View Planning and Automated Data Acquisition for Three-Dimensional Modeling of Complex Sites

Paul S. Blaer and Peter K. Allen
Department of Computer Science
Columbia University
New York, New York 10027
e-mail: pblaer@cs.columbia.edu, allen@cs.columbia.edu

Received 7 January 2009; accepted 30 July 2009

Constructing highly detailed three-dimensional (3-D) models of large complex sites using range scanners can be a time-consuming manual process. One of the main drawbacks is determining where to place the scanner to obtain complete coverage of a site. We have developed a system for automatic view planning called VuePlan. When combined with our mobile robot, AVENUE, we have a system that is capable of modeling large-scale environments with minimal human intervention throughout both the planning and acquisition phases. The system proceeds in two distinct stages. In the initial phase, the system is given a two-dimensional site footprint with which it plans a minimal set of sufficient and properly constrained covering views. We then use a 3-D laser scanner to take scans at each of these views. When this planning system is combined with our mobile robot it automatically computes and executes a tour of these viewing locations and acquires them with the robot's onboard laser scanner. These initial scans serve as an approximate 3-D model of the site. The planning software then enters a second phase in which it updates this model by using a voxel-based occupancy procedure to plan the next best view (NBV). This NBV is acquired, and further NBVs are sequentially computed and acquired until an accurate and complete 3-D model is obtained. A simulator tool that we developed has allowed us to test our entire view planning algorithm on simulated sites. We have also successfully used our two-phase system to construct precise 3-D models of real-world sites located in New York City: Uris Hall on the campus of Columbia University and Fort Jay on Governors Island.

1. INTRODUCTION

Accurate three-dimensional (3-D) models of large complex sites such as buildings and their surroundings have many uses, including reverse engineering of legacy structures that lack computer-aided design (CAD) models or detailed engineering drawings, visualization and walk-through capability for remote structures, detailed documentation of historic buildings, and temporal change detail of sites over time such as archaeological ruins.

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 use a theodolite to take measurements of the structure and then have a designer manually put together a model from those measurements. Such a model might look

Journal of Field Robotics 26(11–12), 865–891 (2009) © 2009 Wiley Periodicals, Inc.
Published online in Wiley InterScience (www.interscience.wiley.com). • DOI: 10.1002/rob.20318

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 approximation.

More sophisticated tools do exist. There are a number of laser range scanners that will sweep a beam across a large portion of the structure and return a dense point cloud of measurements. Armed with these more advanced instruments, one can take multiple scans around the structure and register them into a single point cloud that accurately represents the structure. Software tools allow the point cloud to 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.

Although the models are now far more accurate and the acquisition process is faster and more automated, there are still manual choices and interventions involved in modeling large, complex sites. View planning is one such component, in which a plan must be laid out to determine where to take each individual scan that comprises the complete model. 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. The views must also satisfy the constraints of resolution and field of view of the scanning system. This is the essence of the so-called view planning problem. Historically, it has been solved either by manually choosing a set of views, which usually results in missing surfaces of the site, or by oversampling the site and hoping the samples are sufficient. Each scan can be very time-consuming, as the scanning sensor must be physically moved from location to location, and each scanning operation itself can take an hour or more depending on the type of sensor and the density of the scan.

We have developed a software system called VuePlan that can help automate the view planning process. VuePlan is a two-stage view planning algorithm that can automatically decide where to acquire scan data to obtain complete coverage of a site in an efficient way. We have also integrated the view planning process with our mobile robot scanning platform, AVENUE (Allen, Stamos, Gueorguiev, Gold, & Blaer, 2001) (see Figure 1). AVENUE is a laser scanner–equipped robot capable of localizing and navigating itself through various environments. The view planning system is added to our previously developed localization (Georgiev & Allen, 2004) and low-level navigation software, allowing the robot to implement its plan and acquire a model of the location with minimal human assistance. The system works by taking some basic approximate information about the site, planning a path to a desired viewpoint, navigating the mobile robot to that viewpoint, acquiring images and 3-D range scans of the site, and then planning for the next viewpoint until the model is complete. In addition, we have developed a convenient simulation tool that has allowed us to test the entire view planning algorithm on simulated structures and environments. We have also successfully used our two-phase system to construct accurate 3-D models of real-world sites located in New York City: Uris Hall on the campus of Columbia University and Fort Jay on Governors Island.

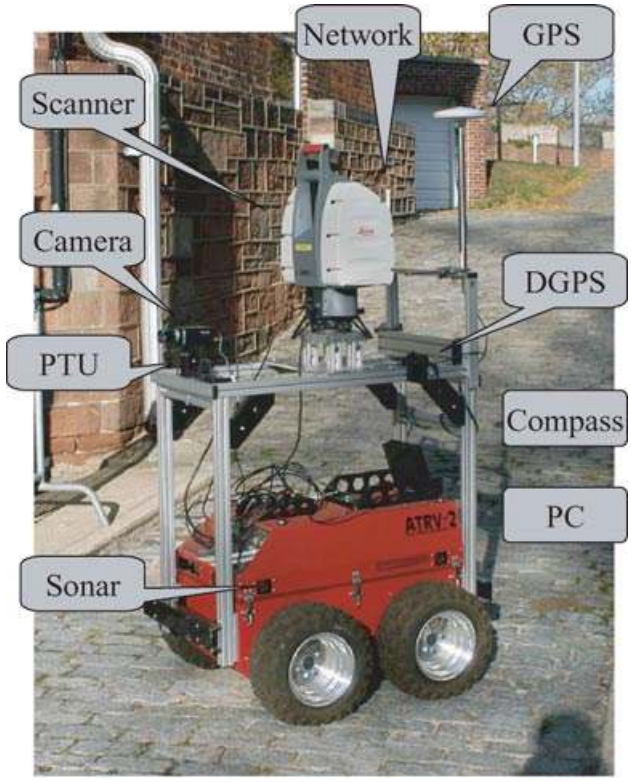

Figure 1. The ATRV-2 AVENUE-based mobile robot.

