THESIS PROPOSAL

Automated Construction of Environment Models by a Mobile Robot

Paul Blaer
Computer Science Department, Columbia University

December 9, 2004

Contents

1 Introduction

Accurate three-dimensional models of large outdoor structures, such as buildings and their surroundings, have many uses. They can provide an educational walk around a structure that is thousands of miles away. These models can allow engineers to analyze the stability of a structure and then test possible corrections without endangering the original. They allow us to preserve historical sites that are in danger of destruction, and they allow us to preserve archaeological sites at various stages of an excavation. Construction of such models is particularly useful when the original blue prints for the structures are either not available or inaccurate. In many cases; structures, ranging from third world oil refineries to medieval cathedrals, were developed over time and from many different designs, so that there is no longer one true blue print.

In all of these cases, it is useful to have an accurate computer based 3-D model of the large scale outdoor structure. Methods for acquiring such models have progressively increased in accuracy and have evolved from manual methods to more automated methods. At the simpler end of the spectrum, one can send a team of surveyors with a theodolite to the site in order to take measurements of the structure and then have a designer put together a model from those measurements. Such a model might look relatively accurate and, in the case of its larger features, be geometrically accurate. However, it does not tell the whole story. It would be extremely inefficient to hand survey all of the small features and details on the surface of the structure. These features would likely not have been surveyed, and the model would be based mostly on the designer's best guess.

More sophisticated tools do exist. There are a number of laser range scanners on the market that will sweep a beam across a large portion of the structure and return a dense point cloud of measurements. Armed with these more sophisticated instruments, one can take a number of scans around the structure and fuse them into a single point cloud that accurately represents the structure. The point cloud could then be triangulated into a mesh to give an accurate model. With a sufficient density of the point cloud that the scanner returns, one can generate models that are accurate to a centimeter or better. Now, instead of having a simplistic model of the structure, we have an accurate model appropriate for preservation and analysis.

Although the models are now far more accurate and the acquisition process is faster and more automated, there is still a major human component involved. The scanning sensor must also be physically moved from location to location, and each scanning operation itself can take up to one hour depending on the type of sensor and the density of the scan. In addition, a plan must be laid out to determine where to take each individual scan. This requires choosing efficient views that will cover the

entire surface area of the structure without occlusions from other objects and without self occlusions from the target structure itself. This is the essence of the so-called view planning problem.

The fusing or registration of the multiple scans is also a complicated process. There are a number of registration techniques currently available. The most basic is to assume that we know the exact position of the sensor at every scan. This is potentially possible using GPS systems or hand measurements. However, GPS often fails in urban environments because visibility of the satellites is obscured, and hand measurements can be inaccurate and time consuming. More robust algorithms require a certain amount of overlap between one scan and the next so that they can attempt to match features between scans. Whichever method is chosen, the operators still must go through the trouble of measuring the exact position of the scanner at each scan or, at least, ensuring there is enough overlap from one scan to the next. This adds an additional manual task to the view planning and model building processes.

As a result, acquiring these models can be expensive and time consuming for a team of human operators. Depending on the environment, it can also be physically stressful. We wish to automate the modeling process by mounting suitable scanning equipment on a mobile robot. The system will be given some basic approximate information about the environment to be imaged. It will then plan a path to a desired viewpoint, navigate the mobile robot to that viewpoint, acquire images and three-dimensional range scans of the building, and plan for the next viewpoint. The ultimate goal is to take the human completely out of this modeling process.

We are developing a mobile robot platform, AVENUE [1], that is capable of localizing and navigating itself through an urban environment. We are contributing an integrated system that will have four main components:

1. An algorithm which uses a combination of omnidirectional vision and wireless access point signals to provide topological localization of the robot on a two-dimensional map.

2. A Voronoi-based path planner which our robot uses for navigating along safe paths between successive viewpoints.

3. An improved two-dimensional view planning algorithm which computes an initial set of views appropriate for a large class of laser scanners and includes the important constraints of minimum and maximum range, maximum grazing angle, and limited field of view.

4. A new voxel-based method for choosing next-best-views which are based on the most recent set of acquired three-dimensional scans.

Our method of model building has two basic steps, the first being static and the second dynamic. The system starts with an approximate two-dimensional ground map of the target region. Based on this map, the system computes an initial set of views for the sensor. This forms the static part of the procedure. After data are collected from these initial views, a preliminary three-dimensional model of the region is constructed. Next, a voxel-based method is used to choose the next best view from the information in this model. The method now becomes dynamic as each new scan updates our model and provides new information for the next best view.

Currently there are a number of other research projects attempting to construct three-dimensional models of urban scenes and outdoor structures, such as the 3-D city model construction project at Berkeley [13], the outdoor map building project at the University of Tsukuba [26], and the MIT City Scanning Project [39]. However, for the most part, these methods leave the actual planning component to a human operator.

Our goal is to integrate all of these components and steps into an autonomous system requiring minimal human intervention. The result will be an automated robot which merges many aspects of path planning, localization, view planning, and model construction.

2 Related Work

The view planning problem can be described as the task of finding a set of sensor configurations to efficiently and accurately fulfill a reconstruction or inspection task. These configurations are often found sequentially, so this is sometimes called the next best view (NBV) problem. Two major surveys on the topic exist including an earlier survey on computer vision sensor planning by Tarabanis et al [36] and a more recent survey of view planning specifically for 3-D vision by Scott et al [31].

The literature can be divided into several separate classifications. First, there are the model based methods. These are the inspection methods in which the system is given some initial model of the scene or object to be imaged. Second, there are the non-model based methods, more generalized than the model based ones. These seek to generate models with no prior information. Finally, we have classified separately a number of the more recent methods that deal with larger scenes and are most applicable to view planning for a mobile robot.

2.1 Model Based Methods

One of the earliest of the model based methods is the work of Cowan and Kovesi [8]. Their task is to use a camera to take 2-D images of a set of features on a viewing plane. They specify and solve for a number of constraints on the views to obtain a minimum resolution, keep the features in focus, and keep the features within the field of view. They also present some of the earliest work on satisfying the visibility constraint, which they define as "the requirement that all designated surfaces to be observed must be completely visible from the selected sensor locations." They were able to solve the visibility constraint for surfaces at which the camera was directly pointed. In fact, all of their constraints were solved assuming that the optical axis of the camera was pointed directly into the targeted surface. The paper also briefly discusses the applicablity of their methods to range cameras such as laser scanners. Work by Tarabanis et al [37] presents a more generalized analysis of the constraints to the view planning problem, though its focus is still on 2-D cameras. In this work, the constraint on the optical axis of the camera is eliminated and equations are derived to solve for the constraints from any camera orientation. In addition, a more robust method for solving the visibility constraints is developed [35] in which a viewing volume for the sensor is computed. Any location within that volume can view the targeted surface.

The model based methods are not, however, limited to 2-D camera vision systems. There are also examples of inspection tasks for 3-D vision systems such as the work of Tarbox and Gottschlich [38], which uses a range camera that moves on a sphere surrounding the object to be imaged. The plan for scanning the object is generated based on a model of what the object is supposed to look like. The viewing sphere on which the sensor moves is discretized into viewing locations. The model of the object is also discretized. A measurability matrix, $C(i, k)$, is then calculated. The i is a location on the object and the k is a range camera location. An entry in this matrix is 1 if location i is visible from range camera position k and is 0 otherwise. The next best views are chosen such that they see the most "difficult to view" points on the object. A point's difficulty to view is computed by taking the inverse of the number of camera locations that can see it. In addition, a grazing angle constraint is considered. Later work by Scott et al [32] expands on this method by adding a constraint on the sensor measurement error, favoring views that are less likely to have erroneous measurements. In addition, they add an incremental process to the system which starts with a very rough model of the object to be inspected. It then plans and executes a set of scans to create a more refined model, and repeats that process to plan views based on an increasingly accurate model of the object. Conceivably this

method could be extended to a situation for which no model exists and could then be used to incrementally build up a model.

The class of Art Gallery problems can also be considered model based view planning techniques. In an art gallery problem, one is trying to solve for the optimal set of views in a region such that all the faces of a 2-D polygon in that region are seen. This can be extended to the 3-D case by making the assumption that if you can see a 2-D wall, then that will translate into being able to view the 3-D wall. The problem is NP hard, so the proposed solutions are approximations. Early works such as Xie el al [11] solve the problem with a deterministic approximation. More recently, González-Baños et al [17, 15] proposed a randomized method for choosing such views. Additionally, there has been some work done by Danner and Kavraki [10] to extend the planning to three dimensions. Instead of planning based solely on the 2-D model of the scene, complete 3-D models are used from the beginning.

2.2 Non-Model Based Methods

In contrast to the model based methods are the non-model based ones which assume no prior information about the objects being imaged. These methods can be further categorized by their representation of the viewing volume. One set of next best view methods uses a volumetric representation of the world such as voxel grids. The other set uses only a representation of the known surface of the object being modeled, which is usually stored as a triangulated mesh.

2.2.1 Volumetric Methods

The earliest of the non-model based methods was done by Connolly [7]. His method has an object on a center stage with a range camera moving on a sphere of fixed distance around the object. The optical axis of the range camera is always pointed toward the center of the object. The object is represented by a set of voxels, stored in an octree. To start, the view volume is labeled as entirely unknown. The view sphere is discretized into points and the next best view is chosen using a brute force method. For each point on the view sphere, the number of unknown voxels that can be seen without occlusion by known voxels is computed and the view that has the maximum visibility is chosen. As late as 1995, researchers were still improving on this basic concept. Banta et al [2] uses a similar method, but instead of using a purely brute force approach, they note that viewing points of maximum curvature (in their case, corners) on the object resulted in the most surface being visible. By using this heuristic, the number of views needed to cover the object is decreased. Massios and

Fisher [22] also use this kind of voxel-based representation to aid in generating models of small objects at the center of a spherical view volume. Next best views are chosen using a weighted sum of the "quality" of the voxels being viewed and the number of new voxels being viewed. The quality is based on the scanner's grazing angle with the estimated surface at that voxel. For the visibility constraint, occlusion planes are formed between jump edges. The more of these occlusion planes that a view can see, the higher that view's weight. Soucy et al [34] also track the boundary voxels between the known and unknown portions of the viewing volume and use the largest boundary as the target for the next best view.

