results are presented in Section 7.

Finally, in Section 8 we list some shortcomings of our work and important extensions to be investigated in future research.

Part I: Concepts and Algorithms

We first introduce the concept of a safe region. Next, we establish a key result, Theorem 3.1, which states that a local safe region is no more complex than the visibility region computed under a classic visibility model (unrestricted "line-of-sight" model). The proof of this theorem provides a means to compute safe regions. Safe regions are then used to iteratively construct a map. This approach leads to a general and flexible NBV algorithm, which is the central component of the experimental system described in Part II.

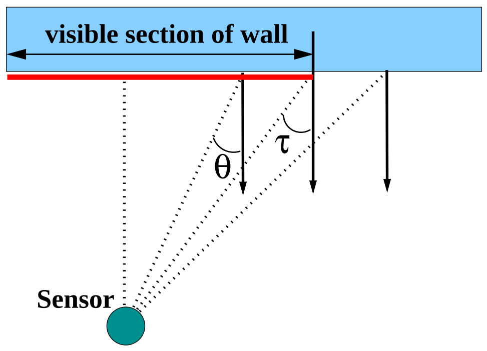


Figure 1: Incidence constraint: the visible section of the wall must satisfy $\mid \theta \mid \leq \tau$.

2 Notion of a Safe Region

Let us assume that the robot is equipped with a polar range sensor measuring the distance from the sensor's center to objects lying in a horizontal plane located at height h above the floor within distance r_{max} (sensor's maximum range). In addition, range-finders cannot reliably detect surfaces oriented at grazing angles with respect to the the line of sight — i.e., when the angle between the surface normal and the line-of-sight is greater than some τ (Figure 1). Hence, we add an *incidence constraint* to our sensor model. Formally, this model is the following:

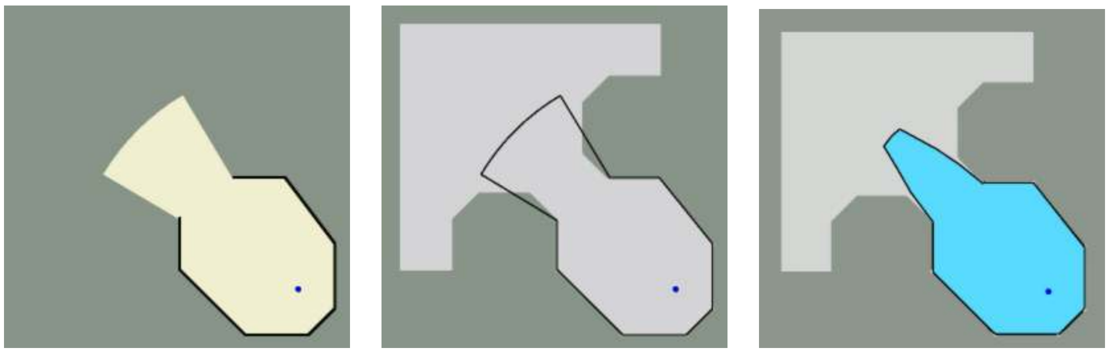
Definition 2.1 (Visibility under Incidence and Range Constraints) *Let the open subset $\mathcal{W} \subset \Re^2$ denote the actual workspace layout. Let $\partial\mathcal{W}$ be the boundary of $\mathcal{W}$. A point $\boldsymbol{w} \in \partial\mathcal{W}$ is* visible *from a point $\boldsymbol{q} \in \mathcal{W}$ if the following conditions are true:*

1. Line-of-sight constraint*: The open line segment $S(\boldsymbol{w}, \boldsymbol{q})$ joining $\boldsymbol{q}$ and $\boldsymbol{w}$ does not intersect $\partial\mathcal{W}$.*
2. Range constraint*: $d(\boldsymbol{q}, \boldsymbol{w}) \leq r_{max}$, where $d(\boldsymbol{q}, \boldsymbol{w})$ is the Euclidean distance between $\boldsymbol{q}$ and $\boldsymbol{w}$ and $r_{max} > 0$ is an input constant.*
3. Incidence constraint*: $\angle(\boldsymbol{n}, \boldsymbol{v}) \leq \tau$, where $\boldsymbol{n}$ is a vector perpendicular to $\partial\mathcal{W}$ at $\boldsymbol{w}$, $\boldsymbol{v}$ is a vector oriented from $\boldsymbol{w}$ to $\boldsymbol{q}$, and $\tau \in [0, \pi/2]$ is an input constant.*

With no loss of generality, we assume that the sensor is located at the origin of the coordinate system. Because of the line-of-sight constraint, any ray departing from the origin will intersect the *visible* portion of $\partial\mathcal{W}$ only once. Let this visible contour be described in polar coordinates by some function $r(\theta)$. This function is piecewise continuous. It is discontinuous at those critical values of θ where either an occlusion occurs or the range/incidence constraints cease to be satisfied. Let a and b be two successive critical values of θ. In the interval $\theta \in (a, b)$, either r is continuous or it is undefined (i.e., $\partial\mathcal{W}$ is not visible). Thus, the visible boundary is composed of pieces, and each piece i can be described by a continuous function $r = r_i(\theta; a_i, b_i)$ $\forall\ \theta \in (a_i, b_i)$. The sensor's output is assumed to be as follows:

Definition 2.2 (Range Sensor Output) *The output of the range sensor is an ordered list Π of polar functions, where every $\{r(\theta; a, b)\} \in \Pi$ is a continuous function of θ over the range interval (a, b) and undefined elsewhere. Each function in Π describes a section of $\partial\mathcal{W}$ visible from the origin under Definition 2.1. Π contains at most one function defined for any $\theta \in (-\pi, \pi]$ (i.e., no two function domains overlap), and the list is ordered counter-clockwise.*

Given an observation Π made at location q, we define the *local safe region* $s_l(q)$ to be the largest region guaranteed to be free of obstacles. While the effect of the range constraint on $s_l(q)$ is obvious, the effect of the incidence constraint is more subtle. To illustrate, consider Figure 2(a). The sensor has detected the obstacle contour shown in bold black segments. A naive approach may join the detected contour to the perimeter limit of the sensor using line segments and declare this region free from obstacles (this region is



(a) (b) (c)

Figure 2: Effect of incidence on safe regions.

shown in light color (yellow) in (a) and its contour is drawn with black lines in (b)). However, because of the incidence constraints, this region may not be safe as it is shown in (b), where the actual workspace is displayed in light-gray. The contour of the true safe region that can be derived from the sensor readings shown in (a) is drawn in black in (c), for an incidence constraint of $\tau = 70$ deg.

3 Computational Complexity of a Safe Region

We calculate s_l by computing a description of its boundary. Let ∂s_l be the boundary of the region s_l. ∂s_l is composed of solid and free curves. A *solid curve* is a visible section of $\partial \mathcal{W}$, and is represented by an entry in the list Π.

Given two solid curves represented by $\{r_1(\theta; a_1, b_1), r_2(\theta; a_2, b_2)\} \subseteq \Pi$, r_2 is said to *succeed* r_1 if no other element in Π is defined in the interval $[b_1, a_2]$. A curve $f(\theta; b_1, a_2)$ joining a pair (r_1, r_2) of successive solid curves is called a *free curve* if: (1) any ray erected from the origin is guaranteed to intersect the curve f before any undetected obstacle in the polar region $b_1 < \theta < a_2$; and (2) the area enclosed by f in this polar region is as large as possible.

In order to compute the local safe region at q, we need to compute the free curves that join each pair of successive solid curves in $\Pi(q)$. First, we make the assumption that ∂W is continuously differentiable. Later, we relax this assumption to include the case when ∂W is piecewise differentiable.

3.1 Continuously Differentiable ∂W

Under the assumption that ∂W is continuously differentiable, the complexity of f is O(1). In fact, a free curve f can be described using no more than 3 function primitives:

Theorem 3.1 (Free Curves) *Let $r_2(\theta; a_2, b_2)$ succeed $r_1(\theta; a_1, b_1)$ in the output list Π of a sensor operating under Definition 2.2 and located at the origin. If $\partial \mathcal{W}$ is continuously differentiable, then the free curve $f(\theta; b_1, a_2)$ connecting r_1 to r_2 consists of at most three pieces. Each piece is either a line segment, a circular arc, or a section of a logarithmic spiral of the form $r = r_o \exp(\pm\lambda\theta)$ (where r_o is a constant and $\lambda = \tan\tau$).*

The proof of Theorem 3.1 is given in the appendix, and we only present some aspects of it here. It is based on a continuity argument. Although its shape may be arbitrary, $\mathcal{W}$ represents a physical space containing physical objects. Its boundary $\partial \mathcal{W}$ is a set of Jordan curves (i.e., closed loops in the plane which do not intersect themselves), and each Jordan curve is a continuous curve. If additionally we enforce that the curve is differentiable, then we can show that the unobserved section of ∂W lying between a pair of successive solid curves cannot be arbitrarily close to the sensor. To illustrate this point, we sketch here one case of the proof.

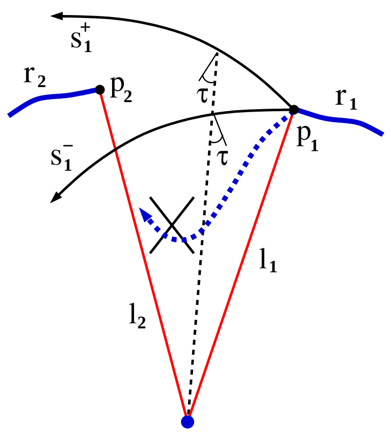


Figure 3: An example of a free curve construction.

Let $r_1(\theta; a_1, b_1)$ and $r_2(\theta; a_2, b_2)$ be two successive solid curves, and suppose that the incidence constraint was exceeded at $\theta = b_1$ (Figure 3). Hence, the normal to $\partial\mathcal{W}$ after r_1 is oriented at an angle larger than τ with respect to the sensor. What can we say about the unobserved section of ∂W lying between r_1 and r_2?

Define the points $p_1 = (\rho_1, b_1)$ and $p_2 = (\rho_2, a_2)$, where $\rho_1 = r_1(b_1)$ and $\rho_2 = r_2(a_2)$. Suppose that the boundary of $\mathcal{W}$ continues after p_1 with its surface normal constantly oriented at *exactly* an angle τ with respect to the sensor's line-of-sight. This curve in polar coordinates has the form $r = r_o \exp\left[\pm\lambda(\theta - \theta_o)\right]$, where $r_o = \rho_1$ and $\theta_o = b_1$. This last equation defines two spirals: a spiral s_1^+ growing counter-clockwise from $p_1 = (\rho_1, b_1)$, and a spiral s_1^- shrinking counter-clockwise from p_1. So far, we may conclude that $\partial\mathcal{W}$ must continue from p_1 in the counter-clockwise direction either "above" s_1^+ or "below" s_1^-; otherwise, the incidence constraint would not have been violated.