2. RELATED WORK

Currently a number of research projects are attempting to construct 3-D models of urban scenes and outdoor structures. These projects include the 3-D city model construction project at Berkeley (Frueh & Zakhor, 2003), the outdoor map building project at the University of Tsukuba (Ohno, Tsubouchi, & Yuta, 2004), the MIT City Scanning Project (Teller, 1997), the volumetric robotic mapping project by Thrun et al. (2003), the 4D Cities project at Georgia Tech (Dellaert, 2005), and the UrbanScape project (Akbarzadeh et al., 2006). However, for the most part, these methods focus on data fusion issues and leave the actual planning component to a human operator. Jensen, Weingarten, Kolski, and Siegwart (2005), Lamon, Kolski, and Siegwart (2006), and Pfaff et al. (2007) also address the large-scale mapping problem by mounting a series of SICK laser scanners on a small car that is then driven through the environment. The works focus on registration and localization issues but also include an automated path planning component. The planning component focuses on navigating through traversable areas but still requires the user to specify the final destination; the view planning problem is not addressed.

Ikeuchi, Nakazawa, Hasegawa, and Ohishi (2003) concentrate on methods for digitizing, registering, merging, and texture mapping large-scale models of cultural heritage sites. Data are acquired primarily through manually placed scanning and imaging equipment, and the focus is on automatic techniques to merge and correct the acquired data. This group has done work constructing models of a number of heritage sites such as the Great Buddha of Kamakura (Ikeuchi, et al., 2003) and, more recently, the Angkor Tom in Cambodia (Ikeuchi & Miyazaki, 2007).

The view planning problem can be described as the task of finding a set of sensor configurations that efficiently and accurately fulfill a modeling or inspection task. Two major surveys on the topic exist, including an earlier survey on computer vision sensor planning by Tarabanis, Allen, and Tsai (1995) and a more recent survey of view planning specifically for 3-D vision by Scott, Roth, and Rivest (2003).

Model-based methods are inspection methods in which the system is given some initial model of the scene. Early research focused on planning for two-dimensional (2-D) camera-based systems. Included in this are works by Cowan and Kovesi (1998) and by Tarabanis, Tsai, and Allen (1995). Later, these methods were extended to the 3-D domain in works by Tarbox and Gottschlich (1995) and by Scott, Roth, and Rivest (2001). We can also include art gallery problems in this category. In two dimensions, these problems can be approached with traditional geometric solutions such as in Xie, Calvert, and Bhatacharya (1986) or with randomized methods such as in González-Banos and Latombe (2001). The art gallery approach has also been applied to 3-D problems by Danner and Kavraki (2000). In this case a complete 3-D model of the scene is known beforehand.

Non-model-based methods seek to generate models with no prior information. These include volumetric methods such as in Banta et al. (1995), Connolly (1985), Low and Lastra (2006), Massios and Fisher (1988), and Soucy, Callari, and Ferrie (1998). There are also surface-based methods, which include those of Maver and Bajcsy (1993), Pito (1999), Reed and Allen (2000), and Sequeira and Gonçalves (2002). A different appoach is taken by Whaite and Ferrie (1997), who use the error in a parametric fit to improve the fit of the model and thereby drive the view planning process. View planning for 2-D map construction with a mobile robot is addressed by González-Baños, Mao, Latombe, Murali, and Efrat, (1999) and Grabowski, Khosla, and Choset (2003).

In the work that most closely resembles ours, Nüchter, Surmann, and Hertzberg (2003) address view planning for 3-D scenes with a mobile robot. Next best views (NBVs) are chosen by maximizing the amount of 2-D information (on the ground plane only) that can be obtained at a chosen location; however, 3-D data are actually acquired. These authors 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. This has some similarity to our first-stage algorithm, but unlike our second-stage algorithm, 3-D data are never used in the NBV computation.

3. VUEPLAN: A VIEW PLANNING SYSTEM FOR 3-D MODELING

There are many possible paradigms for mobile robot exploration and mapping of complex sites. One method involves making use of a fast scanner that allows real-time scanning as the robot or scanning vehicle moves quickly throughout the environment.

This presents a trade-off. To gain the ability to scan quickly, one has to sacrifice resolution. Our particular interest is in the construction of dense and detailed models of these environments, which is not possible with lower resolution scanners. Our scanning equipment is designed to maximize detail, and this causes scan times to be long and forces us to remain stationary during a scan. Under these conditions, view planning first, before taking a scan, is the most practical approach.

The overall goal of our system is to construct an accurate 3-D model of a complex site. We have created a system called VuePlan that automates the construction of the model. The 3-D model construction process proceeds in two stages. The first stage of the modeling process utilizes a 2-D map to construct an initial model of the scene and an initial set of views. The second stage uses 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 are able to minimize the number of high-resolution scans needed to construct the model. Our initial run through the environment uses views planned only from the 2-D map (Blaer & Allen, 2006a). 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 have some 3-D data about the environment's structures and how they occlude the scene. We then make use of the data to plan unoccluded views for further modeling. With each data set acquired, we have more information to help plan more-efficient subsequent views.

It could be argued that requiring a 2-D ground map for environment exploration is too restrictive. For our work, however, the scenes in which we have the most interest, historical and urban sites, 2-D ground maps already exist and are quite ubiquitous. Nevertheless, we have explored alternative methods that utilize only the second stage of the algorithm (Blaer & Allen, 2006b) even though this stage is slow relative to the 2-D planner. By using the 2-D planner first, we reveal most of the scene in advance, greatly decreasing the amount of unknown volume that the 3-D algorithm must examine.

3.1. Phase I: Initial Model Construction

In the first stage of our modeling process, we wish to compute and acquire an initial model of the target region. This model will be based on limited information about the site and will most likely have gaps in the data that must be filled in during the second stage of the process. The data acquired in the initial stage will serve as a seed for the boostrapping method used to complete the model.

The procedure for planning the initial views makes use of a 2-D ground map of the region to plan a series of environment views for our scanning system to acquire. All scanning locations in this initial phase are planned in advance, before any data acquisition occurs.

We wish to find a set of positions for our scanner such that it can image all of the known walls in our 2-D map of the environment. This view planning strategy makes the simplifying assumption that if we can see the 2-D footprint 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.

Planning these locations resembles the classical art gallery problem, which asks where to optimally place guards such that all walls of the art gallery can be seen by the set of guards. This assumes that the guards can see all the way around their location, that is, they have a 360-deg field of view. It also assumes that the guards have an unlimited distance of vision and that they can view a wall at any grazing angle. None of these assumptions is true for most laser scanning systems, so the traditional methods do not apply exactly to our problem.

