Robot Navigation for Automatic Model Construction using Safe Regions

Héctor González-Baños Jean-Claude Latombe

Department of Computer Science
Stanford University, Stanford, CA 94305, USA
e-mail:{hhg,latombe}@robotics.stanford.edu

Abstract: Automatic model construction is a core problem in mobile robotics. To solve this task efficiently, we need a motion strategy to guide a robot equipped with a range sensor through a sequence of "good" observations. Such a strategy is generated by an algorithm that repeatedly computes locations where the robot must perform the next sensing operation. This is called the next-best view problem. In practice, however, several other considerations must be taken into account. Of these, two stand out as decisive. One is the problem of safe navigation given incomplete knowledge about the robot surroundings. The second one is the issue of guaranteeing the alignment of multiple views, closely related to the problem of robot self-localization. The concept of *safe region* proposed in this paper makes it possible to simultaneously address both problems.

1. Introduction

Automatic model construction is a fundamental task in mobile robotics [1]. The basic problem is easy to formulate: 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. Despite this simple formulation, the problem is difficult to solve in practice. First, there is the problem of choosing an adequate representation of the environment — e.g., topological maps [2], polygonal layouts [1], occupancy grids [4], 3-D models [12], or feature-based maps [6]. Second, the representation must be extracted from imperfect sensor readings — e.g., depth readings from range-sensors may fluctuate due to changes in surface textures [3], different sets of 3-D scans must be zippered [13], and captured images must be aligned and registered [11]. Finally, if the system is truly automatic, the robot must decide on its own the necessary motions to construct the model [5].

Past research in model construction has mainly focused on developing techniques for extracting relevant features (e.g., edges, corners) from raw sensor data, and on integrating these into a single and consistent model. There is also prior research on the computation of sensor motions, mostly on finding the *next-best view* (NBV) [3, 11]: Where should the sensor be placed for the next sensing operation? Typically, a model is first built by combining images taken from a few distributed viewpoints. The resulting model usually contains gaps. An NBV technique is then used to select additional viewpoints that will provide the data needed to fill the remaining gaps.

Traditional NBV approaches are not suitable for mobile robotics. One reason is that most of the existing NBV techniques have been designed for systems that build a 3-D model of a relatively small object using a precise range sensor moving around the specimen. Collisions, however, are not a major issue for sensors that are mechanically constrained to operate outside the convex hull of the scene. In robotic applications, by contrast, the sensor navigates within the convex hull of the scene. Therefore, *safe navigation* considerations must always be taken into account when computing the next-best view for a robot map builder.

The second reason why most existing NBV techniques cannot be applied to mobile robots is that very few of the proposed approaches explicitly consider image-registration issues (one exception is the sensor-based technique presented in [11]). Localization problems particularly affect mobile sensors, and image registration becomes paramount when it is the means by which a mobile robot re-localizes itself (this is the so-called *simultaneous localization and map building* problem) [9, 7]. Although many image-registration techniques can be found in the literature, all require that each new image significantly overlaps with portions of the environment seen by the robot at previous sensing locations [9].

The system presented in [5] deals with the safe navigation and localization problems by applying the concept of *safe region* and the NBV algorithm introduced in this paper. With safe regions, it is possible to iteratively build a map by executing union operations over successive views, and use this map for motion planning. Moreover, safe regions can be used to estimate the overlap between future views and the current global map, and to compute locations that could potentially see unexplored areas.

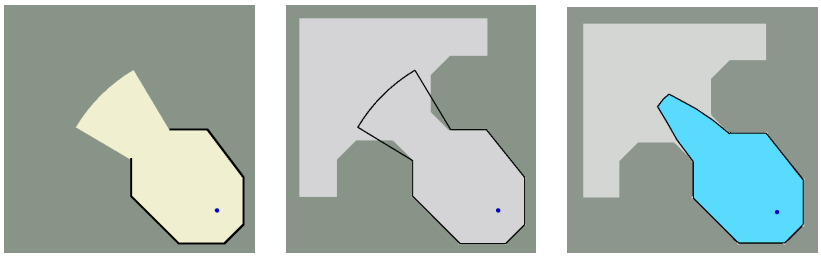
The work in [5] is mainly about system integration and proof of concept. Instead, this paper focuses on the formal definition of a safe region (Section 2), and describes how to compute such region from sensor data (Section 3). An NBV algorithm based on safe regions is outlined in Section 5, and Section 6 describes an experimental run using our system.