2.2.2 Surface-Based Methods

Maver and Bajcsy [23] did early work using surface based methods. Their method was also one of the earliest approaches to use an exclusively occlusion-based analysis of the scene to choose next best views. Their system was specifically designed for their type of ranging system. They use a calibrated system consisting of a laser stripe that is projected onto the viewing plane and a camera that is offset by a fairly large baseline and images that laser stripe. The view planner is a two stage system. The first detects and resolves occlusions caused by the laser stripe not being visible to the camera. The second stage detects and corrects for occlusions caused by the laser stripe not being able to reach a portion of the scene.

Although Pito [29] stores primarily a mesh of the surface of the object being imaged, he also recognizes the importance of keeping track of regions where no data have yet been acquired. Instead of keeping an exact record of these "void volumes," he stores small patches, called "void patches," along the edges of the known meshes. His assumption is that the regions around the boundary of the known mesh and the unknown void volume will yield the most new information about the surface. To choose the next view, a grid of cells called the "positional space" is arranged around the viewing volume. Only sensor locations in this positional space are considered. At each location, the number of void patches viewable is stored. In addition, Pito acknowledges that multiple views will not match exactly and that there will be a need to register one view with the next. To facilitate this registration, it is helpful for neighboring views to have some overlapping region. In addition to storing the number of void patches at each positional space cell, he also stores the number of known surface patches viewable. Next best views are chosen from the positional space such that they maximize the void patches seen while still keeping some minimum amount of known surface visible. Reed and Allen [30] also look at the boundary between the existing mesh and the void region. However, in their approach, the edges of the

current mesh are extruded back to the bounding box of the viewing region along the scanning direction. This results in large "occlusion surfaces" that must bound the actual surface of the object being imaged. The next best view is chosen by targeting the largest patches on this occlusion surface.

The work by Zha et al [42] also represents the target object as a mesh. They use techniques similar to those used in some of the volumetric methods discussed before in order to choose next best views. As in the works of Massios and Fisher and of Soucy et al, they look at the boundaries between the known and unknown regions. After a scan is taken, they look at the edges of the resulting mesh and extend a ray into the unknown area along the line formed by connecting the edge at a particular point and the camera. The directions of these occlusion rays are averaged, and a view that faces in the opposite direction of the average occlusion ray should view the largest portion of the unknown region. Viewpoints are rated based on a weighted sum of a number of factors. The first is how close the direction of a candidate viewpoint is to the "best" direction described above. Zha et al also acknowledge that a certain amount of overlap is necessary to register consecutive scans, so the second factor is how much of the existing mesh can be seen from a given viewpoint.

Whaite and Ferrie [41] take a substantially different approach to the view planning problem. Their system is designed to be extendable to an autonomous explorer moving through an unknown environment. However, in practice, their methods are tested only on single, smaller objects at the center of a sensor's viewing volume (similar to the vast majority of other methods described above). The method approximates its target with a superellipsoid. The regions of the model where the actual data is least likely to fit properly are computed, and the next best view is chosen such that it can see these highly uncertain areas.

2.3 View Planning for Mobile Robots

In recent years, the 3-D view planning literature has begun to focus more on planning for mobile robots in the kinds of outdoor and less structured environments in which we wish to work.

The work of Sequeira et al ([21, 33]), although not specifically geared toward a mobile robot, is designed to work with large indoor and eventually outdoor scenes. It is applicable to mobile robots acting autonomously or for guiding a human operator in placing a sensor. The method maintains two meshes, the known surface mesh and the void mesh. The void mesh is the boundary between the known world and the unexplored void volume. It is this void mesh that their algorithm targets. In addition, they have a target sampling density for each small patches on the mesh as well as an

overlap constraint. A weighted average of these various constraints is maximized to choose the next best view.

The work by González-Baños et al [16] looks only at 2-D map construction but applies a next best view approach to the process. In their method, a line laser scanner is mounted on the robot and oriented horizontally. When a scan is taken, the 2-D boundaries of obstacles are extracted and the free space that has been revealed is added to the model. Next best views are chosen such that the most unknown space on the 2-D plane is acquired, taking into account the occlusions of the obstacle boundaries. Grabowski et al [19] take a similar next best view approach to 2-D map construction, but instead use sonar rather than laser range finders.

The work of Nüchter et al [25] uses a method similar to that in Baños' work but extends it to 3-D model construction. In this case, there are two line scanners. The first is mounted horizontally near the ground plane and the second is mounted vertically. Next best views are chosen by maximizing the amount of 2-D information that can be obtained at a chosen location; however, 3-D data is actually acquired. They also propose a method for planning in three dimensions by taking planar slices of the 3-D data at multiple elevations and then running their 2-D algorithm on each slice.

3 Overview of Our Method

The overall goal of our robotic system is to construct an accurate three-dimensional model of a large outdoor scene. The construction of this environment model is automated, with the robot only given a simple two-dimensional map of the scene from which to start.

The three-dimensional model construction process will proceed in two stages. The first stage of the modeling process will utilize only the two-dimensional map to construct an initial model of the scene and an initial set of views. The second stage of the process will use this initial model and view data to plan more refined views that resolve occlusions that occurred in the first stage.

By using this two stage method, we will be able to minimize the number of high resolution scans needed to construct the model. Our initial run through the environment will use views planned only from the two-dimensional map. As a result, building overhangs, unmapped foliage, and other obstructions might occlude our views of important portions of the scene. After the initial run, we will have some three-dimensional data about the environment's structures and how they occlude the scene. We will now be able to make use of the data to plan unoccluded views for

future modeling. With each data set acquired, we will then have more information to help plan more efficient subsequent views.

The platform for this project is the AVENUE mobile robot ([20], [1]). This robot has as its base unit the ATRV-2 model originally manufactured by RWI (see Fig. 1). To this base unit, we have added additional sensors including a differential GPS unit, a laser range scanner, two cameras (including one omnidirectional camera [24]), a digital compass, and wireless Ethernet. The system is currently capable of localizing itself with its sensor suite, planning a path to a specified location, and navigating itself to that location.

Figure 1: The ATRV-2 Based AVENUE Mobile Robot.

At the beginning of the process, we do not have any 3-D data. We assume, however, that we have an approximate 2-D map of the environment. For many existing sites, there is an existing Geographic Information System (GIS) that has topographic, building foot print, and street layout data. This is especially true for urban areas. A sufficient 2-D map can often be extracted from such information. If

this kind of data is not available, it is possible to use our robot to build the 2-D map. One of our initial test beds will be the northern half of the Columbia University campus. A 2-D map of this region has already been derived from existing CAD models of the campus (see Fig. 2). After the Columbia University test bed, we will then move on to larger sites such as Governor's Island in New York City (see Fig. 3).

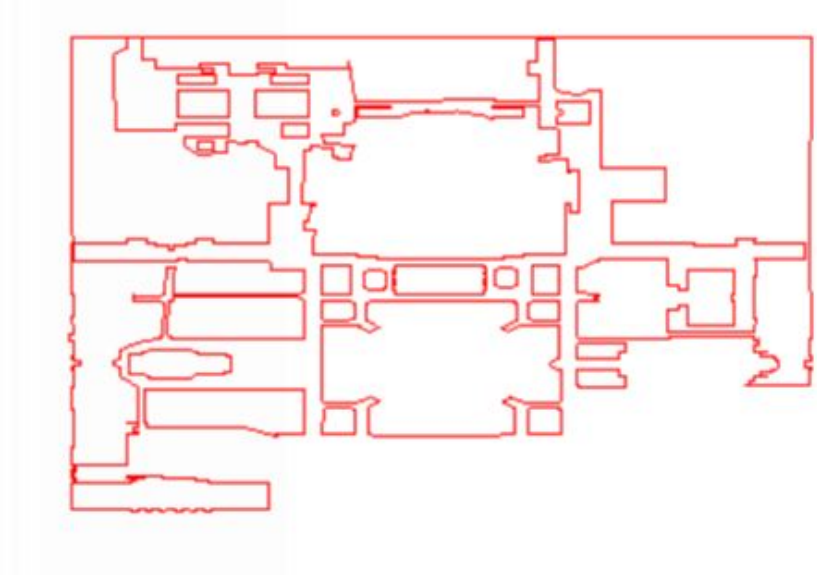


Figure 2: The 2-D map of our test area on the northern half of the Columbia University campus.

For the first phase of our modeling process, we can treat our problem as a variant of the 2-D art gallery problem and then apply our path planning methods to generate a tour that acquires a preliminary 3-D model of the scene. We initially assume that if the robot can view the 2-D foot print of a wall, it can then view the entire wall. Since we are basing our initial views on limited information about the scene, the scans that we take will not acquire all of the needed information for a final model.

In the second stage of the algorithm, we take this initial 3-D data from the scene and use it to compute additional views that resolve occlusions and fill holes that were detected in the initial model. Since this second set of scans will be planned more carefully by taking into account more information about the scene, we can now predict what scan location will give us the most new data. As a result, fewer scans will be necessary to cover the entire scene.

Figure 3: A Map of Governor's Island, another test area for our system.

4 Initial Modeling Stage

In this stage of the modeling process, we choose a series of environment views for our robotic scanning system based entirely on the two-dimensional information we have about the region. Since these planned views are based on limited information, we will most likely have to fill in gaps in the data at a later stage of the algorithm. This initial data will serve as a seed for the bootstrapping method used to complete the modeling process.

4.1 Planning the Initial Views

The classical art gallery problem asks where to optimally place guards such that all walls of the art gallery can be seen by the set of guards. Solutions to this type of problem can be applied to our initial planning task. We wish to find a set of positions for our scanner such that the scanner can image all of the known walls in our 2-D map of the environment. The entire view planning problem can be solved by making the simplifying assumption that if we can see the 2-D foot print of a wall then we can see the entire 3-D wall. In practice this is never the case, because a 3-D part of a building facade or other wall that is not visible in the 2-D map might obstruct a different part of the scene. However, for an initial model of the scene to be used later

for view refinement, this assumption should give us enough coverage to be sufficient.