3.1.1. View Planning with Constraints

In our view planning algorithm we have extended the work of González-Banos and Latombe (2001) to include the constraints of minimum and maximum range, grazing angle, field of view, and scan overlap. We start with a set of initial scanning locations that 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 outlined above. Finally, an approximation for the optimal number of viewpoints needed to cover the boundaries of the free space is computed from this set of initial locations.

We begin with a 2-D floor plan or map of the environment to be modeled. We first digitize this map

and then choose an initial set of N random scanning locations. Next, the visibility of each of the N viewpoints is computed. We use the ray-sweep algorithm (Goodman & O'Rourke, 1997) to compute the visibility polygon. This polygon has two types of edges. The first contains the obstacle edges that are on the boundary of the region's free space. The second contains intermediate edges that 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.

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 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. Most range sensors lose accuracy at grazing angles larger than about 70 deg when the laser light is not reflected back to the sensor. In Figure 2 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 deg) for all points on the subsegment.

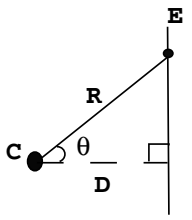

Figure 2. The grazing angle constraint.

We also have the ability to constrain the field of view of the scanner. To make use of this constraint, we must modify the initial step of our algorithm in which we distribute the random potential viewpoints. Because in this case the scanner can see only a fixed field of view, we must also specify a heading for each of our potential positions. To accomplish this, we randomly assign a heading to each potential viewpoint in addition to its location (forcing us to choose more potential viewpoints to consider). This constraint allows our algorithm to be flexible and work with scanning systems that have an arbitrary field of view.

Once these constraints have been computed, we use a greedy cover algorithm to select an approximation for the minimum number of viewpoints needed to cover the scene. We select viewpoints based on the total length of the polygonal edges that can be viewed with constraints from that viewpoint. We then eliminate from consideration those edges that have already been seen and then repeat the process for the next viewpoint until all edges have been seen or we have reached a threshold percentage of coverage of the scene.

We can also enforce an overlap constraint using this algorithm. 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 more-precise registration techniques. There are several potential methods to be used. One possibility is to distribute fiducial marks throughout the scene that can easily be detected by the scanning equipment. By matching corresponding fiducial marks in one scan to the next, we can very tightly register the two scans. Another possibility is to utilize an automatic registration method that does not need known landmarks in the scene. Commonly used registration methods include iterative closest point (ICP) methods such as those contained in Besl and McKay (1992), Chen, Hung, and Cheung (1999), and Turk and Levoy (1994). ICP algorithms begin by associating points in one scan with points in another scan, typically using a nearest neighbor scheme, and then estimating the

transformation that would bring those associated points to the same positions. Using this new registration between the two point clouds, the algorithm repeats until the error in the estimated transformation drops below a certain threshold. To perform the initial association of points between two scans, an initial estimate of the proper registration must be provided. The localization component of our robot system (Georgiev & Allen, 2004) is typically sufficient for such an initial estimate, and other work in our lab (Stamos & Allen, 2002) has addressed this problem.

Both the fiducial mark methods and the ICP methods for registration require a certain amount of overlap between one scan and the next. When using fiducial marks, there need to be at least three marks common to each scan. So, these marks must be placed in regions of overlap between scans. When using ICP, there needs to be a set of corresponding points within the two scans for the algorithm to proceed at all. As a result, we would like to choose viewpoints that have significant overlap in the areas that they cover, regardless of what method we ultimately choose for registration. To implement this overlap, we modify our greedy set cover. When we choose the next best viewing location (the location that sees the most new obstacle boundaries), we enforce a constraint that at least a fixed percentage of the edges that it sees overlap with the already seen portions of the boundary. If this constraint is not satisfied, we skip this potential viewing location and go to the NBV.

Our Phase I View Planning Algorithm starts with a 2-D map of the region of interest and incorporates sensor constraints and overlap constraints to obtain a set of final viewpoints from which to begin the modeling process. Phase I of our method is summarized in Algorithm 1.

Algorithm 1 The Phase I View Planning Algorithm. It must be given an initial 2-D map M to plan the initial views.

```
1: procedure 2DPLAN(M)
2:     Randomly distribute candidate views V in the map
3:     for all views v in V do
4:         Compute the visibility polygon p of v
5:         Clip p to sensor range
6:         Clip p to maximum grazing angle
7:         Clip p to field of view
8:     end for
9:     Find a greedy cover, G, from the potential views V,
           that enforces the overlap constraint.
10:    return G                          ▷ the planned views
11: end procedure
```

3.1.2. Phase I Example

As part of a larger 3-D modeling effort, we have been involved in building a database of 3-D models of the Romanesque churches of the Bourbonnais region of France, which can be viewed online (Romanesque Churches of the Bourbonnais, 2007). For our initial test of this algorithm, we used the Phase I planner on the interior of the church of Saint Menoux. We were given an existing floor plan of the church and then chose an initial set of 500 random scanning locations from this plan (see the left-most panel of Figure 3).

Next, we computed the visibility edges of each potential viewpoint and applied the constraints. In this example, we used a 360-deg field of view, which is consistent with our current scanning equipment (Leica HDS 3000). The visibility edges for one of the potential viewpoints, clipped for the range and grazing angle constraints, can be seen in Figure 3.

Finally, we applied the greedy cover to select the final viewpoints. For our test, we set a threshold such that the algorithm terminated if additional scans added less than 2% of the total boundaries of the target region. Our algorithm returned eight scanning locations for the test area (see the right of Figure 3), giving us a coverage of 95% of the region's obstacle boundary.

In Figure 4, we show floor-level slices of the resulting interior models of the church. Figure 5 shows two 3-D views of part of the interior model of the church. This model was generated entirely from the Phase I view planning scans. Even though holes exist in the model, the coverage is reasonably good. In the next section, we discuss Phase II, in which any remaining holes in the model can be resolved through further view planning.

3.2. Phase II: 3-D View Planning

After the initial modeling phase has been completed, we have a preliminary model of the environment. The model will have holes in it, many caused by originally undetectable occlusions. We now implement a 3-D view planning system that makes use of this initial 3-D model to plan efficiently for further views. This subsequent modeling phase does not plan all of its views at once. Instead, it takes the initial model and plans a single NBV that will gather what we estimate to be the largest amount of new information possible, given the known state of the world. This scan is acquired and the new data are integrated into the model of the world and the NBV is planned.