But $\partial\mathcal{W}$ cannot continue below s_1^- for the case in Figure 3. Indeed, if $\partial\mathcal{W}$ continues below s_1^-, then $\partial\mathcal{W}$ must bend inwards (i.e., ∂W bends toward the sensor immediately after r_1). We know that $\partial\mathcal{W}$ does not cross the origin, otherwise nothing will be visible under Definition 2.1 and Π would be empty. Thus, $\partial\mathcal{W}$ must bend outwards and do so before cutting the ray l_2 passing through the origin and p_2, since otherwise r_2 would be occluded. Since $\partial\mathcal{W}$ is differentiable, there must be a point p where the normal to $\partial\mathcal{W}$ points towards the origin. Therefore, the vicinity of p is a visible portion of $\partial\mathcal{W}$, which violates our assumption that r_2 succeeds r_1. So, $\partial\mathcal{W}$ must continue above s_1^+.

A similar reasoning allows us to account for every possible combination of events, depending on whether the sensor's line-of-sight is occluded, the range constraint is exceeded, or the incidence constraint is exceeded. All cases are described in detail in the appendix.

The region $s_l(q)$ is topologically equivalent to a classic visibility region. Indeed, when the sensor restrictions in Definition 2.1 are relaxed, the safe region becomes the visibility region. Several properties and algorithms that apply to visibility regions also apply to safe regions. For example, the segment connecting q with p is entirely contained in $s_l(q)$ for any $p \in s_l(q)$. Hence, $s_l(q)$ is a *star-shaped region* — a region that are entirely observable from at least a single interior. Likewise, a local safe region is a simply connected region — all the paths joining a pair of points p_1 and p_2 inside $s_l(q)$ are homotopic to one another.

3.2 Piecewise Differentiable ∂W

We now consider the case when ∂W is not differentiable at a finite number of locations. These locations are the *corners* of the boundary.

Let $\boldsymbol{n}^+$ and $\boldsymbol{n}^-$ be the normals to $\partial\mathcal{W}$ immediately after and before a corner. We expand our notion of visibility to include corners by saying that a corner is visible if it is within range and if the average of $\boldsymbol{n}^+$ and $\boldsymbol{n}^-$ satisfies the incidence constraint in Definition 2.1. Although strictly speaking the normal to ∂W at a corner is undefined, we assume that an idealized range sensor will see a corner if the average of $\boldsymbol{n}^+$ and $\boldsymbol{n}^-$ satisfies the incidence constraint.

Under the above assumption, the analysis of the previous subsection remains valid, except that certain entries in Π could be single points (i.e., corners). It is easy to verify that any wedge-shaped corner within range is visible if $\tau \geq 45$ deg. Thus, $s_l(q)$ can always be constructed for reasonable incidence restrictions.

In practice, a range sensor does not have infinite resolution. Instead of returning a list of curves, it usually produces a list of points, and an additional processing step fits curves through these points. Hence, a corner could go undetected even if the conditions in Definition 2.1 are satisfied. In our experimental system, we further assume that the inner angle of the corner is large enough to guarantee that at least one of the two incident edges will be (fully or partially) detected by the sensor. This assumption is similar in nature to other assumptions that are implicit in Definition 2.1 — e.g., that no surface is perfectly transparent or completely reflective.

4 A Next-Best-View Algorithm

In a static environment, a safe region remains safe under the union operation. Hence, the layout model of an environment can be built iteratively. A first partial layout — a local safe region — is constructed from the data acquired by the range sensor at the robot's initial position q_0. At each iteration, the algorithm updates the layout model by computing the union of the safe region built so far with the local safe region generated at the new position q_k. The new safe region is then used to select the next sensing position q_{k+1}. To compute this position, our NBV procedure first generates a set of potential candidates. Next, it evaluates each candidate according to the expected gain of information that will be sensed at this position, the needed overlap between the two partial layout models (to ensure good alignment), and the motion cost required to move to the new position. These steps are illustrated in Figure 4 and described below.

4.1 Incorporating Model Alignment and Merging Operations

Let $M_g(q_{k-1}) = \langle \Pi_g(q_{k-1}), S_g(q_{k-1}) \rangle$ be the partial *global* model built at q_{k-1}. The term $S_g(q_{k-1})$ designates the union of all the local safe regions up to stage $k-1$. The boundary of $S_g(q_{k-1})$ is composed of free and solid curves, the latter representing physical sections of $\partial\mathcal{W}$. $\Pi_g(q_{k-1})$ stands for the list of solid curves in the boundary of $S_g(q_{k-1})$.

The robot now performs a new sensing operation at the location q_k. From this local measurement $\Pi_l(q_k)$, we compute the local safe region $s_l(q_k)$ as indicated by the proof of Theorem 3.1. Let $m_l(q_k) = \langle \Pi_l(q_k), s_l(q_k) \rangle$ be the *local* model at q_k.[1]

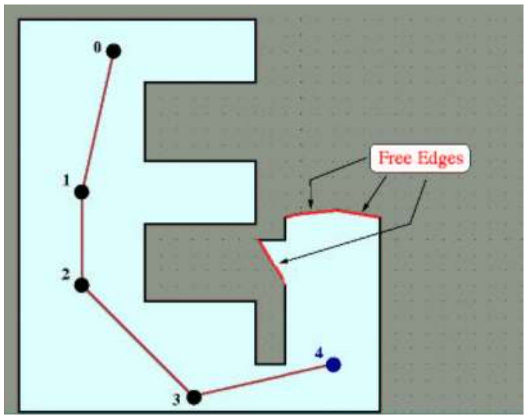
Let ALIGN be the algorithm used to compute the transform T aligning $m_l(q_k)$ with $M_g(q_{k-1})$ by matching the line segments of $\Pi_l(q_k)$ and $\Pi_g(q_{k-1})$. The details of this algorithm are not important here and we assume that ALIGN is given. However, we do *not* assume that the algorithm is perfect: ALIGN computes a correct T only when there is enough overlap between $m_l(q_k)$ and $M_g(q_{k-1})$. The specific amount of required overlap depends on ALIGN and it is an input to the NBV algorithm.

Once T has been calculated, the new global safe region $S_g(q_k)$ is computed as the union of $T(S_g(q_{k-1}))$ and $s_l(q_k)$. The new model $M_g(q_k) = \langle \Pi_g(q_k), S_g(q_k) \rangle$ is represented in a coordinate frame centered at the robot's current position q_k.

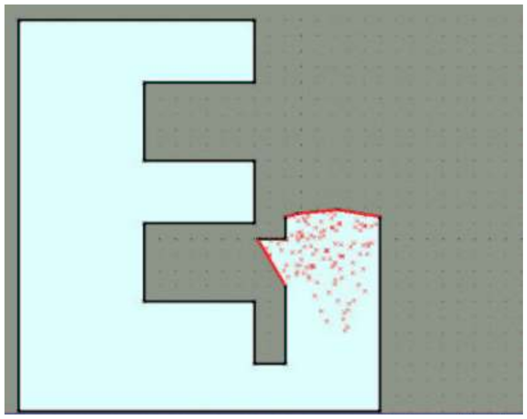
4.2 Candidate Generation

The future position q_{k+1} should potentially see large unexplored areas *through* the free curves bounding $S_g(q_k)$ (by definition, unexplored areas cannot be observed through solid curves). However, we are constrained in our choices for q_{k+1}. The robot has to be entirely contained inside $S_g(q_k)$ at the next location q_{k+1}, which must also be reachable from q_k by a collision-free path. Furthermore, the function ALIGN must successfully find a transform T at the next position q_{k+1}. To achieve all these conditions, we proceed as follows:

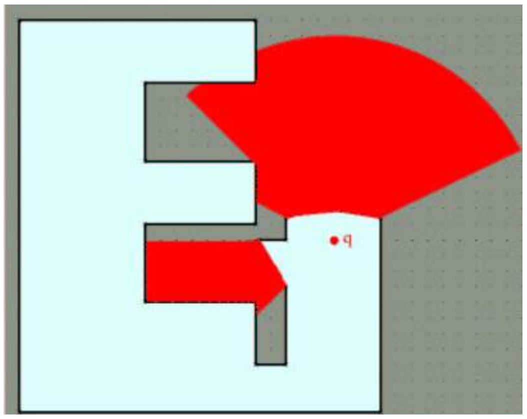
[1]Π_l denotes the same list as Π. The subindex l is now used to differentiate this list from Π_g (the list of solid curves bounding the global model).



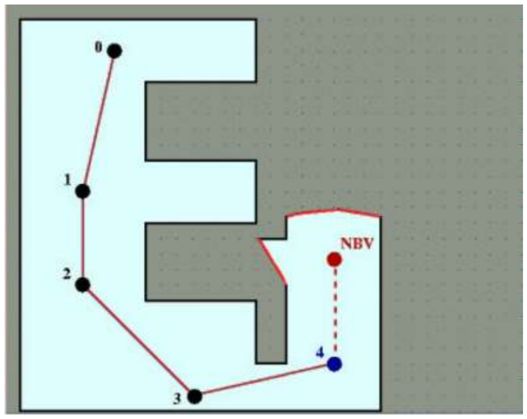
(a)



(b)



(c)



(d)

Figure 4: Steps in a NBV computation: (a) safe region after 5 sensing operations; (b) the region within the visibility range of the free curves is sampled: (c) the potential visibility gain of a candidate q is the area $A(q)$ outside the explored region that is visible through the free curves bounding $S_g(q_k)$; (d) the best candidate is selected to optimize a criterion.

Step 1. Pick a set of possible NBV candidates $\mathcal{N}_{sam} \subset S_g(q_k)$ at random within the visibility range of the free curves bounding $S_g(q_k)$ (Figure 4(b)).

Step 2. For each $q \in \mathcal{N}_{sam}$, compute the length $\zeta(q)$ of the solid curves in $\Pi_g(q_k)$ that are visible from q under Definition 2.1 (see [25] for a survey of methods). If $\zeta(q)$ is smaller than the threshold imposed by ALIGN, then remove q from $\mathcal{N}_{sam}$.

Step 3. Invoke a path planner (many techniques are applicable here) to compute a collision-free path of the dimensioned robot between q_k and each remaining candidate q. If no path exists, remove q from $\mathcal{N}_{sam}$.

After executing these three steps, we are left with a feasible set $\mathcal{N}_{sam}$ of NBV candidates.

4.3 Evaluation of Candidates

The score of every candidate $q \in \mathcal{N}_{sam}$ is defined by the following function:

$$g(q) \quad = \quad A(q)\exp(-\lambda L(q)), \tag{1}$$

where λ is a positive constant, $L(q)$ is the length of the collision-free path computed by the planner, and $A(q)$ is a measure of the unexplored region of the environment that is potentially visible from q ($A(q)$ is defined below). q_{k+1} is selected as the sample $q \in \mathcal{N}_{sam}$ that maximizes $g(q)$.

The constant λ is used to weight the cost of a motion against the expected gain of information. A small λ means that motion is "cheap" and gives priority to the gain of information. Instead, when $\lambda \to \infty$, motion becomes so expensive that only locations near q_k are selected, even if they only produce a marginal information gain. Hence, a small λ leads the robot to first perform a quick exploration of the environment before filling in the details. Instead, a large λ leads the robot to consistently fill in the details while progressing through the environment.

