Planning Robot Motion Strategies for Efficient Model Construction

Abstract

We consider the problem of building a 3D geometric model of an indoor environment by using range sensors mounted on a mobile robot. Our goal is to generate motion strategies that allow the robot to accomplish this task efficiently by avoiding unnecessary motions and sensing operations. We propose an approach to 3D model construction in which the robot first builds a 2D layout (horizontal cross-section) of the environment using an online next-best-view technique to decide where it should perform 2D sensing operations. The robot then applies an offline randomized art-gallery algorithm to the 2D layout in order to compute the locations where it should perform more expensive 3D sensing operations. Our approach takes into account sensing constraints (range, incidence) and deals with localization uncertainty; it also scales to multiple robots. This paper describes our techniques in detail and presents experimental results obtained using our implemented system.

1 Introduction

Model construction is a central problem in mobile robotics. After being introduced into an unknown environment, a robot must perform sensing operations at successive locations and integrate the acquired data into a representation of the environment that can be used by other agents—either human users or other software modules. A solution to this problem can benefit a variety of domains, e.g., architecture [9], virtual telepresence [3], and military intelligence (see http://www.darpa.mil/tto/tmr.html).

Previous research has addressed this problem from multiple perspectives. Several types of models have been proposed to represent an environment, such as topological maps, 2D occupancy grids, polygonal models, 3D models, and semantic maps. There has also been considerable research on the sensing operations themselves and on appropriate ways to integrate the data collected into a single representation.

In this paper, we focus on planning motion strategies for a robot to efficiently build a 3D geometric model (a triangular mesh) of a relatively large indoor environment. The key question we address is the following: Where should the robots go to perform sensing operations and acquire the data that will yield the geometric model? Ideally, we would like to reduce the number of sensing operations and the length of the total motion. However, in order to build reasonably accurate models, it is crucial that the robot localizes itself precisely at each location where it performs a sensing operation. As dead-reckoning alone is very imprecise (because of drift) and does not easily generalize to multiple robots, we must rely on registering sensor data obtained at different sensing positions (alignment of partial models). This requires that the portion of the environment seen by the robot at each new sensing location have enough overlap with the union of the portions of the environment seen at previous sensing locations to make the registration operation both reliable and accurate.

We propose an approach to 3D model construction that first constructs a 2D polygonal layout
(horizontal cross-section) of the environment using an online next-best-view algorithm that assumes no prior knowledge of the environment; this algorithm decides where the robot should move next based on the partial layout built so far. We then use an offline randomized art-gallery algorithm to compute a sequence of locations where the robot should perform (relatively costly) 3D sensing operations. Both our next-best-view and art-gallery algorithms are novel and incorporate techniques that address the robot localization issue. In particular, representing a 2D layout as a polygonal region allows us to accurately compute the portions of a partial model that are visible from a given robot position. In both the next-best-view and the art-gallery algorithms, we use this property to ensure that successive views of the environment can be aligned and merged.

As previously noticed in [11, 13], model construction is made even more difficult by the physical limitations of range sensors. The classical light-of-sight model (where a point on an object is visible if the line segment connecting the sensor to this point does not intersect any object in the environment) is too simplistic in practice. Sensors have range limitations, both minimal and maximal. In addition, surfaces oriented at grazing angles with respect to the sensor may not be detected properly, which limits the incidence angle between the line of sight from the sensor and the normal to the observed surface. See Figure 1. Our algorithms take these limitations into account and read the sensor parameters as inputs. For the same environment, our system produces different motion strategies for different parameters.

Finally, our algorithms plan collision-free motions between sensing positions. Here again, we exploit the polygonal layout generated by the next-best-view algorithms to compute a region in which the robot is guaranteed to move without collision. During the construction of the 2D layout, the fact that some obstacles may have not yet been seen because of the incidence angle constraint seriously complicates the computation of the collision-free region. We give a solution to this problem.

We present an overview of our model construction approach in Section 2. In Section 3, we relate our work to previous research, including research on 3D model construction using fixed range sensors. In Sections 4 and 5, we describe our art-gallery and next-best-view algorithms, respectively. In each of these two sections we also give limited experimental results. In Section 6 we describe our experimental setup made of several Nomad 200 and Super-Scout robots and we present experimental results for the entire process of constructing a 3D model. In Section 7 we discuss how our algorithms scale up for handling a team of robots.