2. Definition of Safe Regions

Suppose 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. Because all visual sensors are limited in range, we assume that objects can only be detected within a distance d_M. In addition, most range-finders cannot reliably detect surfaces oriented at grazing angles with respect to the sensor. Hence, we also assume that surface points that do not satisfy the sensor's incidence constraint cannot be reliably detected by the sensor. Formally, our visibility model is the following:

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

1. Line of sight constraint*: The segment from q to w doesn't intersect $\partial\mathcal{W}$.*
2. Range constraint*: $d(q,w) \leq d_M$, where $d(q,w)$ is the Euclidean distance between q and w, and $d_M > 0$ 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 w, $\boldsymbol{v}$ is oriented from w to q, and $\tau \in [0, \pi/2]$ is an input constant.*



(a) (b) (c)

Figure 1. Effect of incidence on safe regions.

Without any loss of generality we assume the sensor is located at the origin (the workspace can always be re-mapped to a reference frame centered on the sensor). The sensor's output is assumed to be as follows:

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

Given an observation Π made by the robot at a location q, we define the *local safe region* s_l at q as the largest region guaranteed to be free of obstacles. While range restrictions have an obvious impact on s_l, the effect of incidence is more subtle. In Figure $1(a)$, a sensor detects the surface contour shown in black. A naive approach may construct the region in light color (yellow) by joining the detected surfaces with the perimeter limit of the sensor, and consider this region free from obstacles. Because the sensor is unable to detect surfaces oriented at grazing angles, this region may be not be safe, as shown in (b). A true safe region is shown in (c), for an incidence constraint of $\tau = 70$ deg.

3. Computing Safe Regions

The region s_l is bounded by solid and free curves. A *solid curve* represents an observed section of $\partial\mathcal{W}$, and is contained in the list Π. Given two solid curves $\{r_1(\theta; a_1, b_1), r_2(\theta; a_2, b_2)\} \subseteq \Pi$, 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 sections is called a *free curve* if: (1) no undetected obstacle is contained in the polar region $b_1 < \theta < a_2$ bounded by f; and (2) this region is the largest possible.

The main goal of this section is to find the free curves that join each successive pair in Π in order to bound the region s_l. It turns out that the complexity of f is $\mathrm{O}(1)$. In fact, a free curve f can be described using no more than 3 function primitives:

Theorem 3.1 (Free Curves) *Suppose* $r_2(\theta; a_2, b_2)$ *succeeds* $r_1(\theta; a_1, b_1)$ *in the output list* Π*. 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.*

The rest of this section proves this claim. But first we need the following lemma:

Lemma 3.2 *Let $r_2(\theta; a_1, b_1)$ succeed $r_2(\theta; a_2, b_2)$ 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 d_M 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 d_M 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. □

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 joining the origin with $p_1 = (\rho_1, b_1)$ and $p_2 = (\rho_2, a_2)$, 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 at $\theta = b_1$ and $\theta = a_2$. Therefore, the normal to $\partial\mathcal{W}$ immediately after r_1, and immediately before r_2, is oriented at a grazing angle with respect to the sensor. Suppose that $\partial\mathcal{W}$ 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:

$$\boldsymbol{n} \;\doteq\; -r\,\delta\theta\,\hat{\boldsymbol{e}}_r \;+ \delta r\,\hat{\boldsymbol{e}}_\theta\,, \tag{1}$$

$$\boldsymbol{n}\cdot(-r\hat{\boldsymbol{e}}_r) \;=\; r|\boldsymbol{n}|\cos(\tau) \;\;\Longrightarrow\;\; \frac{1}{r}\frac{\delta r}{\delta\theta} \;=\; \pm\lambda,\ \text{with}\ \ \lambda \doteq \tan(\tau). \tag{2}$$

Hence, the curve's equation is $r = r_o \exp[\pm\lambda(\theta - \theta_o)]$, 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 counter-clockwise from p_1 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 (2) 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 clockwise from p_2 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 2(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 3.2, 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^+.