Computation of $\mathbf{A(q)}$ We measure the potential visibility gain $A(q)$ of each candidate q as the area of the maximal region outside the current safe region that would be visible through the free curves bounding $S_g(q_k)$, assuming that the contour ∂W is composed only of the solid curves in $\Pi_g(q_k)$ (Figure 4(c)). So, we compute the region visible from q assuming that the free curves are transparent, and intersect this region with the complement of $S_g(q_k)$. For polygonal models, $A(q)$ can be computed by the same ray-sweep algorithm used to compute classic visibility regions [25], with the following modifications:

1. The sweeping ray may cross an arbitrary number of free curves before hitting a solid one. Therefore, the running-time of the ray-sweep algorithm becomes $\mathrm{O}(n \log(n) + n\, k_{\mathrm{f}})$, where k_{f} is the number of free curves bounding $S_g(q_k)$.

2. The resultant visibility region must be cropped to satisfy the range restrictions of the sensor. This operation can be done in time $\mathrm{O}(n\, k_{\mathrm{f}})$.

4.4 Termination Condition

If the boundary of $S_g(q_k)$ contains no free curve, then the 2-D layout is complete. Otherwise, $S_g(q_k)$ is passed to the next iteration of the mapping process. In practice, however, we use a weaker test: we stop the mapping process when the length of each remaining free curve is smaller than a specified threshold. This test is better suited to handle complex environments containing many small geometric features.

4.5 Iterative Next-Best-View Algorithm

The general NBV algorithm is given below. Implementation details, along with example runs (both in simulation and with a real robot) are described in Part II.

```
Algorithm NEXT-BEST-VIEW
Input: 1.- Current partial model $M_g(q_{k-1})$ and new sensing position $q_k$
       2.- Local sensor measurement $\Pi_l(q_k)$
       3.- Image alignment function $T = \text{ALIGN}(m_l(q_k), M_g(q_{k-1}))$
       4.- Path-planning function $\text{PATH-PLANNER}(S_g(q_k), q_k, q)$
       5.- Visibility constraints $\{r_{max}, \tau\}$
       6.- Number of samples $m$ and weighting constant $\lambda > 0$
Output: Next position $q_{k+1}$
1.  Compute the local safe region $s_l(qk)$.
2.  Compute $T = \text{ALIGN}(m_l(q_k), M_g(q_{k-1}))$ and $S_g(q_k) = s_l(q_k) \bigcup T(S_g(q_{k-1}))$. Set $M_g(q_k)$ to $\langle \Pi_g(q_k), S_g(q_k) \rangle$.
3.  Repeat until $\mathcal{N}_{sam}$ contains $m$ candidates:
        (a) Randomly sample a position $q$ in $S_g(q_k)$ within a distance $r_{max}$ of the free curves bounding $S_g(q_k)$.
        (b) Compute the length $\zeta(q)$ of the solid curves in $S_g(q_k)$ that are visible from $q$. If this number is
            less than the threshold required by ALIGN, discard $q$ and return to Step 3.
        (c) Invoke $\text{PATH-PLANNER}(S_g(q_k), q_k, q)$. If no collision-free path exists, then discard $q$ and return to
            Step 3; else let $L(q)$ be the length of the computed path.
        (d) Compute the visibility gain $A(q)$ and add $q$ to $\mathcal{N}_{sam}$.
4.  Select $q_{k+1}$ to be the candidate in $\mathcal{N}_{sam}$ that maximizes $g(q) = A(q) \exp(-\lambda L(q))$.
```

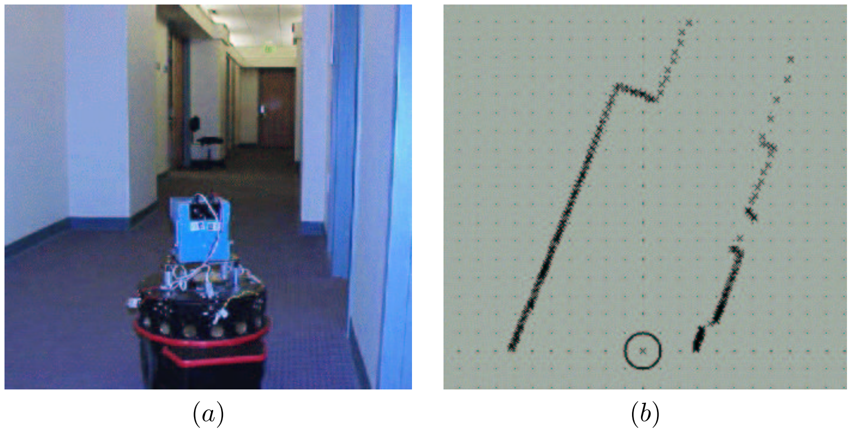


Figure 5: Range sensing: (a) scene; (b) points measured by range sensor.

Part II: Experimental System

In this second part we describe the implementation of an experimental map-building robotic system. This leads us to describe a number of operations and implementation details that were not covered in Part I.

5 Construction of 2-D Layouts

Our robot is equipped with a polar range sensor from Sick Optic-Electronic. This sensor measures the distance between the sensor's center-point and the objects in the environment along several rays regularly spaced in a horizontal plane at a height h above the floor. The sensor driver converts these measurements into a list of points representing a cross-section of the environment with respect to the coordinate system attached to the sensor. Figure 5(b) shows such points for a 180-deg field of view, with 0.5-deg spacing between every two consecutive rays. We model the sensor using Definition 2.1. We add an angular parameter α to represent the sensor's limited field-of-view.

Our goal is to construct a polygonal layout of the environment from the sets of points captured by the range sensor at different locations. Polygonal models have several interesting characteristics. They can represent complex environments at any degree of precision. (Similarly, in computer graphics, curved surfaces are well represented by triangulated meshes.) Polygonal models also make it possible to efficiently compute geometric properties, such as areas and visibility regions. However, we should remark that representing a workspace as a polygonal region is not the same as saying that the workspace is polygonal.

The sequence of sensing positions are chosen by the NEXT-BEST-VIEW algorithm proposed in Section 4. The following steps are executed at each sensing position q: polyline generation, safe region computation, model alignment, model merging, and detection of small obstacles. These are described in detail below.

5.1 Polyline Generation

Let L be the list of points acquired by the sensor at q. L is transformed into a collection Π_l of polygonal lines called *polylines*. The polyline extraction algorithm operates in two steps: (1) group points of L into clusters; (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 (as in Part I) would capture a curve $r(\theta)$ instead of a sequence of points. This curve would be discontinuous at exactly the points where occlusions occur. For our sensor, discontinuities are detected using thresholds selected according to the sensor's accuracy.

The points in each cluster are fitted with a polyline so that every data point lies within a distance ϵ from a line segment, while minimizing the number of vertices in the polyline. The computation takes advantage

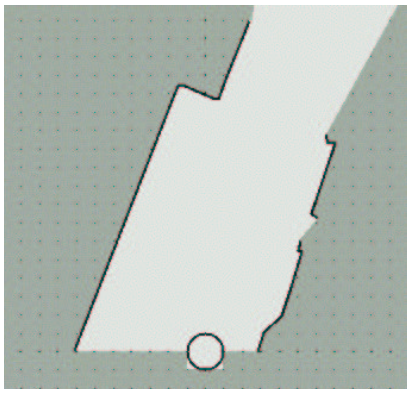
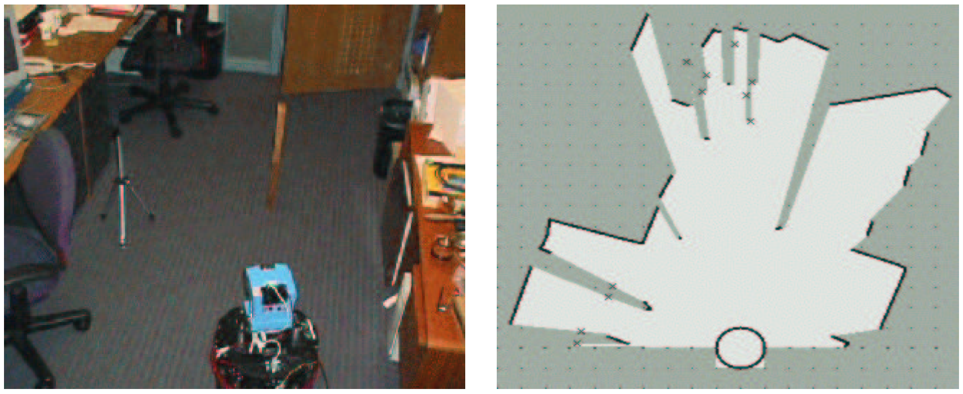


Figure 6: Polyline fit for the data of Figure 5(b).



(a) (b)

Figure 7: A more complicated example of a polyline fit.

of the fact that the data delivered by our polar sensor satisfy an ordering constraint along the noise-free θ-coordinate. By applying the mapping $u = \cos\theta/\sin\theta, v = 1/(r\sin\theta)$, the problem is transformed into a linear fit 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 [28]. By converting ϵ to the position-dependent error bound $e = \epsilon v\sqrt{(a^2 + b^2)}$ in the (u, v)-space, each data point in the (x, y)-space is guaranteed to be within ϵ from the computed polyline. The polyline fitting process also acts as a noise reduction filter.

Figure 6 shows three polylines generated from the data points of Figure 5(b). A more complicated example, in a cluttered office environment, is shown in Figure 7. The area in light color in (b) is the robot's visibility region under the unrestricted line-of-sight model.

5.2 Safe Region Computation

Once a polyline set $\Pi_l(q)$ has been extracted from the sensory data, a safe region $s_l(q)$ is computed according to the definition given in Section 2. This region is bounded by the polylines and the free curves joining the polyline endpoints. The free curves are composed of linear segments, circular arcs and spiral sections. Their construction is described in the proof of Theorem 3.1 given in the appendix. In our implementation, we approximate arcs and spiral sections with polygonal lines to simplify subsequent computations. So, the region $s_l(q)$ is computed as a polygon bounded by solid edges (derived from the polylines) and free edges (derived from the approximated free curves).

Figure 8 shows two local safe regions computed for the scene of Figure 7 for different values of the maximal range r_{max} and the incidence angle τ.

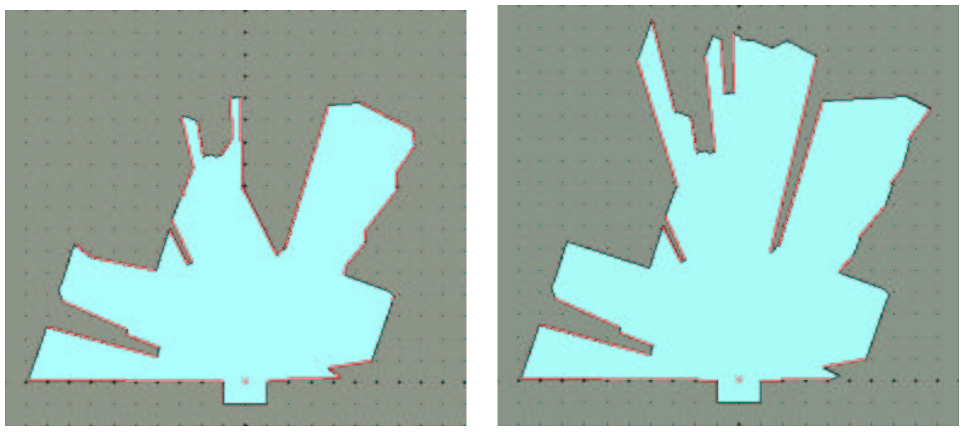

(a) (b)

Figure 8: Computed safe regions: (a) $r_{max} = 275$ cm and $\tau = 50$ deg; (b) $r_{max} = 550$ cm and $\tau = 85$ deg.