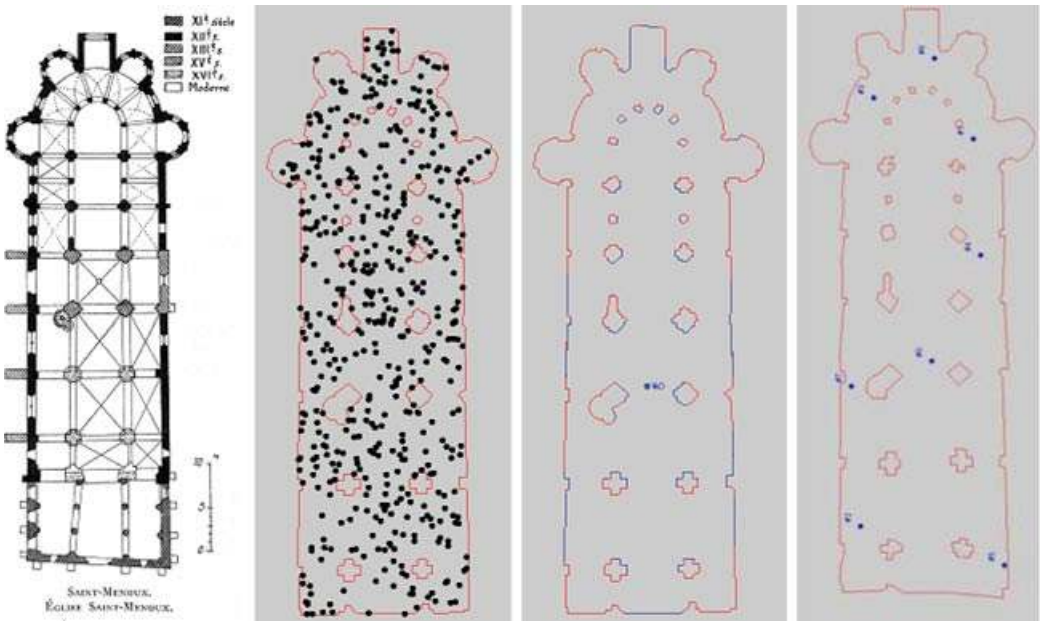

Figure 3. On the far left is the original floor plan of Saint Menoux's interior (Genermont & Pradel, 1938). Next is the digitized 2-D map of the interior (shown in red) with an initial set of 500 viewpoints randomly distributed throughout the free space of the region. Next is the set of clipped visibility edges for a potential scanning location (shown in blue). On the far right are eight scan locations determined by the Phase I view planner using a grazing angle constraint of 70 deg.

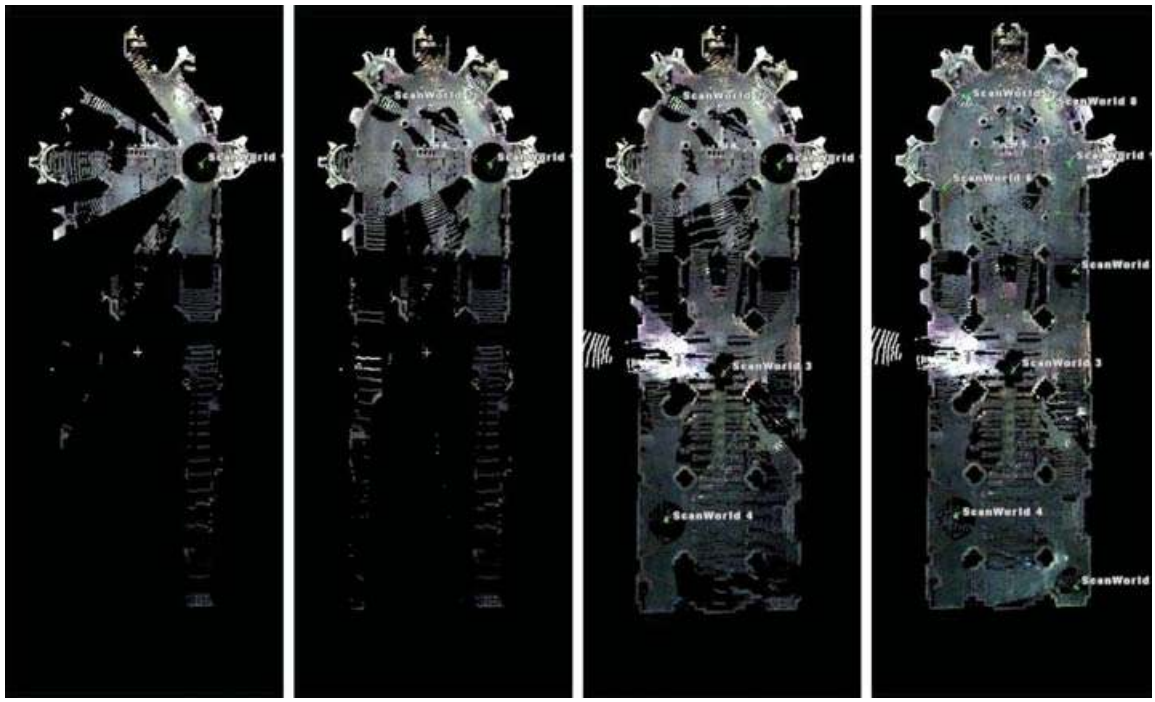

Figure 4. Slices taken at floor level of the actual acquired models of the church, using the Phase I planner. We show the model after the first, second, and fourth scans are acquired, respectively. The final image shows the complete model after eight scans.