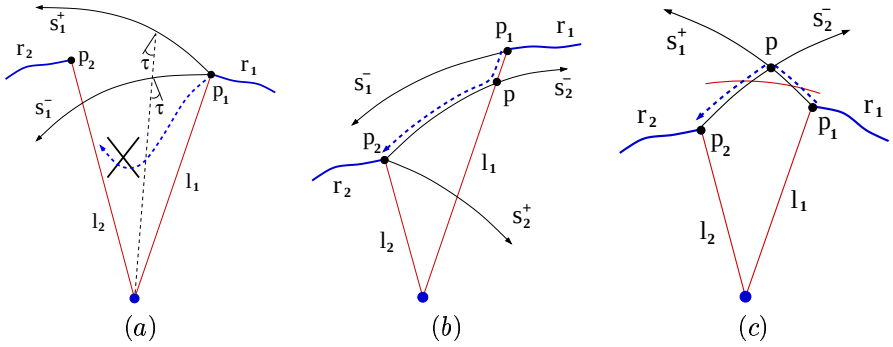


Figure 2. Example of a free-curve construction: (a) this situation is impossible; (b) in this case the free curve is composed of the segment joining p_1 with p and the spiral s_2^- joining p with p_2; (c) here the free curve is composed of the spiral s_1^+ joining p_1 with p and the spiral s_2^- joining p with p_2 (unless p is beyond range, in which case a circular arc of radius d_M is added).

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.

(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 2(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): $\rho_1 \exp(-\lambda\beta) < \rho_2$ 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 d_M, then f is composed of the spiral s_1^+ joining p_1 with p and the spiral s_2^- joining p with p_2 (Figure 2(c)). Otherwise p is beyond range, and f is composed of a section of s_1^+, a circular arc of radius d_M, and a section of s_2^-.

(a) (b)

Figure 3. Dealing with corners: (a) the normal to $\partial\mathcal{W}$ at a corner is generalized as the average of $\boldsymbol{n}^+$ and $\boldsymbol{n}^-$; (b) if we assume that a corner has "thickness", and is therefore detectable by the sensor, then any wedge-shaped object is visible if $\tau \geq 45$ deg.

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 3.2 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. Therefore, $\partial\mathcal{W}$ 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 r_2 was interrupted because the range constraint was exceeded at $\theta = a_2$. That is, $\rho_2 = d_M$.

The point p_1 is within range, hence $\rho_1 \exp(-\lambda\beta) < \rho_2$ because $\rho_2 = d_M$. This is exactly the situation described in Remark 1 of 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 d_M.

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

Cases {o,o} and {e,o}: The reader may verify that these cases are impossible by following the same line of reasoning used throughout this proof. We skip the details for lack of space.

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

4. Extracting Safe Regions from Real Sensor Data

The main practical problem with the theoretical results of the previous section is that the sensor output is usually a list of points, not a list of curves. Therefore, a pre-processing stage is needed to convert the raw data into the output list Π.

Let L be the list of points acquired by the sensor at q. L is transformed into a collection Π of polygonal lines called *polylines*. The 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 in $\mathcal{W}$. Clustering is done 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 of the fact that the data delivered by polar sensors satisfy an ordering constraint along the noise-free θ-coordinate axis. 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 algorithms exist to find polylines in (u, v)-space. We used a divide-and-conquer algorithm. Examples of our polyline-fit technique with real sensor data can be found in [5].

4.1. Corners

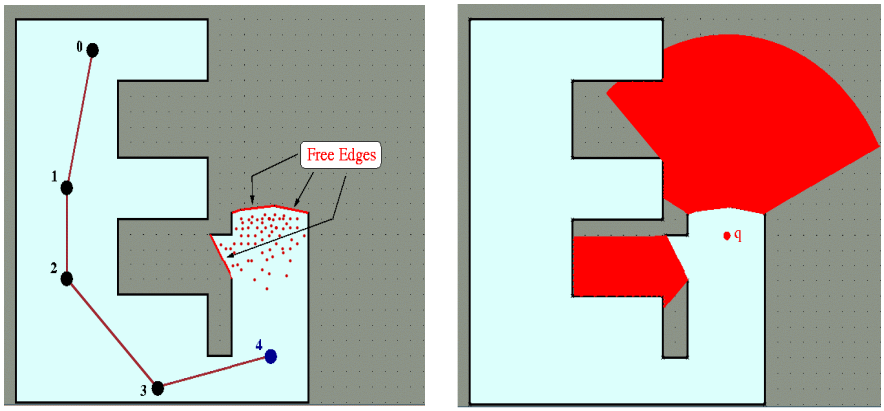
Corners pose a problem even under idealized conditions. Suppose the robot is surrounded by one or several wedge-shaped walls oriented toward the sensor. The sensor is then unable to see any of these wedges, and the safe region is empty. This is not a failure of our mathematical analysis, but a physical limitation of the sensor. This limitation was not taken into account by Definition 2.1, along with several others (e.g., that some surfaces could be perfect mirrors). We can only assume that the angle between any pair of incident walls is large enough such that at least one section at either side of the corner is visible to the sensor. Or that the corner itself is not sharp enough to remain undetected by the sensor (i.e., the corner has "thickness").