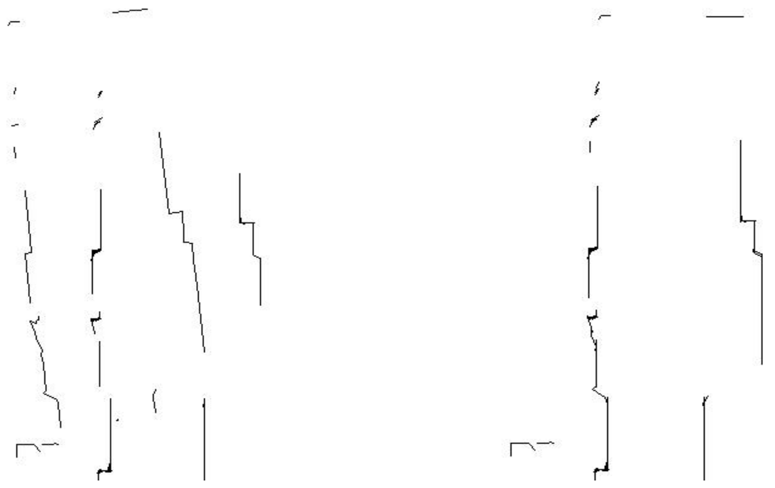

(a) (b)

Figure 9: (a) Unaligned polylines; (b) computed alignment.

5.3 Model Alignment

Recall that $M_g(q_{k-1}) = \langle \Pi_g(q_{k-1}), S_g(q_{k-1}) \rangle$ denotes the partial global model built at location q_{k-1}. Let the robot execute a new sensing operation at location q_k, extracting the local model $m_l(q_k) = \langle \Pi_l(q_k), s_l(q_k) \rangle$. A best match is then computed between the line segments in $\Pi_g(q_{k-1})$ and those in $\Pi_l(q_k)$, yielding a Euclidean transform aligning both sets of polylines. The matching algorithm implemented in our system is based on a technique previously used to discover and align substructures shared by 3-D molecular structures [13]. The algorithm selects pairs of line segments from Π_l at random. For each pair (u_1, u_2), it finds a pair of segments (v_1, v_2) in Π_g with the same relative 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. The algorithm then identifies the segments of Π_l and Π_g 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 the fit. These steps are performed for each pair of line segments sampled from Π_l, and in the end the transform with the best quality is retained. If all segments in Π_l are approximately parallel, the algorithm uses endpoints and odometric sensing to approximate the missing parameter of the transform. Alternative algorithms could be used here as well.

Figure 9 shows two sets of polylines before alignment (a) and after alignment (b).

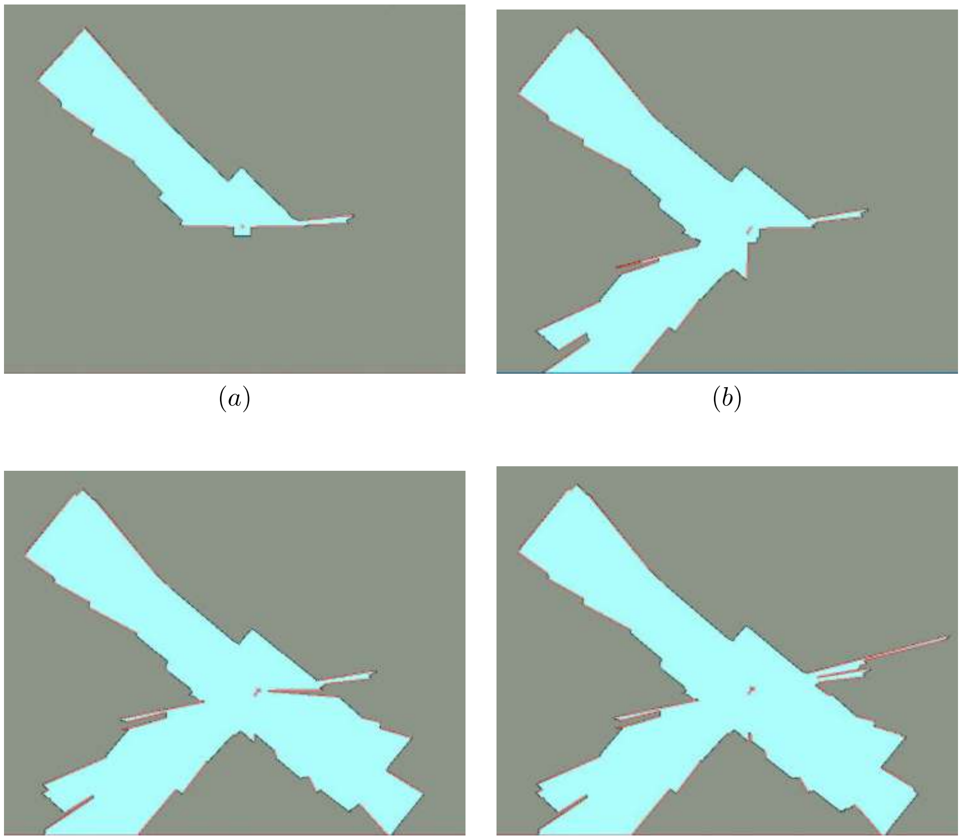

(a) (b) (c) (d)

Figure 10: Merging four local models acquired at the same robot position.

5.4 Model Merging

The selected transform T is applied to $S_g(q_{k-1})$ and the new global safe region $S_g(q_k)$ is computed as the union of $T(S_g(q_{k-1}))$ and $s_l(q_k)$. The solid edges bounding $S_g(q_k)$ form the new polyline set $\Pi_g(q_k)$. To avoid edge fragmentation, consecutive solid (respectively free) edges in the boundary of $\Pi_g(q_k)$ that are collinear within the sensor's resolution are fused into a single edge. The new model $M_g(q_k) = \langle \Pi_g(q_k), S_g(q_k) \rangle$ is represented in the coordinate system attached to the robot at its current position q_k. The 2-D layout is considered complete if no remaining free edge in the boundary of $S_g(q_k)$ is longer than a given threshold.

Figure 10 displays four partial models. The robot is at some location where it rotates to face four successive directions spaced by 90 deg. 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. Figure 10 illustrates both the model merging operation and how this operation compensates for small variations in the robot's position as it rotates to a new orientation. Moreover, it shows the artifice employed in our system to emulate omnidirectional sensing using a sensor with a 180-deg field of view.

5.5 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 set of polylines $\Pi_l(q)$ (see Figure 7). However, small errors in aligning polylines tend to eliminate such obstacles when the union of safe regions is computed. Other model-merging techniques could be used, but modeling small obstacles by closed contours

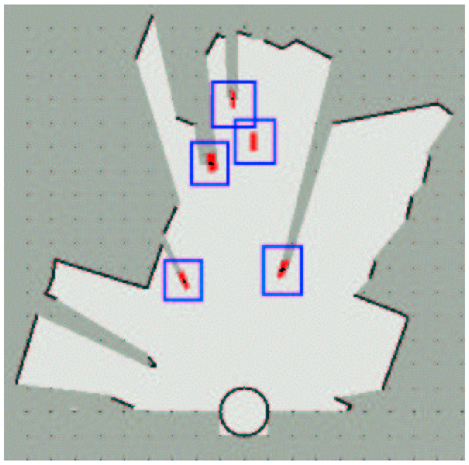


Figure 11: Small obstacles extracted from Figure 3(b).

is difficult and not very useful. In many instances, a map is more useful if small obstacles have been omitted, since the positions of such obstacles is likely to change over time. This leads us to proceed as follows. Small obstacles result into narrow, inward-pointing spikes in the contour of $s_l(q)$. These spikes can be automatically detected. The "apex" of each spike is a small isolated polyline, which is saved in a separate *small-object map*. Hence, the final model consists of a main layout (polylines and safe region) and a secondary map (small-object map). Figure 11 shows the small obstacles (apexes enclosed by square boxes) detected in the scan from Figure 7. These include a metal camera tripod, a narrow wooden bar, and a swivel chair.

Merging partial models by taking the union of safe regions has the added advantage of eliminating transient objects. Comparing edges in successive partial models allows detection of such objects, which can be recorded in a separate structure (in a way similar to small objects). However, this capability is not implemented in the system described here.

6 Implementation of NEXT-BEST-VIEW Algorithm

The NEXT-BEST-VIEW algorithm described in Section 4 is implemented here essentially unchanged. Steps 1 and 2 make use of techniques presented above (polyline extraction, safe region computation, and model alignment and merging). Implementation details for Steps 3 and 4 are described below.

6.1 Candidate Generation

The set $\mathcal{N}_{sam}$ of next-best-view candidates is generated as follows, where σ and ρ are two positive constants:

1. For each free edge e in the boundary of $S_g(q_k)$, pick $\sigma\times\text{length}[e]$ points along e uniformly at random. Group these points under the set $\mathcal{B}$.
2. For each $p \in \mathcal{B}$, compute the visibility region $\mathcal{V}(p)$ inside $S_g(q_k)$, upper-limited by r_{max}, and uniformly select $\rho\times\text{area}[\mathcal{V}(p)]$ random points inside $\mathcal{V}(p)$. Return the set $\mathcal{N}_{sam}$ of all these points.

6.2 Path Planning

We model our robot by a disc of radius R. Path planning between two points (the current position of the robot and a candidate NBV position) is done as follows:

1. Shrink $S_g(q_k)$ by R. (This step is performed only after the robot has moved to a new sensing position q_k, not every time a candidate q is evaluated.)

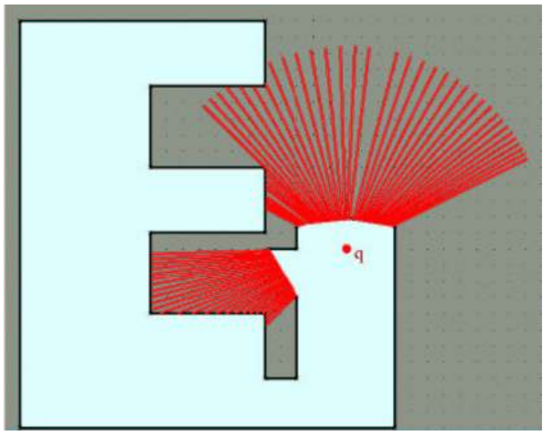


Figure 12: The potential information gain of a candidate q is the area $A(q)$ of the region outside the current safe region $S_g(q_k)$ that may be visible through the free edges; this area is estimated by casting rays from q.