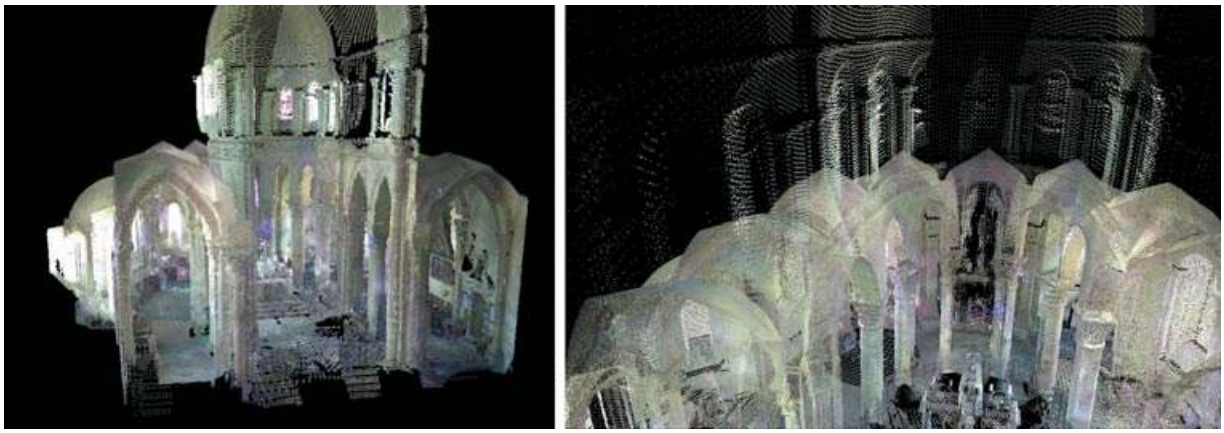

Figure 5. Two views of a portion of the 3-D model of the interior of the Church of Saint Menoux. This model was constructed with scans taken from the locations computed by our Phase I planning alogrithm.

3.2.1. Voxel Space

Our method for the 3-D modeling stage requires a data representation different from just the simple point cloud that we used in the initial stage. We need a way to tell what parts of the scene have been imaged and what parts have not. To do this, we maintain a second representation of the world that keeps track of *seen-empty*, *seen-occupied*, and *unseen* portions of the region to be imaged. This can most easily be represented by a voxel map (Curless & Levoy, 1996). Because this is a large-scale imaging problem, the voxels can be made rather large and still satisfy their purpose. Typically, a voxel size of 1 m^3 is sufficient to allow for computing occlusions in our views. Although the planner uses a reduced-resolution model of the data, the full data are still used in constructing the final model.

The voxel representation is generated from the point cloud. Before the Phase I model is inserted, all voxels in the grid are labeled as *unseen*. For each scan from the initial model, 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 any data point to have been acquired, there must have been an unoccluded line of sight between the scanner position and that data point. A ray is then traced from each data point back to the scanner 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*. Using this method, our Phase I model is inserted into the voxel space and we subsequently update this space.

3.2.2. NBV: Occlusion Constraint and Range Constraint

In Phase II we must now plan and acquire NBVs sequentially. In our case, we are restricted to operating on the ground plane with our mobile robot. We can exploit the fact that we have a reasonable 2-D map of the region. This 2-D map gives us the footprints 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 voxels that intersect this ground plane within the free space defined by our 2-D map as being candidate viewpoints.

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 our method, much of the environment has already been imaged and many of the *unseen* voxels will actually be regions in the interior of buildings. Instead, we need to focus on those *unseen* voxels that are most likely to provide us with useful information about the facades 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, in addition, are likely to be viewable by the scanner. If an *unseen*

voxel is completely surrounded by *seen-occupied* voxels or even by other *unseen* voxels, then there is a good chance that it may never be visible by any scan. We therefore choose to consider only *unseen* voxels that are adjacent to at least one *seen-empty* voxel. Such *unseen* voxels will be labeled as *boundary unseen* voxels.

Now that we have a set of appropriate *unseen* voxels to consider, we proceed with the optimization. As possible positions for the NBV, we use the centers of the voxels that intersect the ground plane within the region's free space. At each such position, we keep a tally of the number of *boundary unseen* voxels that can be seen from that position.

To determine whether a *boundary unseen* voxel can be viewed, we trace rays from its center to the center of each voxel on the ground plane. If the ray intersects any voxel that is *seen-occupied*, we discard the ray because it may be occluded by the contents of that occupied voxel. 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. We must also 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.

3.2.3. NBV: 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 should discard the ray. Because, by definition, these *unseen* voxels are unknown, we do not have a good idea of what the surface normal at that location would be. We can use the normal of the face of the *boundary unseen* voxel that borders on the *seen-empty* voxel as an estimate of the normal of the surface that is contained behind it. We can then use this normal to compute an estimated grazing angle.

3.2.4. Selecting the Next Best Viewpoint

If a ray has not been discarded by the occlusion, range, or grazing angle constraint, we can safely increment the ground plane position that the ray intersects. At the end of this calculation, the ground plane position with the highest tally is chosen as the next scan location. When using the mobile robot, it navigates to the chosen position and triggers a new scan once it has arrived. That scan is integrated into both the point cloud model and the voxel representation. The process 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 *boundary unseen* 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. The Phase II planner is summarized in Algorithm 2.

If the size of one dimension of the voxel space is n, then there could be $O(n^2)$ potential viewing locations. If there are m *boundary unseen* voxels, the cost of the algorithm could be as high as $O(n^2 * m)$. Note that m is typically rather small in comparison to the total number of voxels in the world. In addition, the n^2 term is an overestimate because many of the ground plane voxels are not potential viewpoints because they fall within known footprints of the obstacles. Furthermore, most *boundary unseen* voxels have only one face that is actually exposed to the known world. This face can be used to split the space in half. All potential scanning locations that are in the half-space behind the exposed face can be quickly excluded from consideration.

Algorithm 2 The Phase II 3-D View Planning Algorithm. It must be given an initial model of the world C, a set of possible scanning locations P, and a threshold value for the number of acceptable *unseen* voxels as a stopping condition.

```
1: procedure 3DPLAN (C,P, threshold)
2:     Initialize voxel space from C
3:     for all unseen voxels, u, in the voxel space do
4:         if u has a seen empty neighbor then
5:             add u to U                    ▷ the set of boundary
                                               unseen voxels
6:         end if
7:     end for
8:     loop
9:         for all potential views p in P do
10:            Count members of U that are visible
11:        end for
12:        if count(p) < threshold then
13:            break
14:        end if
15:        Acquire scan at p with largest count
16:        Register and merge new scan into C
17:        Update voxel space with C
18:        Recompute U from the new voxel space
19:    end loop
20:    return C                              ▷ the updated model
21: end procedure
```

4. SIMULATING THE MODELING PROCESS

A typical scan can take anywhere from 45 min to an hour. Depending on the size and complexity of the site, a real-world experiment could take many hours or days. To fully test our algorithms, a faster system is needed. We therefore developed the fast scan simulator described in this section.