2 Overview of Approach

Our overall approach for constructing 3D models of indoor environments is based on the observation that such environments mostly consist of vertical surfaces. As a result, a 2D horizontal cross-section of the environment captures most of its structure effectively. Our algorithm consists of the following three steps:

1. Construct a 2D layout $P$ of the environment, assuming no prior knowledge of the environment. For this step we assume that the robot is equipped with a range finder that scans the environment in a horizontal plane and returns range values at regular angular intervals.

We construct $P$ incrementally by moving the robot to successive locations. At each location, we convert the range data into polygonal chains and merge these chains with the partial layout built so far using an efficient matching algorithm. Using the new partial layout $P'$, we select a new location $p$ to move the robot to. Our next-best-view algorithm picks $p$
by estimating how much our knowledge of the environment will increase by observing the environment at \( p \) and judiciously balances this estimate with the cost of traveling to \( p \). It also ensures that there is enough overlap between \( \mathcal{P}' \) and the environment visible at \( p \), so that the range data acquired at \( p \) can be robustly aligned and merged with \( \mathcal{P}' \). The algorithm also computes a collision-free path to \( p \).

2. **Use \( \mathcal{P} \) to decide where the robot should perform 3D sensing operations.** This step is purely computational in the sense that it does not involve any robot motion. It first selects a set of positions \( S \) such that every point on the boundary of \( \mathcal{P} \) is visible from some position in \( S \). Then it computes a short tour \( \pi \) through the positions in \( S \). The intuition behind this step is the following: since our environment is composed mainly of vertical surfaces, we can build a reasonably complete 3D model of the environment by successively moving the robot to each of these positions and executing a 3D sensing operation at each location.

This step is similar to the “art-gallery” problem [12] of finding a set of guards that can together see the interior of a polygon. When the simple line-of-sight visibility model is used, the problem of finding the smallest set of guards that can see the interior of a given polygon is NP-complete [12]. In our case, it is further complicated by additional sensing constraints (e.g., maximal/minimal range). The inherent complexity of the basic art-gallery problem and these constraints have led us to develop a new randomized art-gallery algorithm.

3. **Perform 3D sensing operations and integrate the data collected into a 3D model.** In this step (not described further in this paper), we move the robot along the tour \( \pi \) and perform a 3D sensing operation at each position in \( S \). Here, our robot is equipped with a laser range finder and a color camera that is calibrated relative to the range sensor. The range finder is mounted so that it casts a vertical plane of light. As the robot’s turret rotates, the laser sweeps the environment and we obtain a 3D range image of the environment. Simultaneously, we obtain color and texture information about the environment from the color camera.

After obtaining range and color data at each position in \( S \), our system converts the range data (set of points in 3D) into an adaptively decimated triangular mesh. It uses the color information to associate a texture map with each triangle of the mesh. Finally, it aligns the new mesh with the partial 3D model computed so far and merges the new mesh with this 3D model. The texture maps are useful for performing realistic graphic walkthrough operations.

Note that, at this step, a simple alternative to generating geometric 3D models is to acquire a single color image at each location in \( S \) and to use mosaicing software to combine all the images obtained. The previous two steps remain unchanged.

The generated 3D model is likely to contain a number of “holes.” First, there are viewpoints that a wheeled robot such as ours cannot reach; these holes, which are intrinsic to the kinematic limitations of the robot, cannot be eliminated. Second, an indoor environment does not consist only of vertical surfaces and, hence, viewpoints computed from a single cross-section are likely to be incomplete. The second type of holes are usually relatively small and techniques exist to reduce or eliminate them. One technique, if the sensor allows it, is to construct a small number of 2D layouts at different heights and run the art-gallery algorithm on all these layouts simultaneously. Another possibility is to add to our system a more expensive technique that fills holes, like the “space carving” algorithm in [8], or a straightforward 3D generalization of our next-best-view techniques.

3  Previous Research

*For lack of space in this short version of the paper, we omit this section. We will include it in the final version.*

4  Art-Gallery Algorithm

We assume here that a 2D layout \( \mathcal{P} \) of the environment is given. We want to determine a sequence of positions where the robot should go and perform 3D sensing operations. Our approach is to compute a set of positions \( S \) so that sensors placed at these locations collectively see the entire boundary of \( \mathcal{P} \), and then to compute a short tour that connects the points in \( S \).