Under the above assumptions, we generalize the concept of a surface normal to include corners. The normal $\boldsymbol{n}$ to $\partial\mathcal{W}$ at a corner is the average of $\boldsymbol{n}^+$ and $\boldsymbol{n}^-$, where $\boldsymbol{n}^+$ and $\boldsymbol{n}^-$ are the normals to $\partial\mathcal{W}$ immediately after and before the corner (Figure $3(a)$). The corner is visible if the conditions of Definition 2.1 are satisfied for this generalized $\boldsymbol{n}$. That is, a corner behaves like any other point in $\partial\mathcal{W}$, as long as our hypotheses about $\partial\mathcal{W}$ hold true.

The system described in [5] expects corners to have thickness, and therefore to be detectable by the sensor. Under this supposition, it is easy to verify that any wedge-shaped object within range is visible if $\tau \geq 45$ deg (Figure $3(b)$).

5. A Next-Best View Algorithm

In a static environment, a safe region remains safe under the union operation. Hence, the layout model can be expanded 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 build 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 next-best-view position, the procedure first generates a set of potential candidates. Next, it evaluates each candidate according to both the expected gain of information that will be sensed at this position, and the motion cost required to move there. These steps are described below.



(a) (b)

Figure 4. Next-best view computation after 5 sensing operations have already taken place: (a) candidate generation; (b) evaluation of one candidate.

5.1. Model Alignment and Merging

Let $\langle \Pi_g(q_{k-1}), S_g(q_{k-1}) \rangle$ be the partial *global* model built at q_{k-1}. The term $S_g(q_{k-1})$ is the union of all 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 $\partial\mathcal{W}$. Let $\Pi_g(q_{k-1})$ be the list of solid curves in the boundary of $S_g(q_{k-1})$.

The robot performs a sensing operation once it moves into a new location q_k. From the local measurement $\Pi_l(q_k)$, we compute a local safe region $s_l(q_k)$ using the techniques from Section 3. Let $\langle \Pi_l(q_k), s_l(q_k) \rangle$ be the *local* model at q_k.

Suppose there exists an algorithm ALIGN that computes the transformation T aligning the line segments in $\Pi_l(q_k)$ with those in $\Pi_g(q_{k-1})$. We will *not* assume that this technique is perfect: ALIGN computes a correct T only when there is enough overlap between $\Pi_l(q_k)$ and $\Pi_g(q_{k-1})$.

Once T is 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 $\langle \Pi_g(q_k), S_g(q_k) \rangle$ is represented in a coordinate frame centered over the robot at its current position q_k.

5.2. Candidate Generation

The next location q_{k+1} must be contained inside $S_g(q_k)$. Otherwise, q_{k+1} would not be reachable from the current position q_k.[1]

We generate at random a number of possible NBV candidates in $S_g(q_k)$ within the vicinity of the free curves bounding $S_g(q_k)$ (Figure 4(a)). For each possible candidate q, we compute the total length $\zeta(S_g(q_k), q)$ of the non-free curves bounding $S_g(q_k)$ that are visible from q under Definition 2.1 (this operation is done using a line-sweep technique [10]). ζ is the measure of the expected overlap between a new image $\Pi_l(q)$ and the current list of solid curves $\Pi_g(q_k)$. If $\zeta(S_g(q_k), q)$ is greater than some threshold, then q is actually selected as an NBV candidate. This filtering stage ensures that the function ALIGN will successfully find a transform T.