4.1. Simulator Implementation

We must first obtain a 3-D model of the desired environment in which to simulate scanning. The availability of 3-D models has become much greater in recent years. There are a number of rather large repositories of such models, the most extensive of which is the Google 3D Warehouse (2008). These repositories contain numerous, and sometimes crude, models of many different buildings and historical sites of interest. Alternatively, we may want to construct models of our own to put into our scan simulator. There exist a number of professional-grade tools for constructing 3-D models, including packages such as Solidworks, Maya, AutoCad, and 3DMax. There also exist simpler and less expensive 3-D model-creation packages such as Google Sketchup (2008). For our simulator, we made use of the Sketchup package to construct our simulated environments; however, any of the above packages could have been used.

Once we have that model, we need to mimic the behavior and the limitations of our scanning system as closely as possible. We first choose a point within the synthetic model that will act as our scanning location. We must then cast rays from that point into the scene and compute the point of intersection. Each ray that we project into the scene represents a laser reading taken by a real scanner. As we cast out rays in the same pattern that our scanning system shoots the laser, we should obtain a point cloud generated from the synthetic object that is very similar to what the real scanner would have returned had the simulated environment been real. Casting this many rays into the scene is an expensive task and requires great care when dealing with boundary conditions. However, it is essentially the same task that rendering software performs routinely. We render the model with a viewpoint, centered at the desired scan location, and read off the depths from the Z-buffer. If we then transform these data from window coordinates back into object coordinates, we get the same information and can generate the same kind of point cloud. By implementing this process with OpenGL, we can automatically have GL-enabled graphics cards compute the point cloud for us with fast specialized hardware.

Ultimately, we wish to render the entire 360-deg-around and 180-deg-up-and-down field of view surrounding this viewpoint. However, we cannot easily instruct GL to render the full field of view. Instead, we must chop up the view into manageable windows. To do this, we start with not just a viewpoint but also an initial orientation. The initial orientation does not matter if we are planning to render the full spherical view; however, it does matter if we are simulating a scanner with a limited field of view. Starting at this initial orientation, we render a 60×60 deg window of the scene. We then extract the 3-D data for the simulated scan and then rotate the orientation to the next window over. We repeat this until our windows have covered the entire spherical field of view of the simulated scanner.

From each rendered window, we need to extract 3-D point-cloud data. We use the Z-buffer of the rendering to extract the depth values at each pixel in the rendered image. The depths that we read from the Z-buffer are the distances to the closest object along that particular ray. As a result, we extract only the range to those objects that are actually visible from that viewpoint. This satisfies the visibility constraint of a range scanner, and occlusions are properly computed. Each pixel will represent a laser beam emanating from the range scanner. The x and y coordinates, combined with the depth reading from the Z-buffer, can then be transformed back into world coordinates. These world coordinates give us a simulated 3-D data point. This procedure is repeated for each window rendered from a particular viewpoint. Ultimately, all of the values are transformed back into the world coordinate system and therefore give a complete simulated scan from the given viewpoint. Minimum and maximum range constraints can be enforced by properly setting the near and far clipping planes.

4.2. Guggenheim Museum Bilbao

In this section, we give an example of the simulator in use on a model of a real-world structure. The Guggenheim Museum Bilbao (see the top of Figure 6) by Frank Gehry in Bilbao, Spain, is a complex structure with many interesting 3-D occlusions. We retrieved a model of this structure from Google 3D Warehouse (2008) and ran it through our simulated

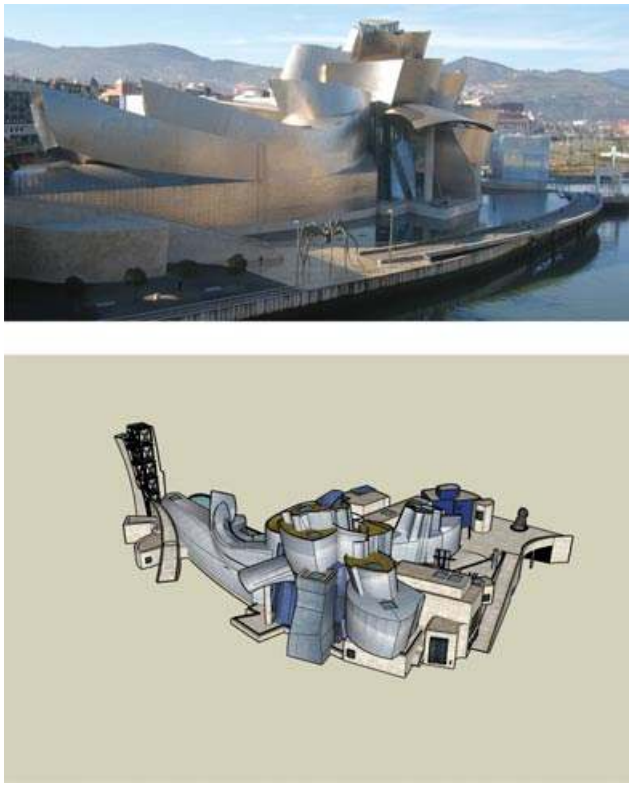

Figure 6. Top: The Guggenheim Museum Bilbao by Frank Gehry in Bilbao, Spain (picture was taken by Michael Reeve). Bottom: A model of the Guggenheim Museum Bilbao retrieved from Google 3D Warehouse.

view planning system. The model we retrieved can be seen at the bottom of Figure 6.

We first took a floor plan of the building and used it for the Phase I 2-D planning system. This resulted in eight initial viewing locations (see Figure 7) chosen by the planner. The point cloud obtained after acquiring these eight simulated scans can be seen at the top of Figure 8. At this stage, the model looks relatively sparse because much of the upper structure has not been fully acquired due to the large number of occlusions in the upper areas of the building.

We then voxelized this model and ran our Phase II 3-D planning algorithm to compute NBVs sequentially. We ultimately computed 23 NBVs before the algorithm reached its terminating threshold. A view of the final model can be seen at the bottom of Figure 8. The final model is significantly more dense in some areas than was the model produced at the end of Phase I. Even though the initial modeling phase scanned those areas, they needed to be scanned again several times from slightly different angles in order to fully acquire additional portions of the curved roof surface. Even after the algorithm terminated, there were still missing portions of the sides of the roof that could not be acquired from the ground plane.