The traditional art gallery problem assumes that the guards can see all the way around their location, that is, they have a 360 degree field of view. In many scanning systems, this is not true. So to achieve a complete 360 degree view, one might have to rotate the scanner in place and take several scans. The Leica scanner that we use actually does take 360 degree scans, so this is not a problem that has to be considered here. There is, however, an issue of the range of the sensor. The traditional art gallery problem assumes unlimited distance. This is not true in the case of our sensor, which can only accurately measure distances of up to 100 meters. In addition, we have to consider the grazing angles at which the laser beam hits a wall. Large grazing angles can return very poor samples. Beyond a maximum grazing angle, no laser return beam can be detected. Furthermore, the sampling resolution of a structure's surface becomes worse at larger grazing angles. Traditional art gallery solutions do not address the grazing angle constraints.

The art gallery problem has been proven to be an NP hard problem; however, early works such as Xie el al [11] have used deterministic approximations to find good solutions. More recently, González-Baños et al [15] proposed a randomized method for choosing such views.

We initially considered the first-phase two-dimensional view planner for our system with traditional, non-randomized approaches [11, 28]. This, however, does not extend well to our environments since the traditional method places observers directly on the vertices of the free space, something that cannot be done because of the size of our scanning equipment. In addition, the method does not take into account any constraints on the field of view of the sensor. As a result, we used the González-Baños' randomized algorithm [15] by extending it to include the visibility constraints of our sensor such as minimum and maximum range as well as grazing angle.

In our version of the randomized algorithm, a set of initial scanning locations are randomly distributed throughout the free space of the region to be imaged. The visibility polygon of each of these points is computed based on the constraints of our sensor. Finally, an approximation of the optimal number of viewpoints needed to cover the boundaries of the free space is computed from this set of initial locations.

For our initial test of this algorithm, we used a simulated environment. The region (see Fig. 4) represents a long hallway with eight hexagonal pillars evenly spaced and located not far from the walls. In this test region, we chose to use an initial set of 200 random scanning locations (see Fig. 4).

Next, the visibility of each of the viewpoints is computed. We use the ray-sweep algorithm [18] to compute the visibility polygon (also see Fig. 4). This polygon has two types of edges. The first contains the obstacle edges that are on the boundary

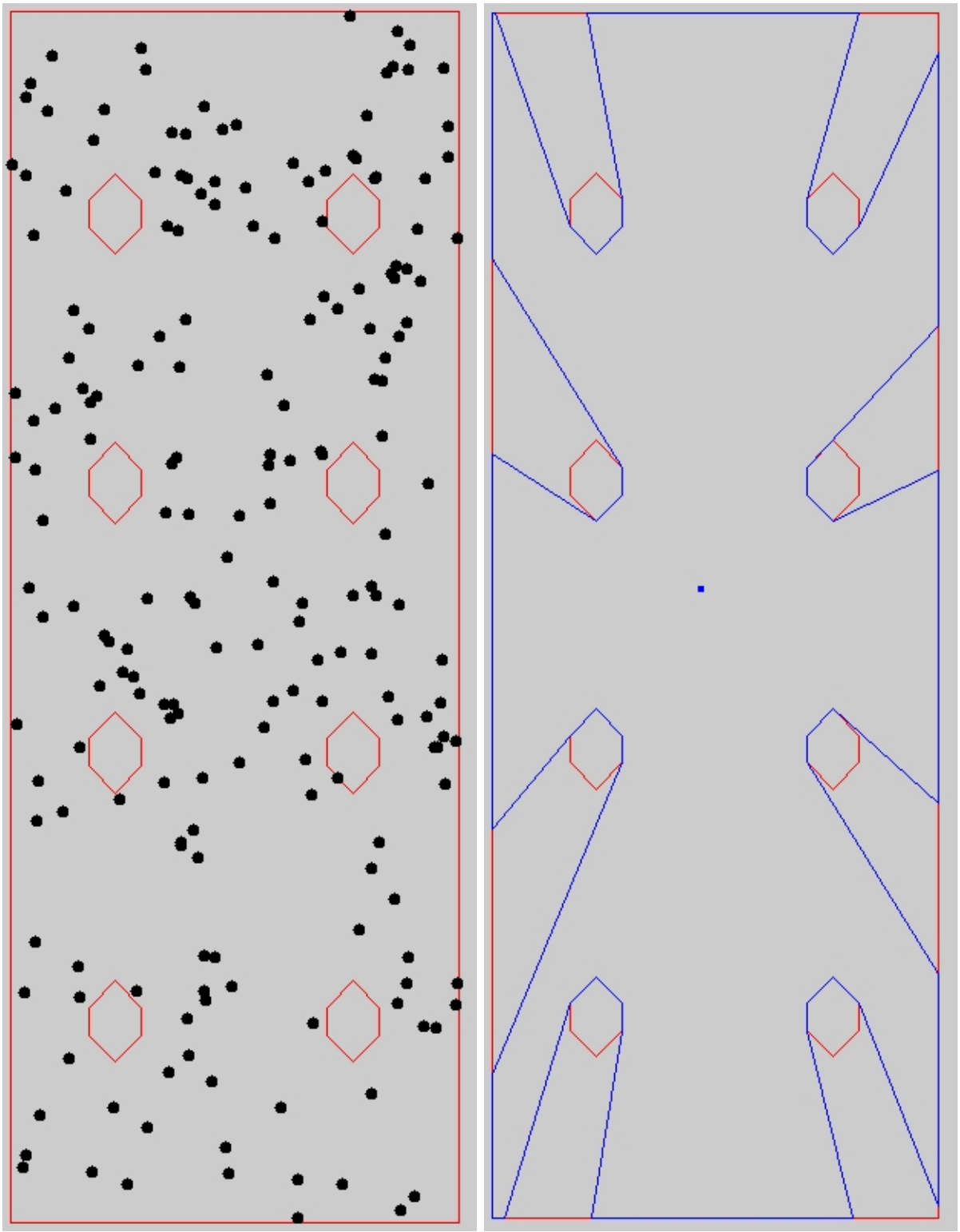
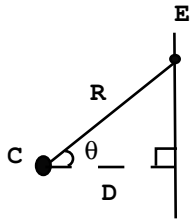


Figure 4: The test region for the first phase of our view planning algorithm together with an initial set of 200 viewpoints randomly distributed throughout the free space of the region (left). One selected viewpoint together with its visibility polygon (right).

of the region's free space. The second contains intermediate edges which lie in the interior of the free space. We then discard the intermediate edges so that the only remaining edges of this polygon are on the boundary of the free space. These edges are then clipped based on the constraints of our scanner. This gives a set of obstacle edges on the boundary that a viewpoint at a given location can actually image (see Fig. 5).

For the range constraints, we set a maximum and minimum range for the scanner. To apply the constraint, we first use the maximum range of the scanner to create a circle around our device location. We then clip all of the currently visible obstacle edges with this circle. There are three cases to consider. (1) If both end points of the edge are inside the circle, then the edge is completely visible and we keep it entirely. (2) If only one end point is inside the circle, then we replace this line segment with a clipped segment that goes from the point already inside the circle to the point at which the line intersects the circle. (3) If both end points are outside of the circle, we must determine whether or not the line intersects the circle at all. If the line does intersect the circle, we replace the original end points with the two intersection points; otherwise, we simply discard the line. For the minimum scanner range, we use a similar procedure. A circle whose radius is the minimum range is used and the parts of line segments that fall inside it are dropped.

We also constrain the grazing angle. Our sensor loses accuracy at grazing angles larger than 70^o. Our camera is located at point C and the edge that we are attempting to clip is E. The known distance between the camera and the edge is D, and the grazing angle for an arbitrary point on the edge is θ at distance R from point C. We simply find the subsegment of E for which θ is no greater than some fixed value (in our case 70^o) for all points on the subsegment.



We could additionally constrain our sensor to have a limited field of view, however this is not necessary with our current scanner and its 360^o field of view.

Finally, we utilize a greedy cover algorithm to select an approximation of the optimal number of viewpoints to cover the entire scene. We first select the viewpoint which sees the largest amount of the boundary and we then remove that section of the

boundary from the coverage of the remaining potential viewpoints. We repeat this process until either the entire boundary has been covered or until adding additional viewpoints gains no new information. Our algorithm usually returns between eight and ten scanning locations for our test region (see Fig. 5) with 100% coverage of the region's obstacle boundary.

We then ran the same process on our two-dimensional map of the Columbia campus. In this case, we used an initial set of 1000 randomly chosen viewpoints. The algorithm usually chose between 30 and 45 viewpoints (see Fig. 6) which covered, on average, 95% of the obstacle boundaries on the campus map.

It is important to note that not all of the boundary can be covered by this randomized algorithm. There are quite a few small, recessed areas on the campus map. It is not very likely that all of these areas will be covered because the free space from which they can be seen is a very small percentage of the entire free space. Since the observation locations are chosen randomly, the likelihood of placing a scanning location within that small region is very low.

This can be dealt with in a number of ways. First, we could increase the number of random samples that our algorithm chooses. If we were to increase the initial sample set too much, the run time of our algorithm would increase substantially. Second, since we know the coverage of each scanning location chosen, we know exactly what regions have not been covered and we could therefore select viewpoints specifically designed to see those areas. Finally, the second stage of our algorithm, which specifically looks for occlusions in the model, will likely fill in any gaps caused by these missed boundary regions. In our work, we will rely on the algorithm's second stage to take care of the small recessed areas.

There is major constraint that we have not yet addressed, namely the overlap constraint. Localization of the robot will never be error free and therefore relying on the robot's position estimate as the sole method for registering one scan to the next in this initial modeling phase may result in poorly registered scans. To improve the accuracy of the model, we need to utilize other registration methods such as the iterative closest point (ICP) method [3, 6, 40]. ICP and algorithms like it require a certain amount of overlap between one scan and the next. As a result, we would like to choose viewpoints that have significant overlap in the areas that they cover. We will implement this by modifying our greedy set cover algorithm so that each potential viewpoint has a weighted score that includes both the amount of unseen boundary that it can scan as well as the amount of already-seen boundary that it can view. This will allow us to apply standard registration algorithms.