Our algorithm to compute \( S \) is incremental and uses a dual sampling scheme. First, it selects a point
Figure 2: The region $O_p$ (light grey) from which a point selected in $\partial P$ is visible, and a collection of sampled positions in this region.

Figure 3: A set of sensor locations computed by our art-gallery algorithm.

$p \in \partial P$, where $\partial P$ denotes $P$’s boundary, and computes the region $O_p$ from which a sensor can observe $p$ under the range and incidence constraints specified. An example of region $O_p$ is shown in light grey in Fig. 2, where we set the minimal sensing range to 0. Next, the algorithm picks a random sample of points in $O_p$ (see Fig. 2) and includes the sampled position that sees the largest portion of $\partial P$ in $S$. We repeat the process with the unobserved portion of $\partial P$, until all of (or most of) $\partial P$ can be seen from the selected sampled positions. Fig. 3 shows a set $S$ computed by our algorithm for a portion of the Robotics Lab at Stanford, with the sensing constraints of Fig. 2. Note that the density of positions is greater in narrower areas due to the incidence-angle constraint. Fig. 4 shows the results obtained when the maximal range of the sensor is reduced (keeping the same incidence-angle constraint). The running time for these examples is on the order of a few seconds on an SGI Indy workstation. Note that several runs of our algorithm, with different seeds for the random number generator, yield different solutions.

In the full version of the paper, we will describe the data structures we use to efficiently maintain and sample the unobserved parts of $\partial P$. We will also show that if $P$ has $n$ edges, our algorithm picks $\mu$ samples at each step, and selects a total of $g$ positions in $S$, then the total running time of our algorithm is $O(\mu g n \log(gn))$.

The path $\pi$ connecting the points in $S$ is simply an approximation to the shortest traveling salesperson path through these points [1]. We introduce additional points along $\pi$ if there is not enough overlap between the views of $P$ from successive points on $\pi$.

Our incremental algorithm has the useful property that a user can prescribe various criteria for the algorithm to stop, e.g., when the reduction in the unobserved portion of $\partial P$ is not a significant fraction of the total length of $\partial P$. This feature is very useful in practice since it is often the case that a few positions can see almost all of $\partial P$, while many more would be needed to see features like narrow corners that may compose a small part of $\partial P$. Our algorithm can also deal with environments that are not fully scannable due to a strong incidence-angle constraint.
5 Next-Best-View Algorithm

In this section, we describe our online next-best-view algorithm to compute a 2D layout \( \mathcal{P} \) of the environment given no \textit{a priori} information about the environment. Given the partial 2D layout \( \mathcal{P}' \) constructed at any step (this layout is initialized by sensing the environment at the location where the robot is first introduced), the algorithm computes a “good” location \( p \) for sensing the environment at the next step. This location is chosen so that \( \mathcal{P}' \) contains a collision-free path from the robot’s current location to \( p \). The goodness of \( p \) is based on (i) an estimate of how much new information about the environment the sensor will acquire at \( p \), (ii) the length of the path from the current position to \( p \), and (iii) whether the sensor data at \( p \) can be robustly aligned and merged with \( \mathcal{P}' \).

We construct \( \mathcal{P} \) in a sequence of stages. At the beginning of stage \( k \geq 1 \), we have a partial layout \( \mathcal{P}_{k-1} \) of the environment and a new position \( p_k \) (\( p_1 \) is the point where the robot is first introduced into the environment). In stage \( k \), we move the robot to \( p_k \) and execute a 2D range sensing operation to obtain a set of (range, angle) data pairs sorted by angle. We then process this range data as follows:

1. Construct a polygonal representation of the new data. We partition the range data into clusters separated by range discontinuities. We use a least-squares technique to fit a polygonal chain (polyline) through each cluster of points. Let \( \mathcal{L}_k \) denote the resulting set of polylines, sorted by angle around the sensor.