2. Compute the shortest path from q_k to a q inside the shrunk region. This computation can be done using the visibility graph method [19]. The procedure can be accelerated by precomputing the *shortest-path map* from q_k — SPM(q_k). This map is a decomposition of the space into cells, such that the shortest paths to q_k from all points within a cell share the same sequence of obstacle vertices. (See [23] for a survey of methods.)

6.3 Evaluation of Candidates

Every NBV candidate q is evaluated using the score function proposed in Section 4.3 — i.e., $g(q) = A(q)\exp(-\lambda L(q))$. Here, $L(q)$ is the length of the shortest path connecting q_{k-1} to q. The parameter λ is set to 20 cm^{-1} in the implementation, a value that prevents the robot from oscillating back and forth between regions with similar visibility potential. The area $A(q)$ is computed using a discretized version of the technique outlined in Section 4.3 (see Figure 6.2):

1. Cast a fixed number of equally spaced rays from q.

2. For each ray:

   (a) Consider the segment between 0 and r_{max} from q. If this segment intersects a solid edge, eliminate the portion beyond the first intersection.

   (b) Compute the length ℓ of the portion of the remaining segment that falls outside $S_g(q_k)$.

3. Estimate $A(q)$ as the sum of all the computed ℓ's.

6.4 Example Runs

Figure 13 shows partial models generated at several iterations (0, 2, 6, and 19) during a run of the implemented NEXT-BEST-VIEW algorithm 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 score function, the robot fully explores the bottom-right corridor before moving to the left corridor. Figure 14 shows another series of snapshots for the same environment, the same initial position, but with greater r_{max} and τ. The motion strategy is simpler, requiring only 7 iterations.

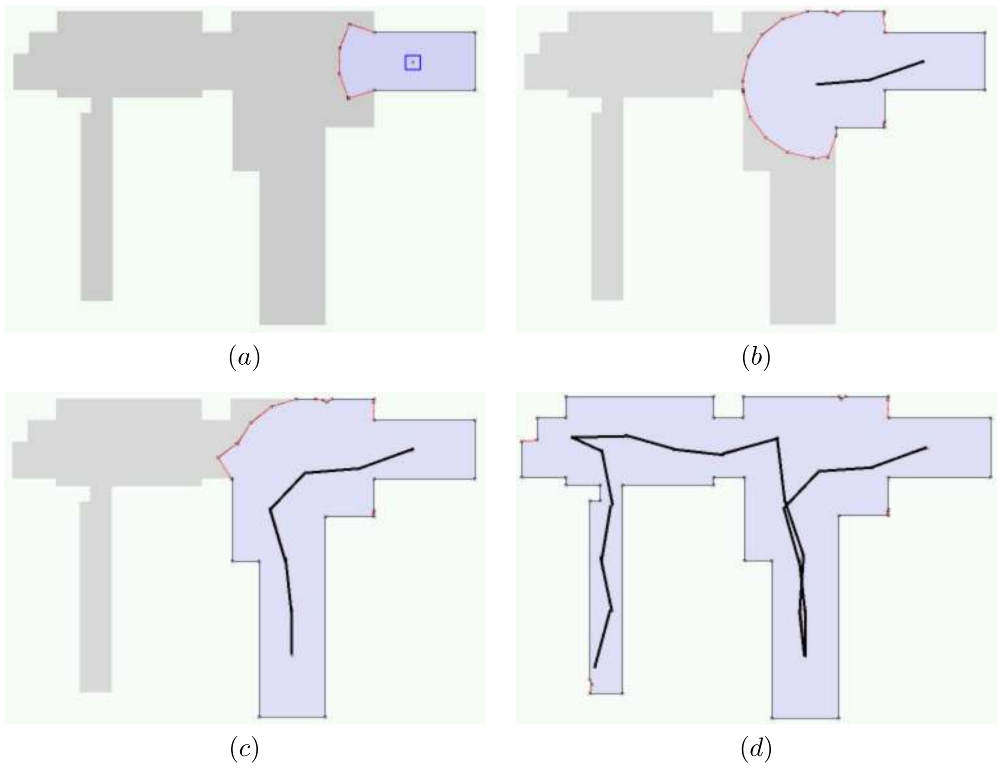


Figure 13: Example 1 of model construction in simulation.

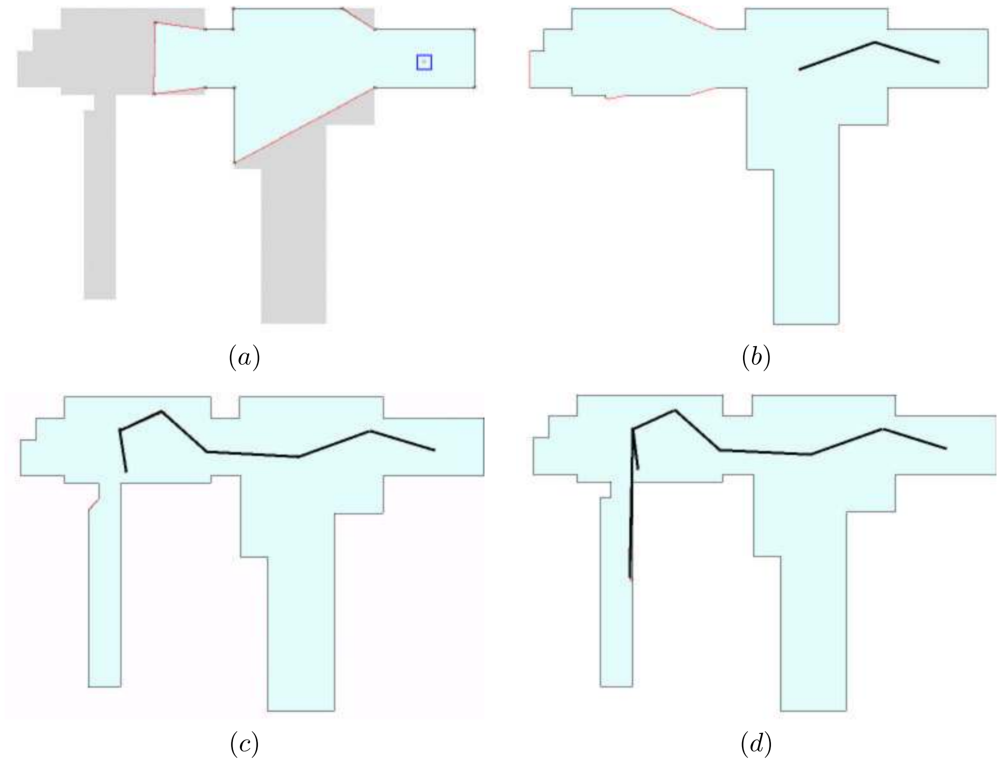


Figure 14: Example 2 of model construction in simulation.

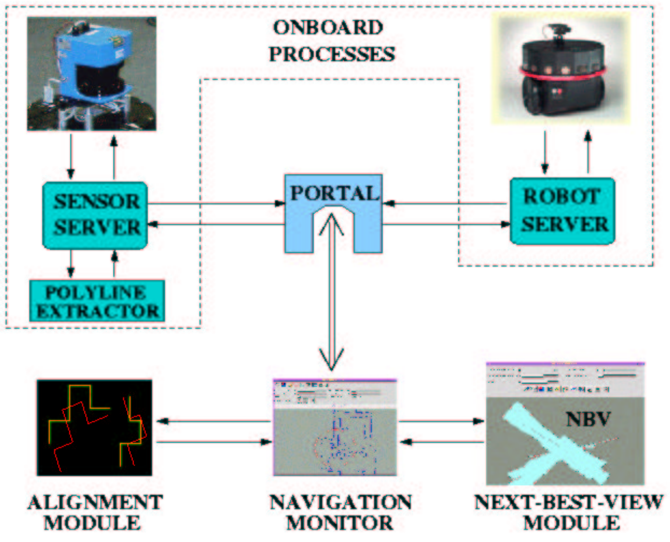


Figure 15: Interaction among the modules of the next-best-view system.

7 Experimental System

Our experimental system consists of a Nomadic SuperScout wheeled platform. We equipped this robot with a laser range sensor from Sick Optic-Electronic. The robot's on-board processor (a Pentium 233 MMX) acquires a 360-point 180-deg scan in 32 ms through a SeaLevel 500 Kbs PCI serial card. At each sensing location, a 360-deg view is obtained by taking 4 scans (Figure 10).

The on-board processor is connected to the local-area network via 2 Mbs radio-Ethernet. The NBV software and the navigation monitor run off-board in a Pentium II 450 MHz Dell computer. The software was written in C++ and uses geometric functions from the LEDA-3.8 library [22].

7.1 System Architecture

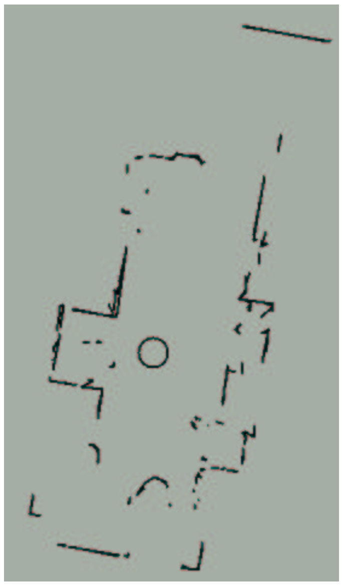
The software consists of several modules executing specialized functions, communicating with each other through TCP/IP socket connections under a client/server protocol. These modules are shown in Figure 15.

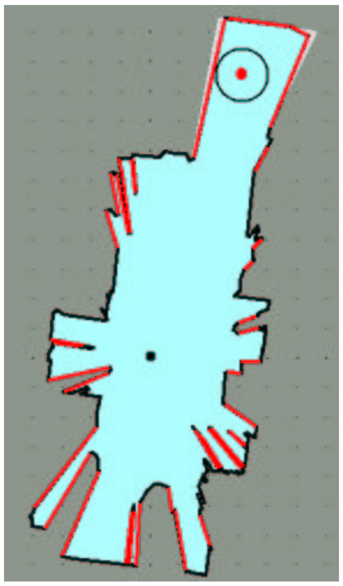
A *Sick sensor server* handles communications with the SeaLevel card. It allows clients to assume that they are connecting to a device resembling an ideal sensor. The server offers the following capabilities: (1) choice among 3 speed modes: 1, 5, and 30 scans/sec; (2) batch transmission of multiple scans on request; (3) scan averaging using the sensor's on-board electronics; (4) operation in continuous mode; and (5) real-time polyline fitting with 2.5-cm error bound.