[1] Strictly speaking, $S_g(q_k)$ must first be shrunk by the radius of the robot before computing a safe route.

5.3. Evaluation of candidates

To decide whether a position q in $S_g(q_k)$ is a good candidate for q_{k+1} we must estimate how much new information about the workspace we expect to obtain at q — i.e., q should potentially see large unexplored areas *through* the free boundary of $S_g(q_k)$.

The score of every NBV candidate q is given by the function $g(q) = A(q)\exp(-\lambda L(q,q_k))$, where λ is a positive constant, $L(q,q_k)$ is the length of the shortest path connecting q_k with q, and $A(q)$ is a measure of the unexplored area of the environment that may be visible from q (see next paragraph). q_{k+1} is selected as the sample q that maximizes the function $g(q)$. The factor λ weights the relative cost of motion with respect to visibility gains. $\lambda = 0$ implies that the map builder incurs no cost while moving, and the NBV planner is allowed to select new locations exclusively in terms of their potential visibility gain. $\lambda \gg 0$ implies that motion is so costly that locations close to q_k are preferred over distant ones, as long as they produce a marginal gain in visibility.

Computation of $\mathbf{A(q)}$ We measure the potential visibility gain of each candidate q as a function of the area $A(q)$ outside the current safe region that may be visible *through the free curves* bounding $S_g(q_k)$ (Figure 4(b)). For polygonal models, $A(q)$ can be computed by the same ray-sweep algorithm used to compute classic visibility regions [10], with the following modifications:

1. The sweeping ray may cross an arbitrary number of free edges before hitting a solid one. Therefore, the computation-time of the ray-sweep algorithm becomes $O(n\log(n) + n\,k_f)$, where k_f is the number of free edges bounding $S_g(q_k)$.
2. The resultant visible region is cropped to satisfy the range restrictions of the sensor. This operation can be done in $O(n\,k_f)$.

5.4. Termination Condition

If $S_g(q_k)$ contains no free curves, the 2-D layout is assumed to be complete; otherwise, $S_g(q_k)$ is passed to the next iteration of the mapping process. A weaker termination test is employed in practice: the length of any remaining free curve is smaller than a specified threshold.

5.5. Iterative Next-Best View Algorithm

The iterative NBV algorithm is summarized below:

Algorithm *Iterative Next-Best View*
Input: A new sensing position q_k and the local measurement $\Pi_l(q_k)$
An image alignment function $T = \text{ALIGN}(\Pi_l(q_k), \Pi_g(q_{k-1}))$
The number of samples m, and a weighting constant $\lambda > 0$
Output: A next-best view position q_{k+1}

1. Compute the local safe region $s_l(qk)$. Set the list of samples $\mathcal{N}_{sam} = \emptyset$.
2. Compute $T = \text{ALIGN}(\Pi_l(q_k), \Pi_g(q_{k-1}))$, and the union $S_g(q_k) = s_l(q_k) \bigcup T(S_g(q_{k-1}))$.
3. Repeat until the size of $\mathcal{N}_{sam}$ is greater or equal than m:
   (a) Randomly generate $q \in S_g(q_k)$ in the vicinity the free curves bounding $S_g(q_k)$.
   (b) If $\zeta(S_g(q_k), q)$ is below the requirements of ALIGN, discard q and repeat Step 3.
   (c) Compute $A(q)$ and $L(q, q_k)$. Add q to $\mathcal{N}_{sam}$ and repeat Step 3.
4. Select $q_{k+1} \in \mathcal{N}_{sam}$ maximizing $A(q)\exp(-\lambda L(q, q_k))$ as the next-best view.