2. Merge the new data with the current partial layout. We align and merge \( \mathcal{L}_k \) with \( \mathcal{P}_{k-1} \) to obtain the updated layout \( \mathcal{P}_k \). To perform the alignment, we need to find a transformation between the current frame of reference of the robot (in which \( \mathcal{L}_k \) is defined) and the global frame of reference (in which \( \mathcal{P}_{k-1} \) is defined). We use an algorithm similar to the one proposed in [5]. We pick a pair of points \((p, q)\) at random in \( \mathcal{L}_k \) and find pairs \((p', q')\) in \( \mathcal{P}_{k-1} \) such that the distance between \( p \) and \( q \) is roughly equal to the distance between \( p' \) and \( q' \). We then compute a transformation \( T \) that brings \( p \) close to \( p' \) and \( q \) close to \( q' \). Let \( T(\mathcal{L}_k) \) be the set of polylines we obtain by moving every point in \( \mathcal{L}_k \) according to \( T \). We evaluate the quality of \( T \) by measuring the amount of overlap between \( T(\mathcal{L}_k) \) and \( \mathcal{P}_{k-1} \). We repeat this procedure for multiple pairs \((p, q)\) and retain the best transformation for the alignment.

3. Compute the next position where to sense the environment. This is the core step of the next-best-view algorithm. The current layout \( \mathcal{P}_k \) is composed of a set of polylines. Every internal vertex of a polyline is a vertex that will appear in the final model. An endpoint of a polyline is where the polyline will continue to “grow” because of subsequent sensing operation. A polyline endpoint can be of three types: (i) occlusion, when it is caused by one surface in the environment occluding another, (ii) incidence, when it is caused by too large an incidence angle, and (iii) maximal, when it is caused by an out-of-range distance. We compute a region \( \mathcal{G}_k \) in which the robot is guaranteed to move without collision, even with unobserved portions of the environment. The boundary of \( \mathcal{G}_k \) contains all the edges in \( \mathcal{P}_k \). To complete \( \mathcal{G}_k \)’s contour, we join polyline endpoints by fictitious edges. We generate these edges using rules based on the types of endpoints we want to connect. In the final version of the paper, we will describe these rules which guarantee that the region bounded by the polylines and the fictitious edges is collision-free.

The computation of the next position \( p_{k+1} \) then proceeds as follows. Clearly, \( p_{k+1} \) should be contained in \( \mathcal{G}_k \). One of our criteria to decide whether a point \( p \in \mathcal{G}_k \) is a good candidate for \( p_{k+1} \) is the length of previously unobserved portions of the environment cross-section that will be visible from \( p \). This leads us to consider points \( p \) in \( \mathcal{G}_k \) that see large portions of the fictitious edges. We generate such points using a technique similar to the art-gallery algorithm presented in the previous section. We pick a random sample of points in \( \mathcal{G}_k \). We disregard samples that do not satisfy the requirements imposed by our alignment algorithm. For every remaining sample \( p \), we measure its goodness by a function of the length of the fictitious edges that will be visible from \( p \) and the length of the shortest path inside \( \mathcal{G}_k \) between \( p \) and \( p \). Finally, we choose as \( p_{k+1} \) the sample that maximizes this function.

Fig. 5 shows \( \mathcal{P}_k \) and \( \mathcal{G}_k \) (region in dark grey) at several iterations (1, 2, 7, and 20) during a run of our algorithm on simulated data, and the path followed by the robot. In this example, the layout was completed in 20 iterations.
Figure 5: Evolution of the current layout and the safe region during a run of the next-best-view algorithm.

6 Experimental Results

For lack of space, we omit this section. We will present both our experimental set-up and the most recent experimental results in the final version of the paper.

7 Multiple Robots

All our algorithms were specifically designed so that they can be scaled to deal with multiple robots. We will discuss this extension in the final paper and show preliminary experimental results.

8 Conclusion

This paper described motion planning algorithms allowing mobile robots with range sensors to efficiently build 2D and 3D models of indoor environments. The two most novel algorithms are the next-best-view algorithm that decides how the robot should move to produce a 2D layout of the environment and the art-gallery algorithm that decides where the robot should go to perform 3D sensing operations. Both algorithms address the issue of localization uncertainty that is inherent in virtually all mobile robots. They also take into account the most blatant limitations of range sensors (range and incidence).

Experiments with our algorithms show that they are fast and that they produce reasonably optimized
paths in environments that are very different geometrically. Thanks to their randomized components, they are insensitive to pathological cases and relatively easy to implement and extend. They also scale up to handle a team of multiple robots.

Acknowledgments: This work is supported by DARPA/Army Contract DAEE07–98–L027, ARO MURI grant No. DAAH04–96–1–007, and NSF grant No. IIS–9618625. The second author is also supported by Rothschild Fellowship. We thank Leo Guibas David Lin, and Rafael Murrieta for valuable discussions throughout this research and for their assistance in carrying out experiments.

References