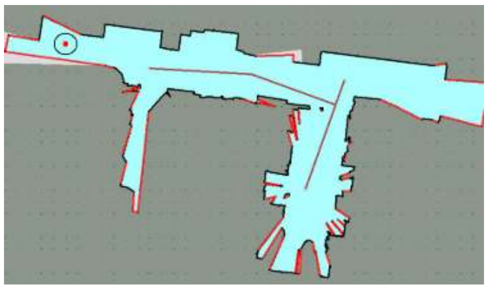
Since the polyline fitting technique is fast enough to be performed in real time under any speed mode, it is embedded into the server's code. This reduces the amount of data transmitted to the clients.

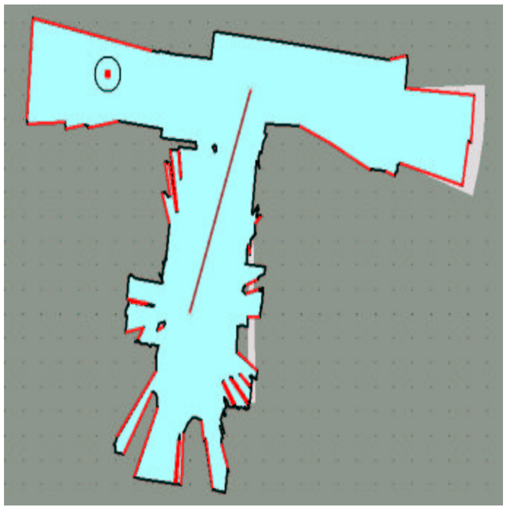
A *navigation monitor* allows a user to supervise the exploration process. The user may query the NBV module for the next position and/or the most recent environment model, or select the next sensing position manually. The user may also teleoperate the robot in continuous mode, receiving scan updates every 0.1 sec. The navigation module is also responsible for aligning new data with the previous model. The module first pre-aligns new data using the robot's odometry, and it afterwards invokes the model matching function. The computed transform is sent to the NBV module with each new scan.

Finally, the *NBV module* computes the next position given the current model of the environment. The model is updated every time a new scan is received.


(*a*)


(*b*)


(*d*)


(*c*)

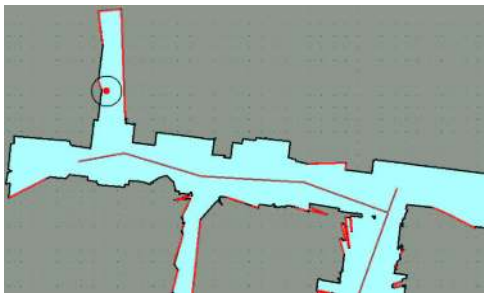

(*e*)

Figure 16: Experiments using the experimental system.

7.2 Example of Layout Construction

Figure 16 shows successive partial layouts built by the experimental system in our laboratory. The robot is initially placed in an office clustered with many small obstacles (chair and table legs, and cables). The sensor parameters are $r_{max} = 550$ cm and $\tau = 85$ deg. 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 module. The safe region is bounded by many free edges forming spikes, but the candidate evaluation function automatically detects that little additional space can be seen through such free edges. Consequently, the NBV module automatically selects the next sensing position near the exit door of the office. Figures (c)-(e) show the safe region after the robot has reached sensing positions 2, 4 and 6, respectively. At stage 6, the layout consists of the initial office, a second office (incompletely mapped), and two perpendicular corridors.

Another run is shown in Figure 17, where the robot mapped a larger section of our laboratory. The first 6 iterations are shown in (a). At this point the executed strategy resembles the one shown in Figure 16 due to similar initial conditions. At the corridor intersection, however, the robot faces more choices than in the previous example because an office door is open. Nevertheless, the NBV module opted to continue moving along a corridor, all the way into the other hall (b). Glass is transparent to the sensor's laser, so the robot failed to detect the glass door indicated in (b). At this point, the operator overrode the decision of the NBV module, which interpreted the lack of solid edge around the location of the glass door as the entry into a large unexplored area. Finally, in (c), the robot moved down the second hall until it reached the lab's lounge. The NBV module decided then to send the robot to explore this newly detected open area.

Figure 18 shows all the observed polylines after the robot completed a circuit around the lab. The polylines shown in light color (red) were captured at the last location, and the area inside the circle is the range of the last scan. Note the final mismatch. This discrepancy appears because every alignment transform is computed *locally* to align the current view $\Pi_l(q_k)$ with the current history $\Pi_g(q_{k-1})$. Once a transform has been computed, it is never revised. Thanks to the overlap constraint taken into account by the NBV module, the final mismatch is quite small (about 30 cms, while the mapped area was approximatively 25×15 mts. Reducing this mismatch further would require some form of global optimization over all the transforms computed so far (or at least a subset of them). Such an operation, which is not implemented in our system, can be found in recent SLAM systems [10, 11, 17]. In this sense, NBV and SLAM are complementary.

8 Conclusion

Our experiments, both in simulated and real environments, show that the NBV algorithm can considerably reduce the number of motions and sensing operations required for map building. This claim is difficult to quantify, as it would require more extensive comparisons between the strategies produced by our NBV module and strategies produced by other means (e.g., trained human operators). Moreover, our module can generate very different strategies, depending on the input parameters to the score function. However, our tests reveal so far that the NBV module produces strategies that cannot be easily out-done by a human operator. Our system also demonstrates that polygonal maps are feasible representations for indoor mapping tasks. As it was argued in Section 5, polygons offer significant advantages over other representations.

The most obvious limitation of our map-building system is that it only builds a cross-section of the environment at a 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 article would remain applicable without significant changes.

A more fundamental limitation of our implementation is the lack of error-recovery capabilities. Any serious error in polyline extraction or during image alignment can result in a completely unacceptable model. For that reason, the design of the experimental system is very conservative. For instance, often many points given by the sensor are discarded to avoid generating incorrect polylines. This may lead the robot to sense the same part of an environment several times, hence producing a longer motion path. Moreover, the image-alignment function always runs under user supervision to avert serious registration errors.

Matching transforms are computed locally to align the global model with the local model generated at the robot's current location. Once a transform has been computed, it is never revised. This has its

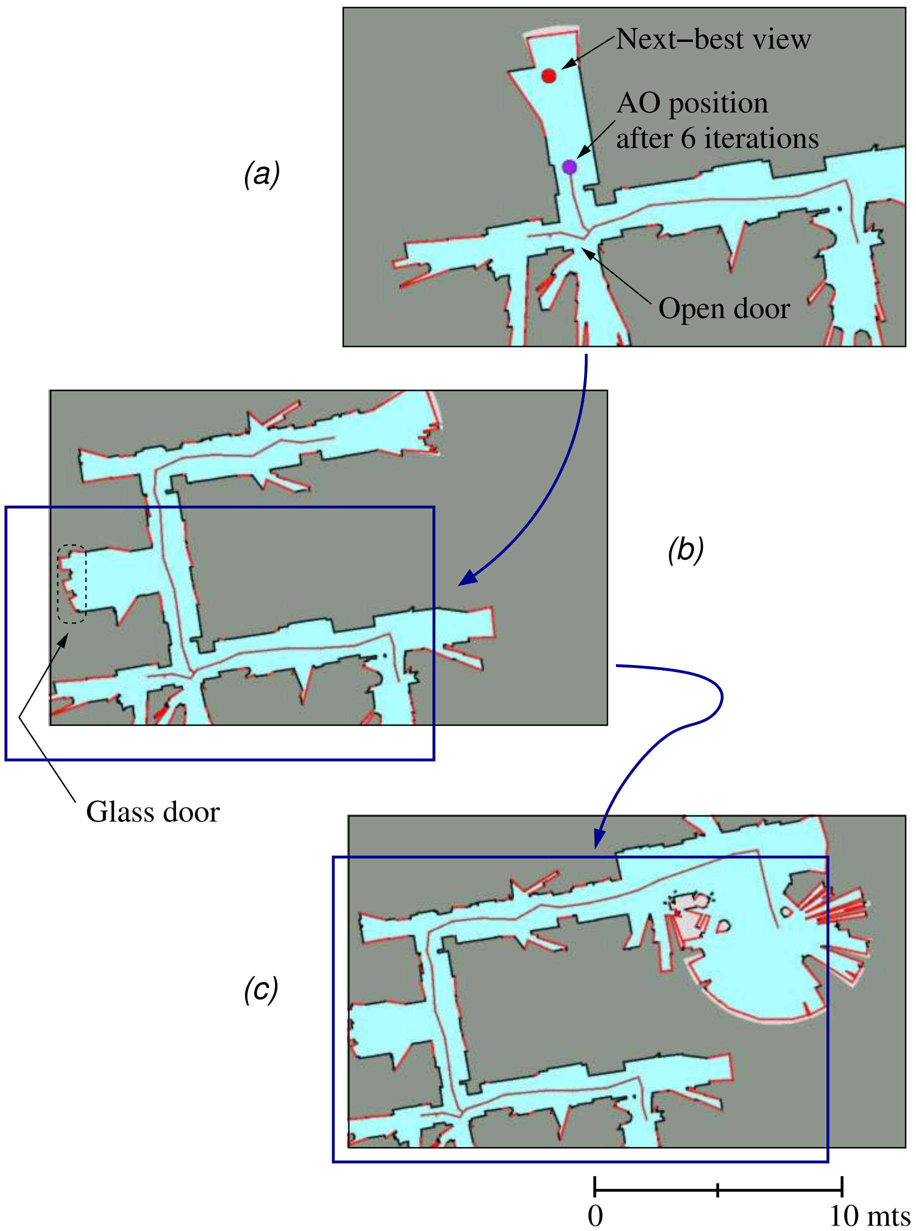


Figure 17: A run around the Robotics Laboratory at Stanford University (Gates building, wing 1-A): (a) map built after 6 iterations; (b) the robot has moved into a second hall, after the user overrode a decision of the NBV module to move toward a glass door; (c) the robot maps the second hall until it reached the large open area, which the NBV module decided to explore next.

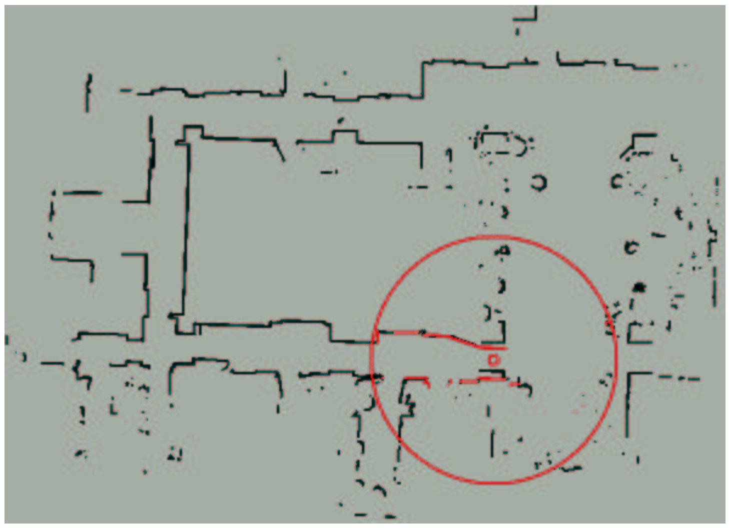


Figure 18: Observed polylines after the robot has completed a tour around our Lab. The polylines shown in light color (red) were captured at the last location of the robot. There is a slight mismatch with polylines sensed at an early stage of the tour.

disadvantages. In corridors bounded by parallel featureless walls, line matching only corrects positioning errors in the direction perpendicular to the walls. Odometry can be used, but imprecision in the direction parallel to the walls grows bigger with distance. A possible solution to this problem is to track successive local models and transformations to enable the periodic optimization of a global matching criterion, especially after the robot has completed a long loop. This could be achieved by including some recent SLAM techniques in our system. For large environments such global optimization would produce more precise layouts, and solve mismatches like the one shown in Figure 18. More generally, we believe that SLAM and NBV solve two distinct and complementary aspects of the same map-building problem. Future systems should combine both SLAM and NBV techniques.