4.3. East Campus

In our final test of the simulator, we hand-modeled a building on the Columbia University campus, the East Campus Building. This is an interesting building from an occlusion standpoint because it is composed of two sections: a low tower in front and a high tower in the back. Between the two towers is a courtyard with a number of enclosed stairwells that jut into the courtyard. The exterior of the building can be seen at the left of Figure 9. Our Google Sketchup model of the building can be seen at the right of Figure 9.

We ran our 2-D planner on the floor plan of the courtyard. This resulted in five initial scanning locations (see Figure 10) chosen by the planner. The resulting model after acquiring those five scans can be

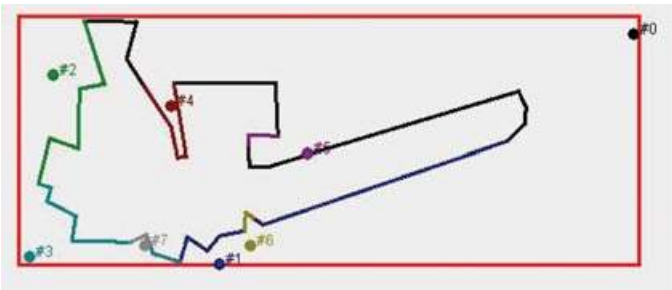

Figure 7. The floor plan of the Guggenheim Museum Bilbao with the eight scan locations chosen by the Phase I 2-D planning algorithm.

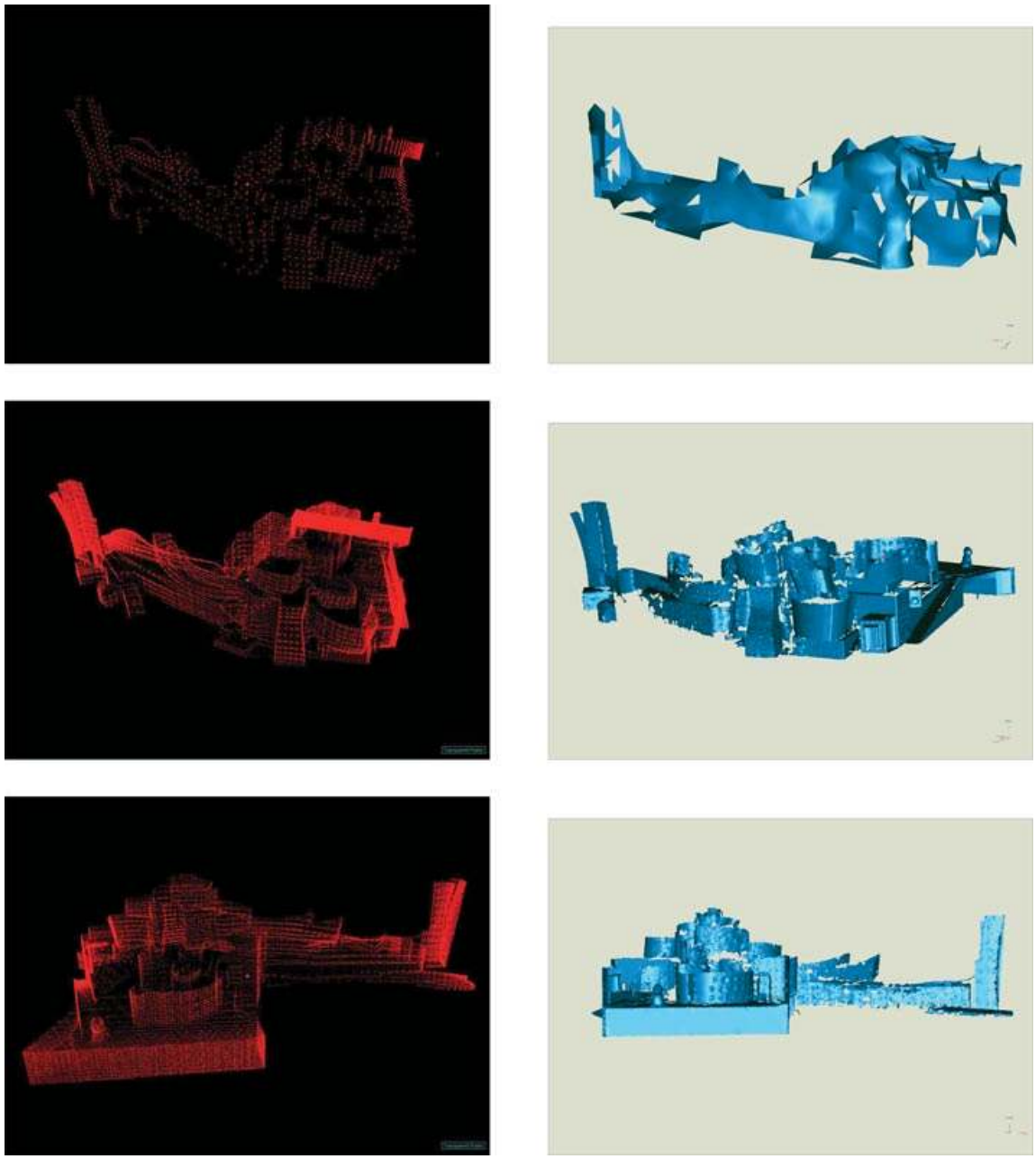

Figure 8. The top row of images shows the resulting model produced by the scan simulator after acquiring the eight scans computed by the Phase I 2-D planner as applied to the Google 3D Warehouse model of the Guggenheim Museum Bilbao. The bottom two rows of images show two different views of the final model produced by the scan simulator after acquiring the 23 scans computed by the Phase II 3-D planner as applied to the Google 3D Warehouse model of the Guggenheim Museum Bilbao. The original eight scans from the Phase I planner are also included. The first image in each row is the point cloud of the model; the second is the meshed version.