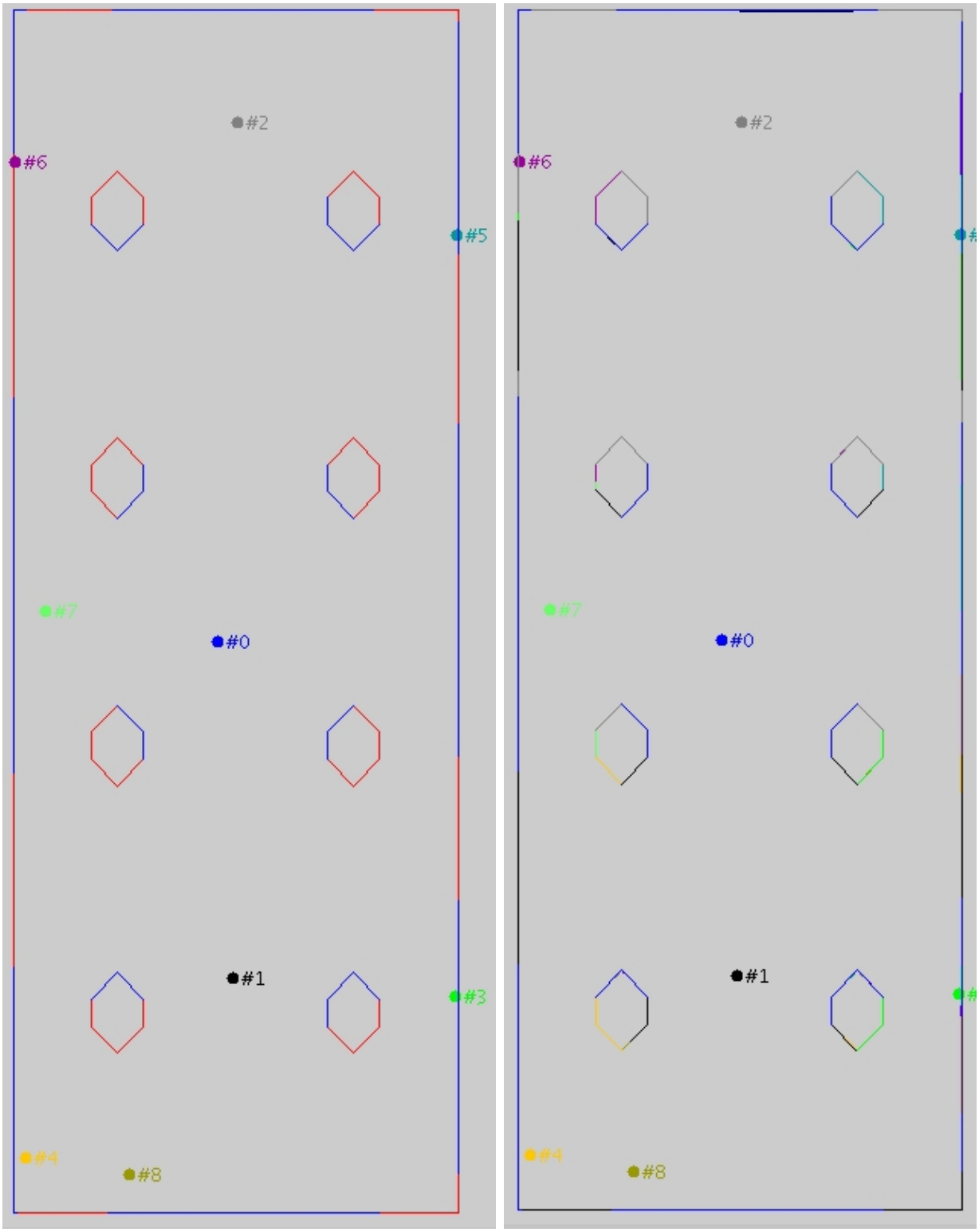


Figure 5: The final set of 9 scanning locations chosen for our simulated test region. On the left, the clipped obstacle edges for scanning location #0 are indicated. On the right, each of the 9 scanning locations is assigned a color and the line segments of the same color are the clipped obstacle edges that the viewpoint images. (Overlapping edges from different viewpoints are indicated in only one color.)

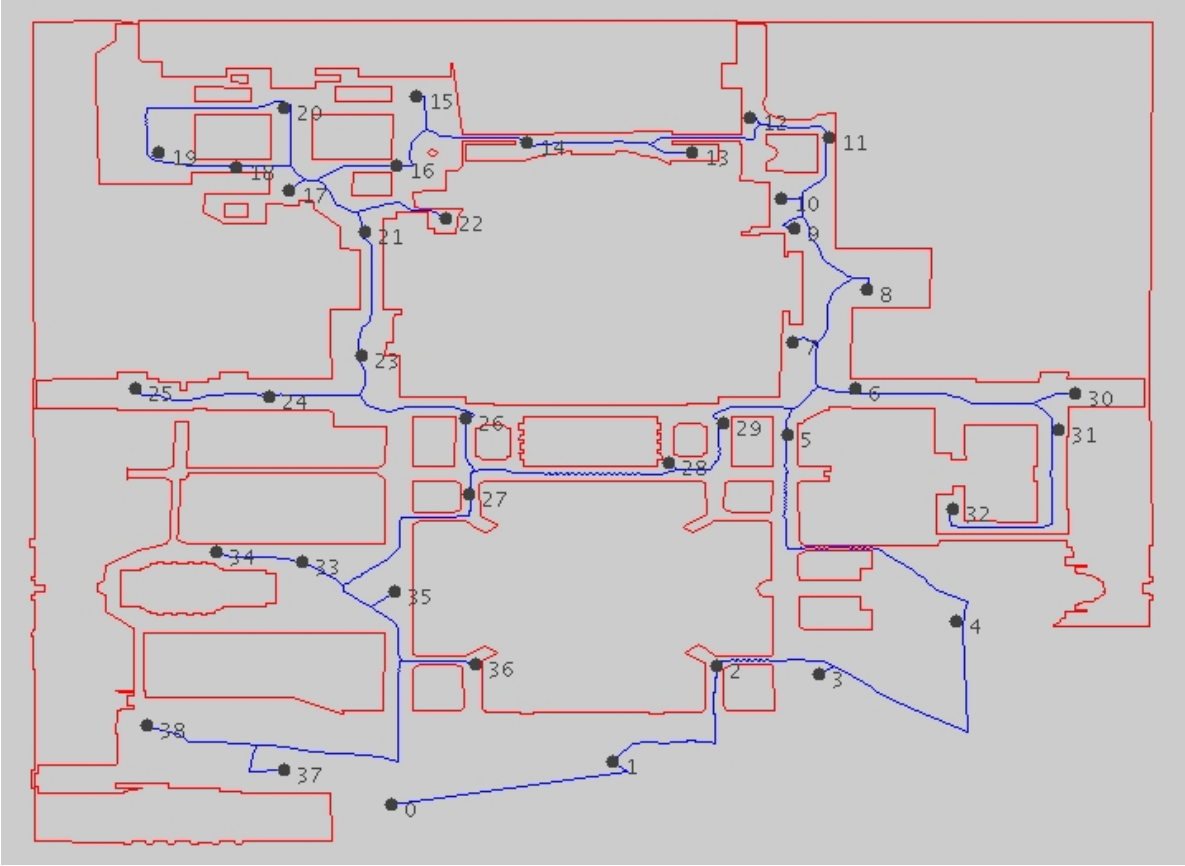


Figure 6: Our 2-D campus map and the set of scanning locations chosen by our algorithm. The numbers indicate the order in which the scanning locations would be visited using a greedy approximation of of the traveling salesman problem to generate the tour. The blue paths indicate the safe paths planned for the robot to travel along.

4.2 Moving to the Chosen Scanning Locations

We have developed the AVENUE system for use with our modeling robot. This system can be broken down into two primary parts: the path planning module (see section 4.2.1) and the localization and navigation module (see section 4.2.4). Given a two-dimensional representation of its environment, the robot has a path planning component that computes the shortest unobstructed path from the current location to a target location, in this case the desired site for the next scan. It then uses its sensor suite to localize and navigate itself along that path.

To integrate our view planning algorithm and the AVENUE system, we need to provide this system with our current location and the location of the next viewpoint so that the robot can travel between the two. In the first stage of our view planning algorithm, we take a global planning approach in which all of the viewpoints are chosen at once. In the second stage, we use online planning in which the viewpoints are chosen sequentially.

Integrating the second phase of the view planning is easier, since we know our current location and we know where we want to go next. The first phase is a bit harder since we have to generate an ordered sequence of viewpoints through which to travel. This harder problem can be formulated as a standard traveling salesman problem (TSP). Although this problem is NP hard, good approximation algorithms exist to compute a near optimal tour of the observation locations. The cost of traveling between two observation points can be computed by finding the shortest unobstructed path between them and using the length of that path as the cost. The method that we describe in section 4.2.1 generates shortest paths between two locations, and we can use that method to compute edge costs for our TSP implementation. Figure 6 shows a potential tour of viewpoints using this method.