Finally, the current system should handle multiple robots with relatively minor changes. If a team composed of N robots is available, and their relative positions are known, a single model can be generated from all the captured scans. A central NBV planner then computes the set of N positions that stations the team for the aggregated next-best view. However, when the relative positions of the robots are not known, the problem becomes considerably more difficult. In this case, the robots act independently (distributed planning), and perhaps communicate only sporadically. The techniques presented in this article would then have to be revised and extended in order to cover this case.

Acknowledgments: This work was funded by DARPA/Army contract DAAE07-98-L027, ARO MURI grant DAAH04-96-1-007, NSF grant IIS-9619625, and a gift from Honda R&D, Americas. We also wish to thank L. Guibas, T.M. Murali, R. Murrieta and S. Thrun, whose contributions enriched the ideas presented in this article, and S. Yao and E. Mao for their assistance during the implementation of the robot system.

References

[1] J.E. Banta, Y. Zhien, X.Z. Wang, G. Zhang, M.T. Smith, and M.A. Abidi. A "next-best-view" algorithm for three-dimensional scene reconstruction using range images. In *SPIE*, volume 2588, pages 418–29, 1995.

[2] A.J. Briggs and B.R. Donald. Automatic sensor configuration for task-directed planning. In *IEEE Int. Conf. on Robotics and Automation*, pages 1345–1350, 1994.

[3] R. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *IEEE Int. Conf. Robotics & Automation*, pages 476–481, 2000.

[4] J.A. Castellanos and J.D. Tardos. *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach*. Kluwer Academic Pub., Boston, MA, 2000.

[5] R. Chatila. Mobile robot navigation: Space modeling and decisional processes. In *Third Int. Symp. on Robotics Research*, pages 373–378, 1985.

[6] R. Chatila and J.P. Laumond. Position referencing and consistent world modeling for mobile robots. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 138–143, 1985.

[7] H. Choset and J. Burdick. Sensor based motion planning: The hierarchical generalized voronoi diagram. In J.-P. Laumond and M. Overmars, editors, *Proc. 2nd Workshop on Algorithmic Foundations of Robotics*. A.K. Peters, Wellesley, MA, 1996.

[8] C. I. Conolly. The determination of next best views. In *IEEE Int. Conf. on Robotics and Automation*, pages 432–435, 1985.

[9] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. ACM SIGGRAPH*, pages 303–312, August 1996.

[10] M.W.M.G. Dissanayake, P. Newman, H.F. Durrant-Whyte, S. Clark, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Tr. Robotics & Automation*, 17(3):229–241, 2001.

[11] H.F. Durrant-Whyte, M.W.M.G. Dissanayake, and P.W. Gibbens. Toward deployment of large scale simultaneous localisation and map building (slam) systems. In J. Hollerbach and D. Koditschek, editors, *Robotics Research - The Ninth Int. Symp.*, pages 161–167. Springer, New York, NY, 2000.

[12] A. Elfes. Sonar-based real world mapping and navigation. *IEEE J. Robotics and Automation*, RA-3(3):249–265, 1987.

[13] P.W. Finn, L.E. Kavraki, J.C. Latombe, R. Motwani, C. Shelton, S. Venkatasubramanian, and A. Yao. Rapid: Randomized pharmacophore identification for drug design. *J. of Comp. Geometry: Theory and Applications*, 10:263–272, 1998.

[14] H. González-Banos, A. Efrat, J.C. Latombe, E. Mao, and T.M. Murali. Planning robot motion strategies for efficient model construction. In J. Hollerbach and D. Koditschek, editors, *Robotics Research - The Ninth Int. Symp.*, Salt Lake City, UT, 1999. Springer-Verlag.

[15] H. González-Banos, L.J. Guibas, J.C. Latombe, S.M. LaValle, D. Lin, R. Motwani, and C. Tomasi. Motion planning with visibility constraints. In Y. Shirai and S. Hirose, editors, *Robotics Research - The 8th Int. Symp.*, pages 95–101, 1998.

[16] H. González-Banos and J.C. Latombe. Planning robot motions for range-image acquisition and automatic 3d model construction. In *Proc. AAAI Fall Symposium Series, Integrated Planning for Autonomous Agent Architectures*, Orlando, Florida, October 23-25 1998. AAAI Press.

[17] J. Guivant and E. Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Tr. Robotics & Automation*, 17(3):242–257, 2001.

[18] B. Kuipers, R. Froom, W.K. Lee, and D. Pierce. The semantic hierarchy in robot learning. In J. Connell and S. Mahadevan, editors, *Robot Learning*. Kluwer Academic Publishers, Boston, MA, 1993.

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

[20] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, 1991.

[21] J. Maver and R. Bajcsy. Occlusions as a guide for planning the next view. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(5):417–433, May 1993.

[22] K. Mehlhorn and St. Nahër. *LEDA: A Platform of Combinatorial and Geometric Computing.* Cambridge University Press, Cambridge, UK, 1999.

[23] J.S.B. Mitchell. Shortest paths and networks. In J.E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 445–466. CRC Press, Boca Raton, FL, 1997.

[24] P. Moutarlier and R. Chatila. Stochastic multisensory data fusion for mobile robot location and environment modeling. In H. Miura and S. Arimoto, editors, *Robotics Research - The 5th Int. Symp.*, pages 85–94. MIT Press, Cambridge, MA, 1989.

[25] J. O'Rourke. Visibility. In J.E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 467–479. CRC Press, Boca Raton, FL, 1997.

[26] R. Pito. A solution to the next best view problem for automated cad model acquisition of free-form objects using range cameras. Technical Report 95-23, GRASP Lab, U. of Pennsylvania, May 1995.

[27] R. Pito. A sensor based solution to the next best view problem. In *Proc. IEEE 13th Int. Conf. on Pattern Recognition*, volume 1, pages 941–5, 1996.

[28] W.H. Press, S.A. Teukolsky, W.T. Vettering, and B.P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1994.

[29] T. Shermer. Recent results in art galleries. *Proc. IEEE*, 80(9):1384–1399, September 1992.

[30] R. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *Int. J. of Robotics Research*, 5(4), 1987.

[31] S. Teller. Automated urban model acquisition: Project rationale and status. In *Proc. 1998 DARPA Image Understanding Workshop*, pages 455–462. DARPA, 1998.

[32] S. Thrun. An online mapping algorithm for teams of mobile robots. *Int. J. of Robotics Research*, 20(5):335–363, 2001.

[33] Jorge Urrutia. Art gallery and illumination problems. In J. R. Sack and J. Urrutia, editors, *Hanbook on Computational Geometry*, pages 387–434. Elsevier Science Publishers, 1997.

[34] L. Wixson. Viewpoint selection for visual search. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 800–805, 1994.

A The Complexity of a Free Curve is O(1)

This appendix demonstrates that the complexity of a free curve is constant. Specifically, we provide here the proof for Theorem 3.1:

Theorem 3.1 (Free Curves) *Let $r_2(\theta; a_2, b_2)$ succeed $r_1(\theta; a_1, b_1)$ in the output list Π of a sensor operating under Definition 2.2 and located at the origin. If $\partial\mathcal{W}$ is continuously differentiable, then the free curve $f(\theta; b_1, a_2)$ connecting r_1 to r_2 consists of at most three pieces. Each piece is either a line segment, a circular arc, or a section of a logarithmic spiral of the form $r = r_o \exp(\pm\lambda\theta)$ (where r_o is a constant and $\lambda = \tan\tau$).*

In order to prove this claim we need the following lemma:

Lemma A.1 (Unobserved Obstacles) *Let $r_2(\theta; a_2, b_2)$ succeeds $r_1(\theta; a_1, b_1)$ in the list Π. Let C be some obstacle, and suppose that neither r_1 nor r_2 are part of the boundary of C (i.e., C is disjoint from r_1 and r_2). If $\partial\mathcal{W}$ is continuously differentiable, then no portion of C lies within a distance r_{max} from the origin in the polar interval $b_1 < \theta < a_2$.*

Proof: Suppose the lemma is not true — that is, there is a portion of C within r_{max} of the origin inside the polar interval (b_1, a_2). Let p be the closest point to the origin in the boundary of C. Because $\partial\mathcal{W}$ is differentiable, the normal of $\partial\mathcal{W}$ at p points toward the origin. Therefore, p and its vicinity should have been observed. The vicinity of p must then be part of an element of Π. But this contradicts our assumption that r_2 succeeds r_1 and that C is disjoint from r_1 and r_2. $\square$

The consequence of Lemma A.1 is that if there exists an obstacle (or a portion of an obstacle) within the sensor's range inside the polar interval (b_1, a_2), then r_1 and/or r_2 represent a portion of this obstacle's boundary. In other words, in order to construct the worst-case scenario in the polar sector (b_1, a_2), we can assume that the workspace has no holes, and consider r_1 and r_2 as boundary sections of the same obstacle.

From here on, let $\beta = a_2 - b_1$, $\rho_1 = r_1(b_1)$ and $\rho_2 = r_2(a_2)$; and let l_1 and l_2 denote the rays connecting the origin with point $p_1 = (\rho_1, b_1)$ and point $p_2 = (\rho_2, a_2)$, respectively.

Each endpoint of a curve in Π represents one of the following events: the sensor line-of-sight was occluded (denoted as case {o}), the range constraint was exceeded (case {e}), or the incidence constraint was exceeded (case {v}). To join p_1 with p_2 there are a total of 6 distinct cases: {v,v}, {v,o}, {v,e}, {e,e}, {o,o} and {e,o}. The cases {o,e}, {o,v} and {e,v} are mirror images of other cases.