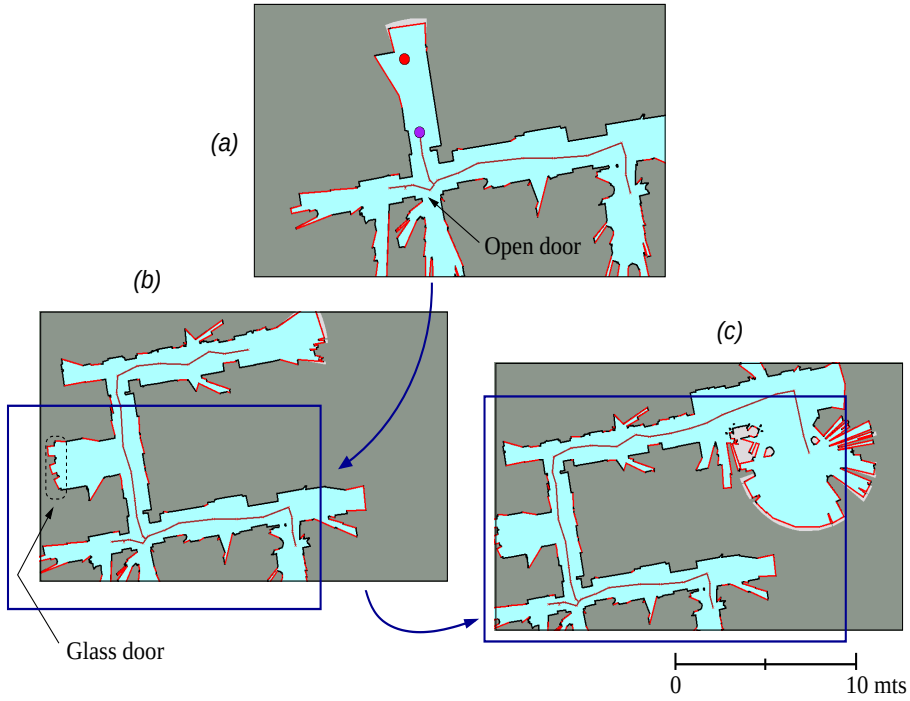


Figure 5. A run around the Robotics Lab. at Stanford University.

6. Experiments

The map-building system was implemented on a Nomadic SuperScout robot. The on-board computer is a Pentium 233 MMX, connected to the local-area network via 2 Mbs radio-Ethernet. The robot is equipped with a laser range sensor from Sick Optic Electronic which uses a time-of-flight technique to measure distances. The NBV planner runs off-board in a Pentium II 450 MHz Dell computer. The software was written in C++ and uses geometric functions from the LEDA library [8].

The sensor acquires 360 points in a single 180-deg scan request. A 360-deg view is obtained by taking 3 scans. The sensor readings where observed to be reliable within a range of 6.5 mts, at grazing angles not exceeding $\tau = 85$ deg. For the NBV planner, $\lambda = 20$ cm^{-1}, a value that prevents the robot from oscillating back and forth between regions with similar visibility gains.

An experimental run is shown in Figure 5. The robot mapped a section of the Robotics Lab. at Stanford U. The first 6 iterations are shown in (a). At the corridor intersection, the robot faces three choices, including going into an office. Nevertheless, the planner opted to continue moving along a corridor, all the way into the upper 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 planner, who interpreted the vicinity of the glass door as the threshold of an unexplored open area. Finally, in (c), the robot moved down the second hall until it reached the lab's lounge. The planner decided then to send the robot to explore this newly detected area.

7. Conclusion

Motion planning for model building applications has received little attention so far despite its potential to improve the efficiency of autonomous mapping. In this paper we introduced the concept of safe region, and described how it can be used to produce collision-free motions and next-best view locations under image-alignment considerations. Our research combines theoretical investigation of planning problems with simplified visibility models to produce algorithms that reach a compromise between algorithmic rigor and system practice. The result is a system able to construct models of realistic scenes.

Acknowledgments: This work was funded by DARPA/Army contract DAAE07-98-L027, ARO MURI grant DAAH04-96-1-007, and NSF grant IIS-9619625.

References

[1] 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.

[2] 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.

[3] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. ACM SIGGRAPH*, 1996.

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

[5] H. González-Banos, A. Efrat, J.C. Latombe, E. Mao, and T.M. Murali. Planning robot motion strategies for efficient model construction. In *Robotics Research - The Eight Int. Symp.*, Salt Lake City, UT, 1999. Final proceedings to appear.

[6] 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.

[7] J.J. Leonard and H.F. Durrant-Whyte. Stochastic multisensory data fusion for mobile robot location and environment modeling. In *Proc. IEEE Int. Conf. on Intelligent Robot Syst.*, 1991.

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

[9] 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.

[10] 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.

[11] 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.

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

[13] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proc. ACM SIGGRAPH*, pages 311–318, 1994.