4.2.1 Path Planning

A method using generalized Voronoi diagrams to generate a mobile robot's path greatly reduces the possibility that the robot will actually come in contact with an obstacle. The Voronoi diagram for a collection of given points (called sites) is the graph formed by the boundaries of specially-constructed cells. Each of these cells surrounds one of the given sites and has the property that all points within the cell are closer to the enclosed site than to any other site. Voronoi diagrams can be generalized to situations in which the given sites are two-dimensional obstacles rather than mere points. In this type of problem, the boundaries of the specially-constructed cells are equidistant between the two nearest obstacles. These cell boundaries form

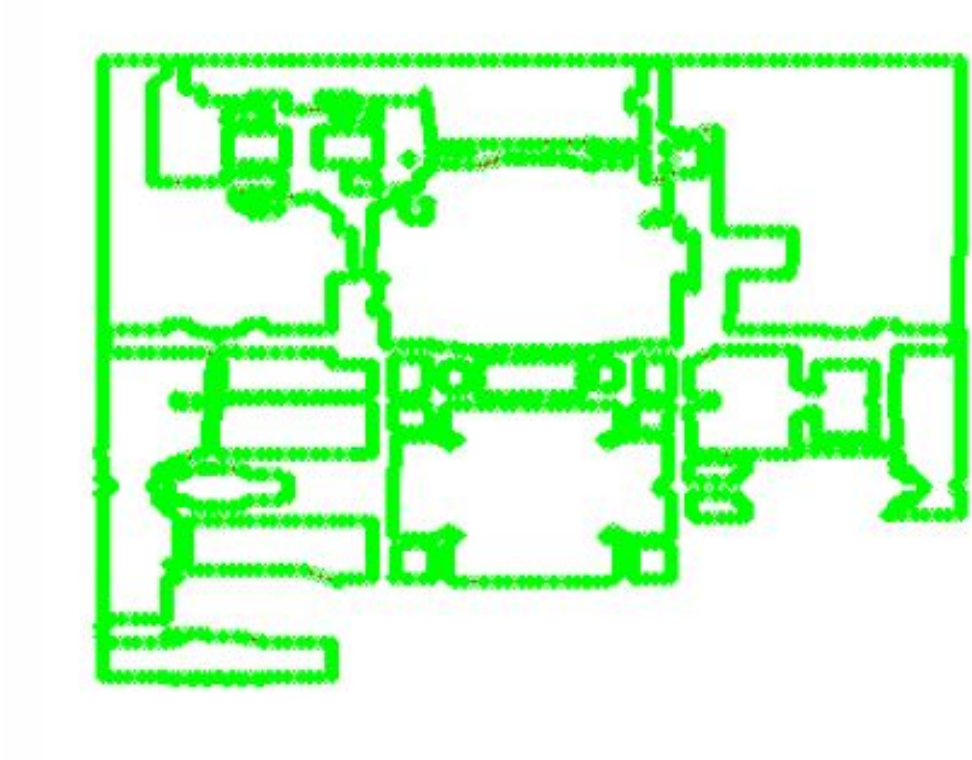


Figure 7: The points that approximate the polygonal obstacles for the generalized Voronoi diagram.

the generalized Voronoi diagram and are ideal for a mobile robot's path. By having the robot travel midway between the closest obstacles, one minimizes the chance that an error in the localization of the robot or an inaccuracy in the base map will result in a robot-obstacle collision.

4.2.2 Computing the Generalized Voronoi Diagram

Computing the Voronoi diagram for a collection of n given points (sites) in two dimensions is a well-known problem. Steven Fortune [12] presented a novel and extremely efficient method for the computation. By using a sweep-line algorithm, he was able to reduce the time complexity to order $O(n \log(n))$.

The obstacles encountered in robot path planning are actually extended two-dimensional objects, not simple points. One therefore needs an appropriate generalization of the Voronoi diagram in order to deal with such obstacles. Although the

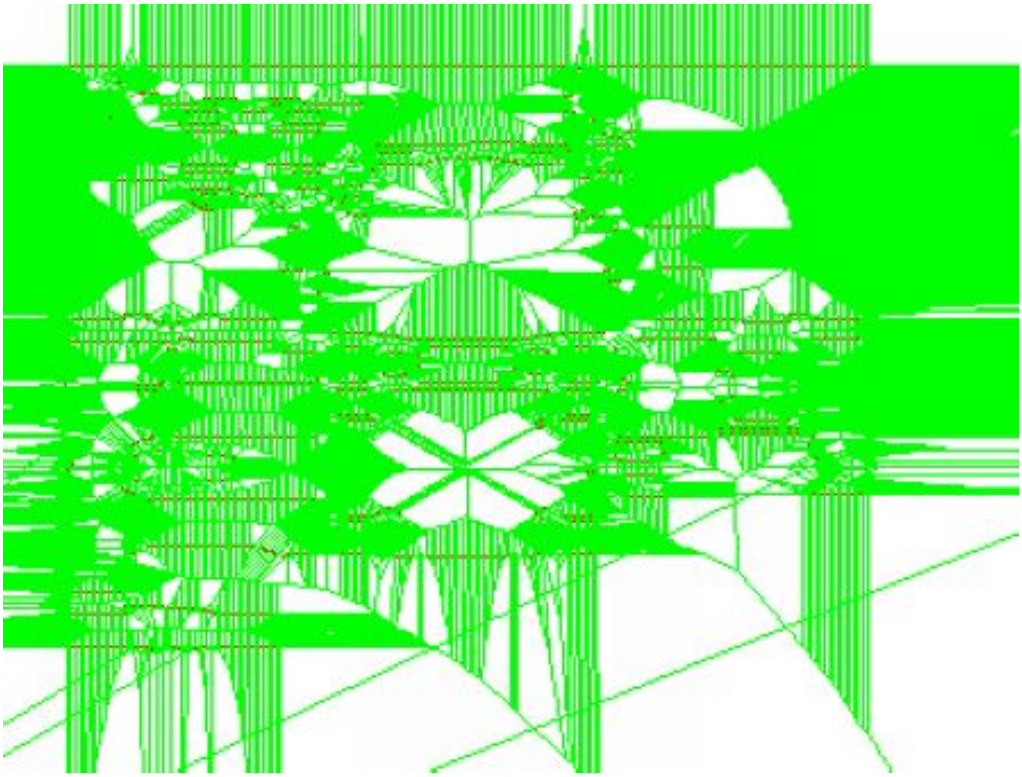


Figure 8: The Voronoi diagram for all of the points which approximate the polygonal obstacles.

usual Voronoi diagram for a discrete set of given points only contains edges that are straight line segments, the edges of a generalized Voronoi diagram will contain parabolic arcs as well as straight lines. Although possible, it is algorithmically difficult to generate such a Voronoi combination of parabolas and straight lines (linked together at point vertices). Even if one did produce this generalized Voronoi diagram exactly, one would still have to approximate each parabolic arc by a collection of small line segments so as to produce usable commands for the robot's motion. Okabe, Boots, and Sugihara [27] have suggested a very useful approximation method for finding the generalized Voronoi diagram for a collection of two-dimensional obstacles. The boundaries of the polygonal obstacles can be approximated by the large number of endpoints that result from subdividing each side of the original polygons into very small segments (see Fig. 7). In this way, one obtains a set of discrete points for which one can compute the ordinary Voronoi diagram (see Fig. 8). Okabe, Boots,

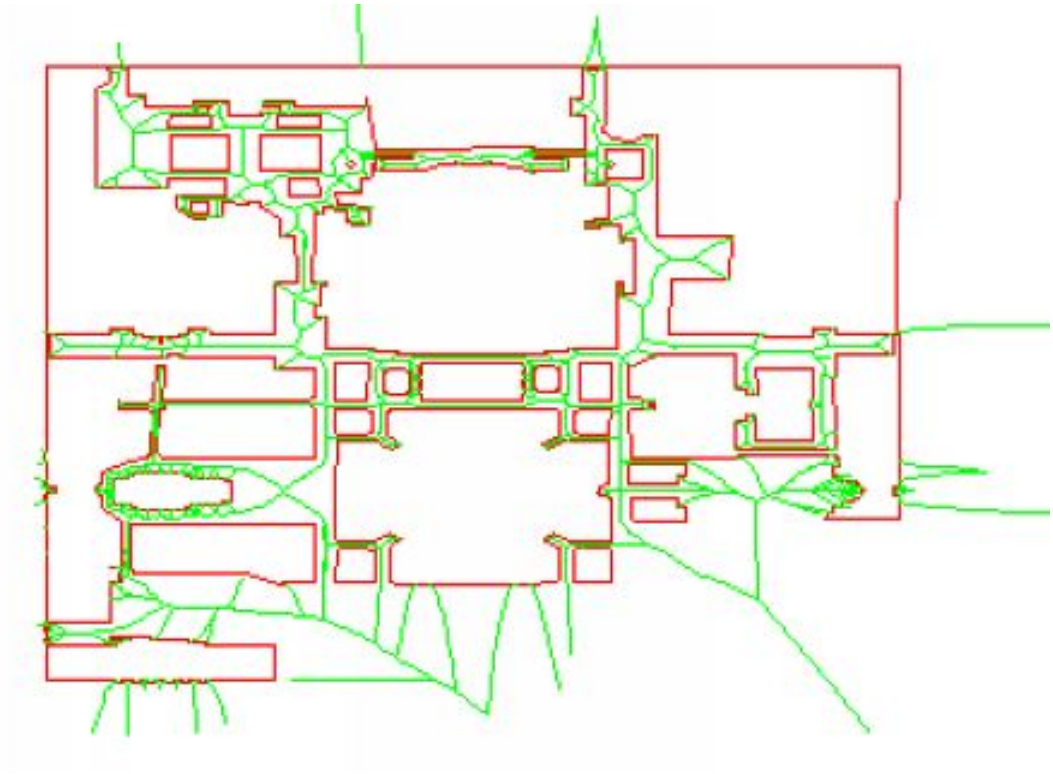


Figure 9: The final, generalized Voronoi diagram after our elimination procedure.

and Sugihara suggest that one then eliminate each and every Voronoi edge that was generated by two points located on the same obstacle. The remaining edges presumably give a good approximation for the generalized Voronoi diagram determined by the original two-dimensional obstacles. Although we use the suggested discrete-point approximation for the polygonal obstacles, we have modified the elimination procedure. Instead of removing edges that were generated by points on the same object, we remove all edges that intersect an obstacle boundary or are located entirely inside an obstacle. To accomplish this, we simply check the endpoints of each Voronoi edge to see if either endpoint lies inside any of the obstacles. If either endpoint does, we then eliminate the edge. The benefit of our procedure is clear when concave obstacles are present. Pieces of the generalized Voronoi diagram we construct can actually go into the recesses of a building, which is useful if one wants the robot to travel into such areas. The elimination procedure of [27] would actually remove these paths, which are generated by points on the same obstacle but on different sides of a recess. The