Case {v,v}: The incidence constraint was exceeded immediately after $\theta = b_1$ and immediately before $\theta = a_2$. Therefore, the normal to $\partial\mathcal{W}$ just after r_1 and just before r_2 is oriented at an angle larger than τ with respect to the sensor. Suppose that the boundary $\partial\mathcal{W}$ continues after r_1 with its surface normal constantly oriented at exactly an angle τ with respect to the sensor's line-of-sight. This curve in polar coordinates satisfies the following relations:

$$\boldsymbol{n} \;\doteq\; -r\,\delta\theta\;\hat{\boldsymbol{e}}_r\; + \delta r\;\hat{\boldsymbol{e}}_\theta\;, \tag{2}$$

$$\boldsymbol{n}\cdot(-r\hat{\boldsymbol{e}}_r\;) \;=\; r|\boldsymbol{n}|\cos(\tau) \;\;\Longrightarrow\;\; \frac{1}{r}\frac{\delta r}{\delta\theta} \;=\; \pm\lambda,\ \text{with}\ \ \lambda \doteq \tan(\tau). \tag{3}$$

Hence, the curve's equation is $r = r_o \exp\left[\pm\lambda(\theta - \theta_o)\right]$, with $r_o = \rho_1$ and $\theta_o = b_1$. The equation now defines two spirals: a spiral s_1^+ growing counter-clockwise from p_1 (or shrinking clockwise), and a second spiral s_1^- shrinking counter-clockwise from p_1 (or growing clockwise). $\partial\mathcal{W}$ must continue from p_1 in the counter-clockwise direction either "above" s_1^+ or "below" s_1^-; otherwise, the incidence constraint would not have been violated.

Similarly, for the opposite end p_2, let $r_o = \rho_2$ and $\theta_o = a_2$. The solution to equation (3) now defines a spiral s_2^- growing clockwise from p_2 (or shrinking counter-clockwise), and a second spiral s_2^+ shrinking clockwise from p_2 (or growing counter-clockwise). $\partial\mathcal{W}$ must continue from p_2 in the clockwise direction either "above" s_2^- or "below" s_2^+.

Remark 1. $\partial\mathcal{W}$ cannot continue below s_1^- when $\rho_1 \exp(-\lambda\beta) < \rho_2$. In other words, $\partial\mathcal{W}$ cannot continue below s_1^- if this spiral curve cuts l_2 below the point p_2 (Figure 19(a)). To show this, suppose $\partial\mathcal{W}$ continues below s_1^-, which implies that $\partial\mathcal{W}$ bends toward the sensor immediately after r_1. We know that $\partial\mathcal{W}$ does not cross the origin, else nothing is visible under Definition 2.1 and Π would be empty. Hence, $\partial\mathcal{W}$ would have to bend outwards before cutting the ray l_2, otherwise r_2 will be occluded. Since $\partial\mathcal{W}$ is differentiable, there must then be a point p where the normal to $\partial\mathcal{W}$ points towards the origin. Because of Lemma A.1, this point p is not occluded by any other section of $\partial\mathcal{W}$ that is disjointed from r_1 and r_2. Therefore, the vicinity of p is a visible portion of $\partial\mathcal{W}$. This violates our assumption that r_2 succeeds r_1. Thus, when $\rho_1 \exp(-\lambda\beta) < \rho_2$, the first section of the curve f joining r_1 to r_2 coincides with s_1^+.

Remark 2. By symmetry, when $\rho_2 \exp(-\lambda\beta) < \rho_1$ (i.e., s_2^+ cuts l_1 below p_1), the last section of the curve f coincides with s_2^- (which grows clockwise from p_2).

The point p_2 may lie below the intersection of s_1^- with l_2, above the intersection of s_1^+ with l_2, or between both intersections. Likewise, the point p_1 may lie below the intersection of s_2^+ with l_1, above the intersection of s_2^- with l_1, or between both intersections. There are total of 9 combinations of events for case {v,v}, but only 3 of them are independent:

(a) s_1^- cuts l_2 above p_2. Thus, $\rho_1 \exp(-\lambda\beta) > \rho_2$, and this is equivalent to $\rho_2 \exp(\lambda\beta) < \rho_1$. That is, s_2^- cuts l_1 below p_1.

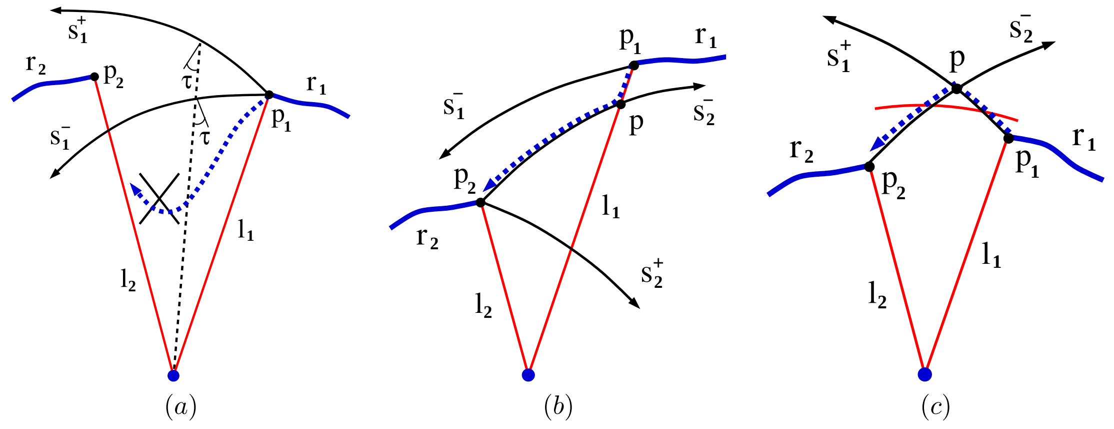


Figure 19: Example of a free-curve construction: (a) this situation is impossible; (b) in this case the free curve is composed of the segment joining p_1 with p and the spiral s_2^- joining p with p_2; (c) here the free curve is composed of the spiral s_1^+ joining p_1 with p and the spiral s_2^- joining p with p_2 (unless p is beyond range, in which case a circular arc of radius r_{max} is added).

(b) s_1^+ cuts l_2 below p_2. Thus, $\rho_1 \exp(\lambda\beta) < \rho_2$, and this is equivalent to $\rho_2 \exp(-\lambda\beta) > \rho_1$. That is, s_2^+ cuts l_1 above p_1.

(c) s_1^- cuts l_2 below p_2 and s_1^+ cuts l_2 above p_2. Thus, $\rho_1 \exp(-\lambda\beta) < \rho_2 < \rho_1 \exp(\lambda\beta)$, and this is equivalent to $\rho_2 \exp(-\lambda\beta) < \rho_1 < \rho_2 \exp(\lambda\beta)$. That is, s_2^+ cuts l_1 below p_1, and s_2^- cuts l_1 above p_1.

Let us analyze the first situation. $\rho_1 \exp(-\lambda\beta) > \rho_2$ is equivalent to $\rho_2 \exp(\lambda\beta) < \rho_1$, which in turn implies that $\rho_2 \exp(-\lambda\beta) < \rho_1$. In other words, both the clockwise-growing s_2^- and the clockwise-shrinking s_2^+ cut l_1 below p_1 (see Figure 19(b)). From Remark 2, the last section of the free curve f coincides with s_2^-. Let p be the intersection between s_2^- and l_1. The free curve f joining r_1 to r_2 is thus composed of the segment joining p_1 with p and the spiral s_2^- joining p with p_2.

A symmetric argument applies to the second situation, when $\rho_2 \exp(-\lambda\beta) > \rho_1$ (i.e., s_2^+ cuts l_1 above p_1), except that Remark 1 is used in this case.

The only remaining situation is (c). Here, $\rho_1 \exp(-\lambda\beta) < \rho_2$ and $\rho_2 \exp(-\lambda\beta) < \rho_1$. From Remarks 1 and 2, these inequalities imply that the first section of f coincides with s_1^+ while the last section of f coincides with s_2^-. Let p be the intersection of s_1^+ and s_2^-. If p is within r_{max}, then the free curve f is composed of the spiral s_1^+ joining p_1 with p and the spiral s_2^- joining p with p_2 (Figure 19(c)). Otherwise p is beyond range, and f is composed of a section of s_1^+, a circular arc of radius r_{max}, and a section of s_2^-.

Case {v,o}: As in the previous case, the curve r_1 was interrupted at $\theta = b_1$ because the incidence constraint was exceeded. The curve r_2, however, was interrupted at $\theta = a_2$ because a portion of $\partial\mathcal{W}$ blocked the sensor's line-of-sight. In order to produce the occlusion, $\partial\mathcal{W}$ must be tangent to l_2 at some point p_t below p_2. We know from Lemma A.1 that the portion of $\partial\mathcal{W}$ producing the occlusion cannot be disjointed from r_1. Thus, p_t is part of the same curve as r_1.

$\partial\mathcal{W}$ cannot continue from r_1 below s_1^-. To show this, suppose $\partial\mathcal{W}$ continues below s_1^-. This implies that $\partial\mathcal{W}$ bends toward the sensor immediately after r_1. But to cause the occlusion, $\partial\mathcal{W}$ has to bend outwards before it reaches the tangent point p_t. Since $\partial\mathcal{W}$ is differentiable, there must be a point where the normal to $\partial\mathcal{W}$ points towards the origin. But we already know that this violates our assumption that r_2 succeeds r_1.

Given that $\partial\mathcal{W}$ cannot continue from r_1 below s_1^-, then $\partial\mathcal{W}$ must continue above s_1^+.

For case {v,o}, it is always true that s_1^+ cuts the ray l_2 below p_2 at some point p. Otherwise, it will be impossible to produce the occlusion at p_t, because $\partial\mathcal{W}$ continues from r_1 above s_1^+. Thus, f is composed of the spiral s_1^+ joining p_1 with p, and the segment joining p with p_2.

Case {v,e}: As before, the incidence constraint was exceeded at $\theta = b_1$. But the curve r_2 was interrupted because the range constraint was exceeded at $\theta = a_2$. That is, $\rho_2 = r_{max}$.

$\rho_1 < r_{max}$ because the point p_1 is within range, which implies that $\rho_1 \exp(-\lambda\beta) < \rho_2$ since $\rho_2 = r_{max}$. This is exactly the situation described in Remark 1 for case {v,v}. Thus, $\partial\mathcal{W}$ cannot continue below s_1^-, and the first section of f coincides with s_1^+.

If s_1^+ cuts the ray l_2 below p_2 at some point p, then f is composed of the spiral s_1^+ joining p_1 with p, and the segment joining p with p_2. Otherwise p is beyond range, and f is composed of a section of s_1^+ and a circular arc of radius r_{max}.

Case {e,e}: This case is trivial. The free curve is a circular arc connecting p_1 with p_2.

Cases {o,o} and {e,o}: These cases are impossible because of Lemma A.1. In case {o,o}, both r_1 and r_2 are occluded. We know from Lemma A.1 that the portion of $\partial\mathcal{W}$ occluding r_2 cannot be disjointed from r_1, and that the portion of $\partial\mathcal{W}$ occluding r_1 cannot be disjointed from r_2. But this situation is impossible.

For the case {e,o}, $\partial\mathcal{W}$ must be tangent to l_2 at some point p_t below p_2. But $\partial\mathcal{W}$ continues after r_1 beyond the maximum range. Therefore, $\partial\mathcal{W}$ has to bend toward the sensor to fall back within range, and then bend outwards before it reaches the tangent point p_t. Again, this violates our assumption that r_2 succeeds r_1, because $\partial\mathcal{W}$ is differentiable and there must be a point where the normal to $\partial\mathcal{W}$ points towards the origin. Hence, this case is impossible.

We have accounted for all possible cases. This concludes our proof of Theorem 3.1. □