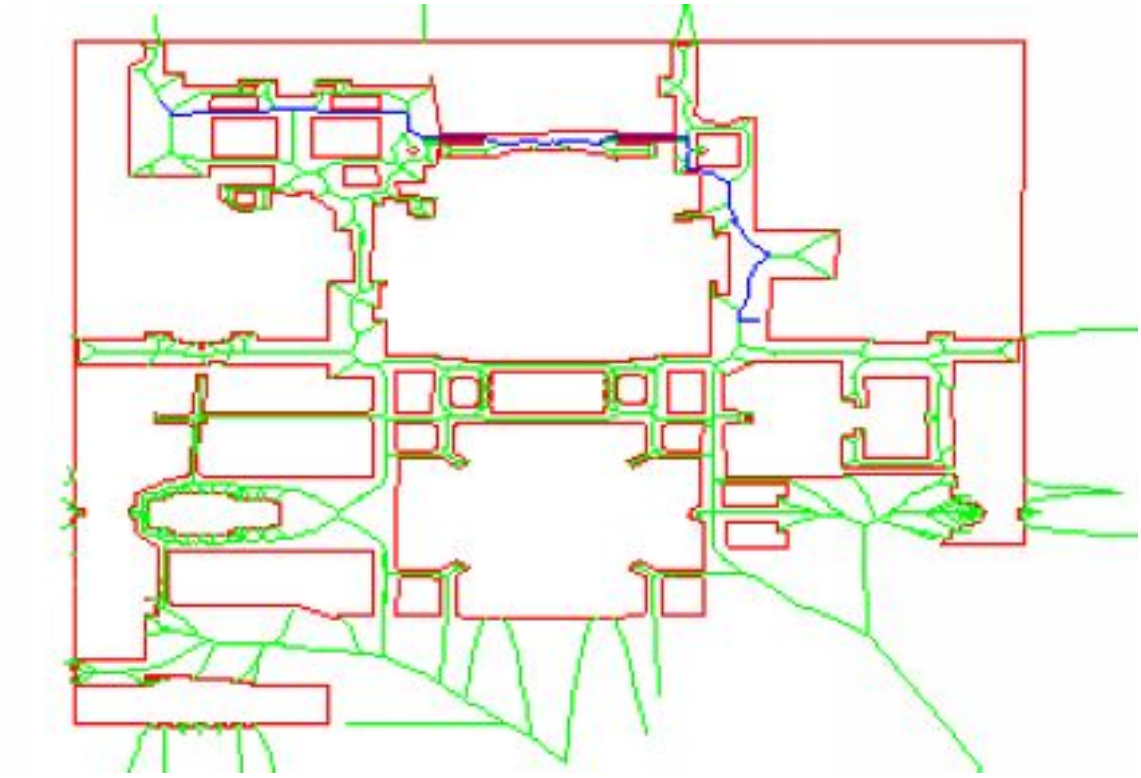


Figure 10: A simple path chosen along the Voronoi diagram by the search algorithm.

graph resulting from our elimination procedure (see Fig. 9) is a good approximation of the generalized Voronoi diagram.

4.2.3 Computing the Path

Once the generalized Voronoi diagram for a given set of obstacles has been computed, one must connect the robot's actual starting and stopping points to the diagram. In our method, one computes the generalized Voronoi diagram only once for a given map and one is then able to choose many different starting and stopping points on that map.

Once the robot's starting point is specified, we sort all the Voronoi vertices by ascending distance from the starting point. This can be done using a quick sort, so the cost is only $O(n \log(n))$. Then, beginning with the closest Voronoi vertex, we test to see if the straight line from the starting point to this vertex intersects any of the polygonal obstacles. This test uses a standard polygon-clipping algorithm. If there

is no intersection, then we have an appropriate starting Voronoi vertex connected to the robot's starting position. If there is an intersection, then we repeat the test with the next closest vertex. The process continues until we find a Voronoi vertex that can be connected to the robot's starting point by a straight line that does not touch any of the obstacles. In practice, the closest vertex usually works. The entire procedure is repeated for the robot's stopping point.

Now that starting and stopping vertices on the generalized Voronoi diagram have been chosen, one has to search for a practical path between these two vertices. Dijkstra's algorithm is sufficient to produce optimal paths along the computed generalized Voronoi diagram (see Fig. 10).

4.2.4 Localization and Navigation

The navigation portion of the AVENUE system [14] currently localizes the robot through a combination of three different sensor inputs. It makes use of the robot's built-in odometry, a differential GPS system, and a vision system. Odometry and GPS are the primary localization tools. The vision input is a useful supplement for localization when some preliminary data are available about the region and its structures. The vision system matches edges on nearby buildings with a stored model of those buildings in order to compute the robot's exact location. Initially the model is rough and approximate. It becomes more refined and precise as the actual model construction process progresses. However, to pick the correct building model for comparison, the robot needs to know its approximate location. Odometry can be problematic because of slippage. In urban environments with tall buildings, GPS performance can fail when not enough satellites can be seen. To alleviate these problems, we use our two-level, coarse-to-fine vision scheme that can supplement GPS and odometry for robot localization. First we topologically locate the robot [4] with a coarse position estimate and then we use this estimate as the initial approximation for the precise localization which matches building edges and models.

Our coarse localization method involves building up a database of reference omnidirectional images (such as Fig. 11) taken throughout the various known regions that the robot will be exploring at a later time. Each reference image is then reduced to three histograms, using the Red, Green, and Blue color bands. Omnidirectional images are used because they are rotation invariant in histogram space. When the robot is exploring those same regions at a later time; it will take an image, convert that to a set of three histograms, and attempt to match the histograms against the existing database. The database itself is divided into a set of characteristic regions. The goal is to determine in which specific physical region the robot is currently lo-

cated. This method was improved by looking at the histograms of each image at multiple resolutions rather than just at the image's original resolution.

Figure 11: Our robot's omnicamera (left) and a typical image from that camera (right).

This method still suffers somewhat from sensitivity to outdoor lighting changes. It also can have some difficulty distinguishing between very similar looking topological regions. Therefore, we have developed an additional system [5] to be used as a secondary discriminator. We have chosen to utilize information from wireless ethernet networks, which are becoming very common in urban environments. A profile of the signal strengths of nearby access points is constructed and then used for matching with an existing database.

The combination of these two systems allows us to topologically localize the robot with good accuracy. To test the system, we took readings in 13 different regions throughout the northern half of the Columbia Campus and used each localization method alone and then the combination of both methods. On average, the correct region was identified by our combined method 89% of the time.

In addition, we have used our wireless signal based system to generate a map of the signal coverage (see Fig. 12) on our test bed, the Columbia campus. We might wish to choose our paths and scanning locations such that we favor regions of good wireless ethernet coverage, since that would allow for improved topological localization.

4.3 Scan Acquisition and Integration

Using the AVENUE system's localization and navigation, the robot moves to each of the precomputed viewpoints. At each viewpoint, it will trigger a scanning operation

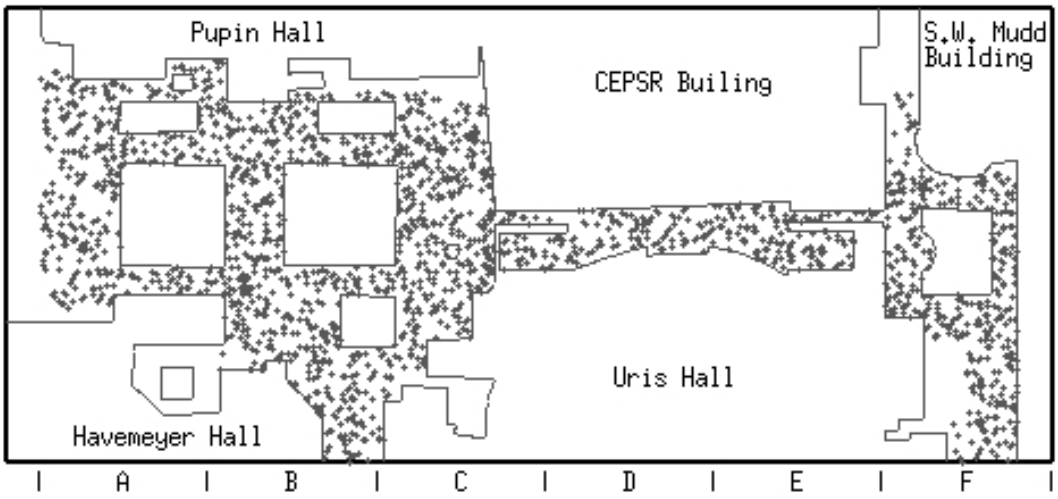


Figure 12: A map of the locations at which wireless signals were measured in a region of the Columbia campus. The subregions A, B, C, D, E, and F extend in the north-south direction for approximately 220 feet and are each about 80 feet wide.

that results in a point cloud of data (see Fig 13).

These data sets will be used to generate meshes which form the environment model. Meshes will be merged with one another using a combination of the robot's own estimate of its location at which a scan was taken and existing registration techniques, including ICP, when there is sufficient overlap between two meshes; and merging techniques such as Curless and Levoy's VripPack system [9].

5 Final Modeling Phase

After the initial modeling phase has been completed, we have a preliminary model of the environment. This model will have many holes in it caused by originally undetectable occlusions. We now implement a view planning system that makes use of this model to efficiently plan further views.

Unlike the first modeling phase, this system does not plan all of its views at once. Instead, it takes its initial three-dimensional model of the environment and plans a single next best view that will acquire what we estimate to be the largest amount of new information possible, given the known state of the world. Once this scan has been acquired, the new data are integrated into the model of the world using registration

Figure 13: A sample scan taken from the AVENUE system. The hole at the center of the scan is where the scanner was positioned. The parts of the scan that appear striped are actually regions where the grazing angle between the laser and the surface was very high, causing loss of resolution.

techniques similar to those used in the first modeling phase described earlier.

Our method for the second modeling stage requires a different data representation than just a simple point cloud or mesh. We need a way to tell what parts of the scene have been imaged at all and what have not. To do this, we maintain a second representation of the world. This new representation keeps track of seen-empty, seen-occupied, and unseen portions of the region to be imaged. This would most easily be represented by a voxel map. Because this is a large scale imaging problem, the voxels could be made rather big and would still satisfy their purpose. A voxel size of one meter cubed will be sufficient to allow for computing occlusions in the views.

The voxel representation is generated from the point cloud. Initially all voxels in the grid are labeled as unseen. When a new scan is taken, voxels that contain at least

one data point from that scan are marked as seen-occupied (even if a previous scan had labeled a voxel as seen-empty). For a data-point to have been acquired, there must have been an unoccluded line of sight between the camera position and that data-point. A ray is then traced from each data-point back to the camera position. Each unseen voxel that it crosses is marked as seen-empty. If the ray passes through a voxel that had already been labeled as seen-occupied, it means that the voxel itself may already have been filled by a previous scan or another part of the current scan. This means that the voxel itself is only partially occupied and we allow the ray to pass through it without modifying its status as seen-occupied.

Our approach to the final modeling phase takes its cue from Pito's work [29], in which a grid of cells called the "positional space" is arranged around the object to be modeled. In Pito's work, the objects being imaged are small and the sensor is free to move anywhere around the object. He considers only patches of unknown information at the boundary of his current mesh and projects rays back from them into the positional space. Pito chooses the cell in the positional space that views the largest number of these unknown patches as the next view.

We extend this idea to a voxel-based representation. In our case, we are restricted to operating on the ground plane with our wheeled robot. We can exploit the fact that we have a reasonable two-dimensional map of the region. This 2-D map gives us the foot prints of the buildings as well as a good estimate of the free space on the ground plane in which we can operate. We mark the cells on this ground plane which are within the free space defined by our 2-D map as being candidate views. We can then use these marked voxels on the ground plane as our version of the "positional space."

We wish to choose a location on this ground plane grid that maximizes the number of unseen voxels that can be viewed from a single scan. Considering every unseen voxel in this procedure is unnecessarily expensive and should be avoided. At the end of the first stage of view planning, much of the environment will already have been imaged and many of the "unseen" voxels will actually be regions in the interior of buildings that, without actually venturing into those buildings, can never be discovered by our system. Instead, we need to focus on those unseen voxels that are most likely to provide us with useful information about the faces of the buildings. These useful voxels are the ones that fall on the boundaries between seen-empty regions and unseen regions. These boundary regions are most likely to contain previously occluded structures and are similar to the "occlusion planes" discussed in the work of Massios and Fisher [22]. If an unseen voxel is completely surrounded by seen-occupied voxels or even other unseen voxels, then there is a good chance that it may never be visible by any scan. As in Massios and Fisher, we therefore choose to consider unseen

voxels that are adjacent to at least one seen-empty voxel.

Now that we have a set of unseen voxels to consider, we proceed with the optimization. We keep a tally at each position on our ground plane grid of the number of unseen voxels that can be seen from that position. Each ground plane voxel's tally starts at 0 and is incremented by 1 for every unseen voxel that it can view.

To determine whether an unseen voxel can be viewed, we trace rays from its center to the center of each voxel on the ground plane. For each of these rays we must check that it satisfies two initial constraints:

1. The Occlusion Constraint – If the ray intersects any voxel that is seen-occupied, we discard the ray because it will be occluded by the contents of that occupied voxel. In addition, if the ray intersects any voxel that is unseen, we discard the ray because we are uncertain of the contents of that voxel and it is still possible that it will be occluded.

2. The Range Constraint – We must consider the minimum and maximum range of the scanner. If the length of the ray is outside the scanner's range, then we discard the ray.

If these constraints are satisfied, we can safely increment the ground plane grid that the ray intersects.

At the end of this calculation, the ground plane voxel with the highest tally is chosen as the robot's location to take the next scan. The robot then plans a path and navigates to the chosen position. It triggers a new scan once it has arrived, and that scan is integrated into both the mesh model and the simplified voxel model. This entire second stage is then repeated until we reach a sufficient level of coverage of the site. To decide when to terminate the algorithm, we look at the number of unknown voxels that would be resolved by the next iteration. If that number falls below some small threshold value, then the algorithm terminates; otherwise, it continues.

As an initial test of the algorithm, we ran it on simulated test data. In Fig. 14, we see a 2-D projection of the 50 voxel long by 50 voxel wide by 10 voxel high grid used in these tests. Two initial scans were simulated and then the algorithm was applied. The first two next-best-views chosen are illustrated in Fig. 14, along with the final set of views chosen by this iterative algorithm.

There is, however, more to be considered for the second modeling phase. There are a number of other constraints that we could use to narrow down our possible views. The view planning problem is essentially an optimization problem with a variety of variables that we might want to control. We need each view to acquire data from

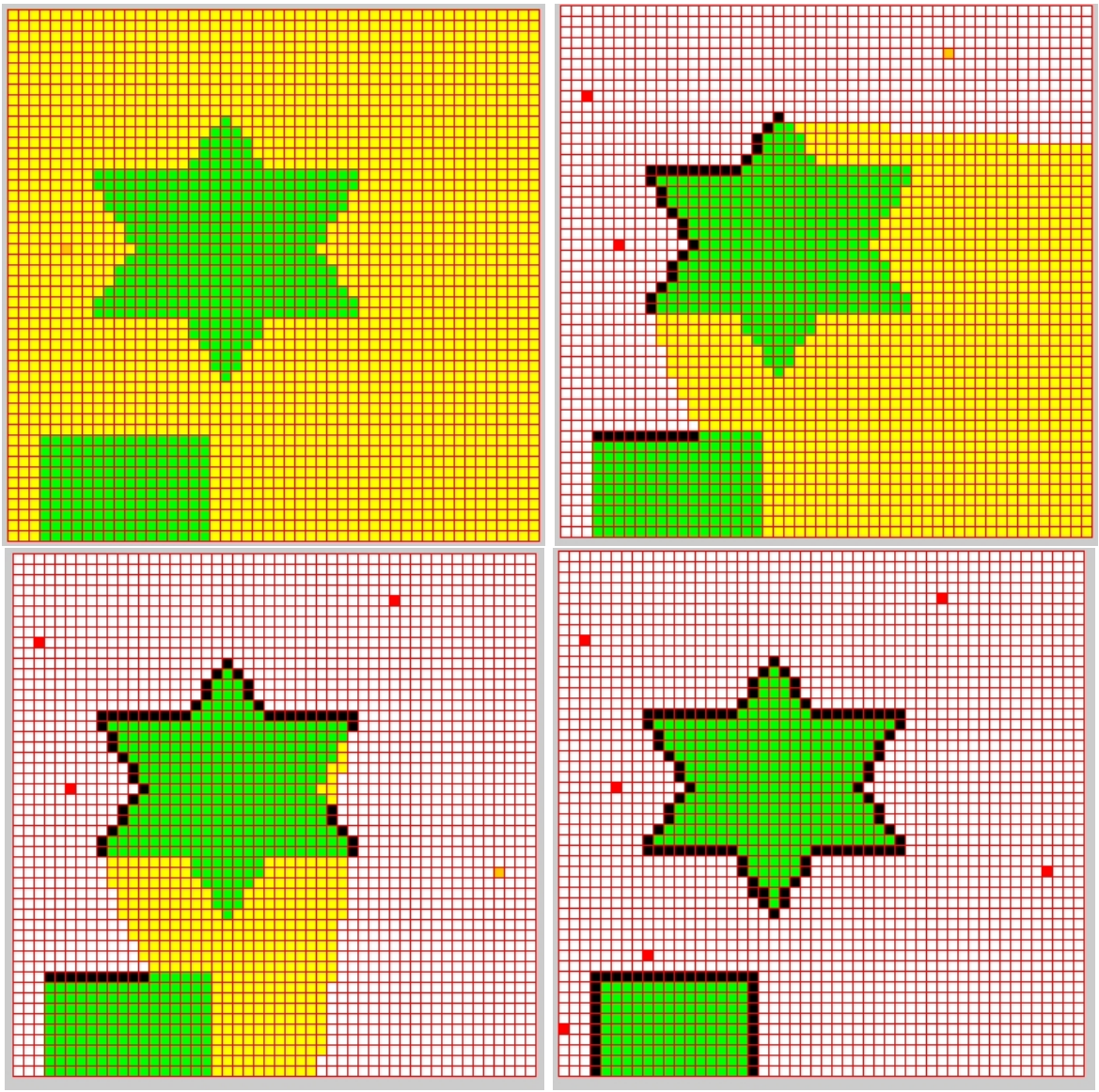


Figure 14: A test run of the basic voxel-based planning algorithm. All images are 2-D projections of the voxel map on the ground plane. Yellow and green areas are cells marked unseen, with green cells indicating the footprints of buildings for reference purposes. Black cells have been marked as seen-filled, and white cells have been marked as seen-empty. Red cells are viewpoint locations of earlier scans, and orange cells indicate the currently chosen next best view. The footprints of the test buildings (top left). Voxels after the initial two scans of the scene, along with the chosen next best view (top right). Voxels after the third scan, along with the chosen next best view (bottom left). Voxels after the algorithm terminates (bottom right).

as many of the unknown portions of the model as possible, but we also want a fast process. The two major remaining constraints to implement are:

1. The Grazing Angle Constraint – If the grazing angle between a ray and the surface that we expect at an unseen voxel is larger than the maximum angle allowed by our sensor, we must discard the ray. Since, by definition, these unseen voxels are unknown, we do not have a good idea of what the surface normal at that point would be. To get around this, we will look at the local area of voxels surrounding the unseen voxel in question. We can construct a plane that on average divides the unseen voxels in that local region from the seen-empty voxels. This plane is an estimate of the surface that divides the unseen region from the seen-empty region, and its normal can be used to compute an estimated grazing angle.

2. The overlap constraint – Depending on the registration method that we use to integrate new scans into the model, we may need to have a good amount of overlap between each new scan and the existing model. In this case, we would add an additional weighted score to each potential scanning location that keeps track of the fraction of the existing model that is visible from that location.

There are other possible constraints. To aid in localization, we might wish to optimize the position of the robot's views by placing them in well-known regions with good GPS coverage. Another possible constraint involves more heavily weighting potential viewpoints that are closer to the current viewpoint, in order to prevent the robot from traveling long distances between scans and thereby keep cumulative error down. Finally, we might wish to use an even more detailed weighted sum for evaluating our candidate viewing locations on the ground plane. For example, we might decide to more heavily weight a view of unseen voxels that have smaller grazing angles rather than just specifying a hard threshold for the maximum angle.

It is important to mention that there are other approaches to this kind of sequential view planning. Since this portion of our research is in its early stages, it is possible that we may explore these other approaches. The work of González-Baños et al [16, 17] and Nüchter et al [25] also look at the boundary regions between empty and unknown portions of the scene. However, both use only range data at the ground plane to detect these boundaries, effectively making their work a two-dimensional planning algorithm. Work by Sequeira et al ([21, 33]) is somewhat similar to our approach to the final phase of modeling, though their work is only a single stage algorithm. They maintain two meshes, the known surface mesh and the void mesh.

The void mesh is the boundary between the known world and the unexplored void volume. It is this void mesh that their algorithm targets.

6 Road Map to the Thesis

In this thesis proposal, we have presented the plans for an integrated system which automates the site modeling process. Significant progress has already been made on all components of the system. However, several components need further development, the various components must be integrated together, and extensive testing on complex environments should be conducted. The following list indicates the major parts of our system and their current status.

1. A topological localization algorithm using a combination of omnidirectional vision and wireless access point signals has already been developed and used to localize the robot on a two-dimensional map and to assist the navigation through planned paths. This system has been tested in complicated outdoor environments and is now operational.

2. A Voronoi-based path planner has already been fully implemented to navigate the robot. This path planner is used to generate safe paths between particular start and goal locations as well as to generate safe and efficient tours through an entire set of initially chosen view points.

3. An improved two-dimensional view planning algorithm has been developed to include the constraints of a large class of laser scanners. This system computes an initial set of views for our sensor by treating the problem as two-dimensional rather than three-dimensional and by approaching it as an art gallery style problem. Our algorithm includes the important constraints of minimum and maximum range, maximum grazing angle, and limited field of view. This part of the project has been successfully tested on both simulated and real-world data. Additional constraints, such as the overlap constraint, must still be added to the system.

4. A theoretical voxel-based method for choosing next-best-views based on the current set of acquired three-dimensional scans is being developed. This will allow us to bootstrap the creation of our model: with each additional scan, our choice of future views improves. This part of the system has not yet been completed.

At this point, there is still important work which remains to be done. The primary focus will be on improving the voxel-based next-best-view algorithm, which is currently at an early stage of demonstrating its viability. This algorithm must use more sophisticated constraints when choosing its views and must have its efficiency improved because of the large voxel grids involved. In addition to the next-best-view algorithm, there are a number of system integration issues that need to be addressed. Primarily, we still need to integrate the voxel-based next-best-view planner into the robot's main control system. Once this is done, we can more fully implement the bootstrapped model construction. In the work done so far, we have been relying on the robot's position estimates to register the multiple scans. This will most likely be insufficient for the final model construction, so a registration component will have to be added into the system. Finally, we need to do extensive testing of the system on a variety of sites. Up to now, some algorithms have been run mostly on simulated data. Tests on real-world environments may force us to make adjustments.

The mobile robot system presented in this proposal combines two-dimensional and three-dimensional methods and is applicable to large outdoor structures. We believe the final system to be developed in this thesis will make a significant contribution to the automated modeling of real-world environments.

References

[1] Peter K. Allen, Ioannis Stamos, Atanas Gueorguiev, Ethan Gold, and Paul Blaer. Avenue: Automated site modeling in urban environments. In *International Conference on 3-D Digital Imaging and Modeling*, pages 357–364, 2001.

[2] J. E. Banta, Y. Zhien, X. Z. Wang, G. Zhang, M. T. Smith, and M. A. Abidi. A "best-next-view" algorithm for three-dimensional scene reconstruction using range images. *Proceedings of SPIE (Intelligent Robots and Computer Vision XIV: Algorithms, Techniques, Active Vision, and Materials Handling)*, 2588:418–429, October 1995.

[3] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analyis and Machine Intelligence*, 14(2):239–256, Feb 1992.

[4] Paul Blaer and Peter K. Allen. Topological mobile robot localization using fast vision techniques. In *IEEE International Conference on Robotics and Automation, 2002*, pages 1031–1036, 2002.

[5] Paul Blaer and Peter K. Allen. Topbot: automated network topology detection with a mobile robot. In *IEEE International Conference on Robotics and Automation, 2003*, pages 1582–1587, 2003.

[6] C. S. Chen, Y. P. Hung, and J. B. Cheung. Ransac-based darces: a new approach to fast automatic registration of partially overlapping range images. *IEEE Transactions on Pattern Analyis and Machine Intelligence*, 21(11):1229–1234, Nov 1999.

[7] C. I. Connolly. The determination of next best views. In *IEEE International Conference on Robotics and Automation, 1985*, pages 432–435, 1985.

[8] Cregg K. Cowan and Peter D. Kovesi. Automatic sensor placement from vision task requirements. *IEEE Transactions on Pattern Analyis and Machine Intelligence*, 10(3):407–416, May 1988.

[9] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[10] Tim Danner and Lydia E. Kavraki. Randomized planning for short inspection paths. In *IEEE International Conference on Robotics and Automation, 2000*, pages 971–978, 2000.

[11] Shun en Xie, Thomas W. Calvert, and Binay K. Bhatacharya. Planning views for the incremental construction of body models. In *IEEE International Conference on Pattern Recognition*, pages 154–157, 1986.

[12] S. J. Fortune. A sweepline algorithm for Voronoi diagrams. In *Algorithmica 2 (1987)*, pages 153–174.

[13] Christian Frueh and Avideh Zakhor. Constructing 3d city models by merging ground-based and airborne views. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003*, pages 562–569, 2003.

[14] Atanas Georgiev and Peter K. Allen. Vision for mobile robot localization in urban environments. In *IEEE International Conference on Intelligent Robots and Systems*, pages 472 – 477, 2002.

[15] H. González-Baños and J. C. Latombe. A randomized art-gallery algorithm for sensor placement. In *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 232–240, 2001.

[16] H. González-Baños, E. Mao, J. C. Latombe, T. M. Murali, and A. Efrat. Planning robot motion strategies for efficient model construction. In *International Simposium of Robotics Research*, pages 345–352, 1999.

[17] H. H. González-Baños, L. Guibas, J. C. Latombe, S. M. LaValle, D. Lin, R. Motwani, and C. Tomasi. Motion planning with visibility constraints: Building autonomous observers. In *The Eighth International Symposium of Robotics Research*, 1997.

[18] Jacob E. Goodman and Joseph O'Rourke. *Handbook of discrete and computational geometry*. CRC Press, Inc., 1997.

[19] Robert Grabowski, Pradeep Khosla, and Howie Choset. Autonomous exploration via regions of interest. In *IEEE International Conference on Intelligent Robots and Systems*, pages 27–31, 2003.

[20] Atanas Gueorguiev, Peter K. Allen, Ethan Gold, and Paul Blaer. Design, architecture and control of a mobile site modeling robot. In *IEEE International Conference on Robotics and Automation, 2000*, pages 3266–3271, 2000.

[21] Konrad Klein and Vítor Sequeira. View planning for the 3d modelling of real world scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000.

[22] Nikolaos A. Massios and Robert B. Fisher. A best next view selection algorithm incorporating a quality criterion. In *British Machine Vision Conference*, 1998.

[23] Jasana Maver and Ruzena Bajcsy. Occlusions as a guide for planning the next view. *IEEE Transactions on Pattern Analyis and Machine Intelligence*, 15(5):417–433, May 1993.

[24] Shree Nayar. Omnidirectional video camera. In *DARPA Image Understanding Workshop*, pages 235–242, May 12-14 1997.

[25] Andreas Nüchter, Hartmut Surmann, and Joachim Hertzberg. Planning robot motion for 3d digitalization of indoor environments. In *International Conference on Advanced Robotics*, pages 222–227, 2003.

[26] Kazunori Ohno, Takashi Tsubouchi, and Shin'ichi Yuta. Outdoor map building based on odomoetry and rtk-gps position fusion. In *IEEE International Conference on Robotics and Automation, 2004*, pages 684–690, 2004.

[27] Atsuyuki Okabe, Barry Boots, and Kokichi Sugihara. *Spatial tessellations: concepts and applications of Voronoi diagrams*. John Wiley & Sons, Inc., 1992.

[28] Joseph O'Rourke. *Art gallery theorems and algorithms*. Oxford University Press, Inc., 1987.

[29] Richard Pito. A solution to the next best view problem for automated surface acquisition. *IEEE Transactions on Pattern Analyis and Machine Intelligence*, 21(10):1016–1030, October 1999.

[30] Michael K. Reed and Peter K. Allen. Constraint-based sensor planning for scene modeling. *IEEE Transactions on Pattern Analyis and Machine Intelligence*, 22(12):1460–1467, December 2000.

[31] William R. Scott, Gerhard Roth, and Jean-François Rivest. View planning for automated three-dimensional object reconstruction and inspection. *ACM Computing Surveys*, 35(1):64–96, 2003.

[32] William R. Scott, Gerhard Roth, and Jean-François Rivest. View planning for multistage object reconstruction. In *International Conference on Vision Interface*, 2001.

[33] Vítor Sequeira and Jo ao G.M. Gonçalves. 3d reality modelling: Photo-realistic 3d models of real world scenes. In *First International Symposium on 3D Data Processing Visualization and Transmission*, 2002.

[34] Gilbert Soucy, Francesco G. Callari, and Frank P. Ferrie. Uniform and copmlete surface coverage with a robot-mounted laser rangefinder. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1682–1688, 1998.

[35] Konstantinos Tarabanis and Roger Y. Tsai. Computing occlusion-free viewpoints. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1992*, pages 802–806, 1992.

[36] Konstantinos A. Tarabanis, Peter K. Allen, and Roger Y. Tsai. A survey of sensor planning in computer vision. *IEEE Transactions on Robotics and Automation*, 11(1):86–104, February 1995.

[37] Konstantinos A. Tarabanis, Roger Y. Tsai, and Peter K. Allen. The mvp sensor planning system for robotic vision tasks. *IEEE Transactions on Robotics and Automation*, 11(1):72–85, February 1995.

[38] G. H. Tarbox and S. N. Gottschlich. Planning for complete sensor coverage in inspection. *Computer Vision and Image Understanding*, 61(1):84–110, January 1995.

[39] Seth Teller. Automatic acquisition of hierarchical, textured 3d geometric models of urban environments: Project plan. In *Procedings of the Image Understanding Workshop*, 1997.

[40] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318, 1994.

[41] Peter Whaite and Frank P. Ferrie. Autonomous exploration: Driven by uncertainty. *IEEE Transactions on Pattern Analyis and Machine Intelligence*, 19(3):417–433, March 1997.

[42] Hongbin Zha, Ken'ichi Morooka, Tsutomu Hasegawa, and Tadashi Nagata. Active modeling of 3-d objects: Planning on the next best pose (nbp) for acquiring range images. In *International Conference on 3-D Digital Imaging and Modeling*, pages 68–75, 1997.