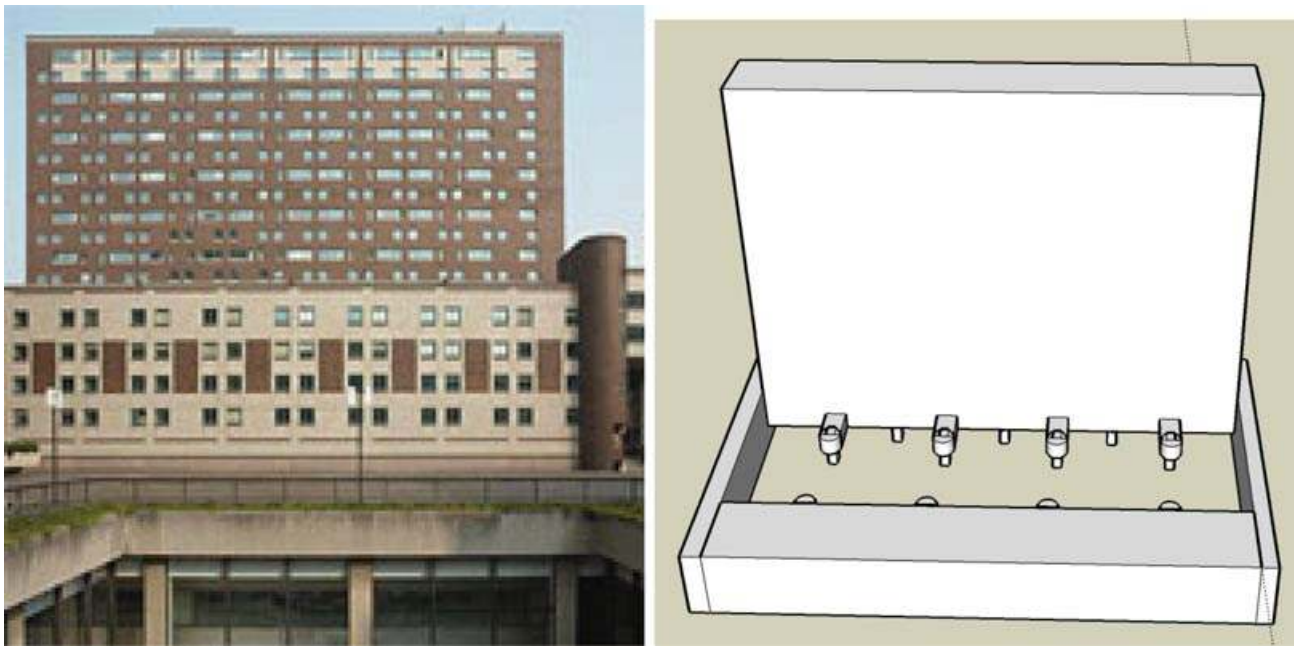

Figure 9. Left: The East Campus Building at Columbia University. Right: The synthetic model of East Campus.

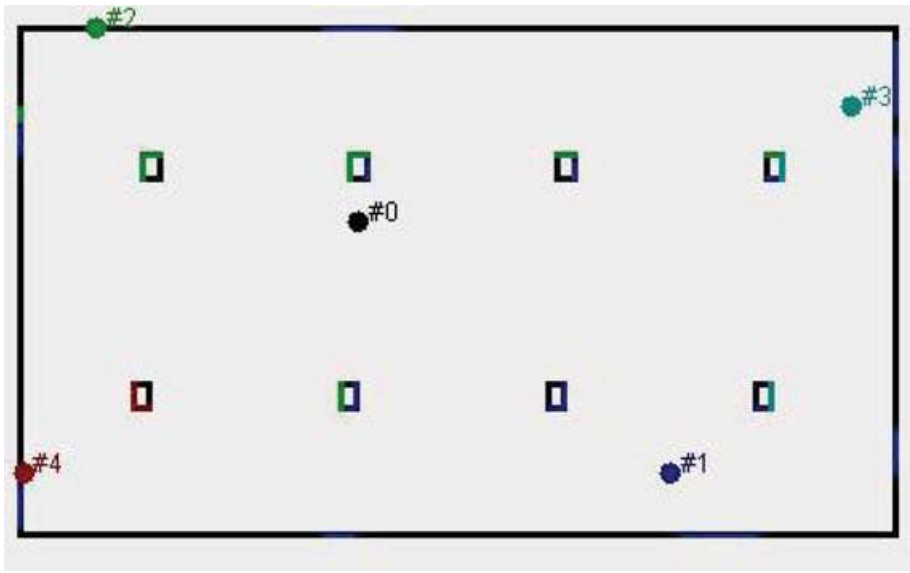

Figure 10. The floor plan of the East Campus courtyard along with the five scan locations chosen by the Phase I 2-D planning algorithm.

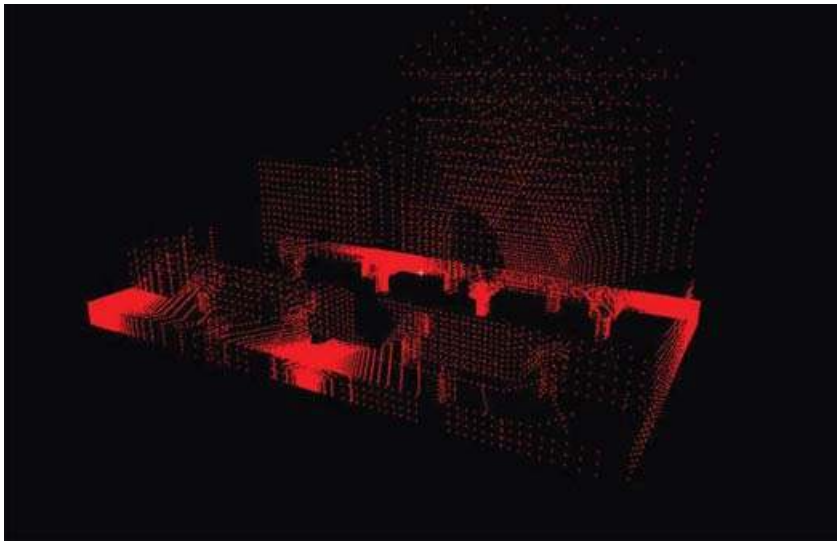

Figure 11. The resulting model from the scan simulator after acquiring the five scans computed by the Phase I 2-D planner for the synthetic model of East Campus.

seen in Figure 11. This point cloud was then passed into our 3-D planner, and the planner went through four additional iterations before reaching its threshold stopping condition. The final simulated point cloud after all nine scans from both Phase I and Phase II can be seen in Figure 12.

Because we were modeling a real building on campus, we had an opportunity to scan the actual building using our planning algorithm. The floor plan we used for the 2-D phase of the algorithm for the actual building was the same as the floor plan of our synthetic model. The Phase I planner therefore returned the same five preplanned viewpoints. These five scans were then acquired from the actual building, and the resulting model at the conclusion of Phase I is shown in Figure 13. This model was then

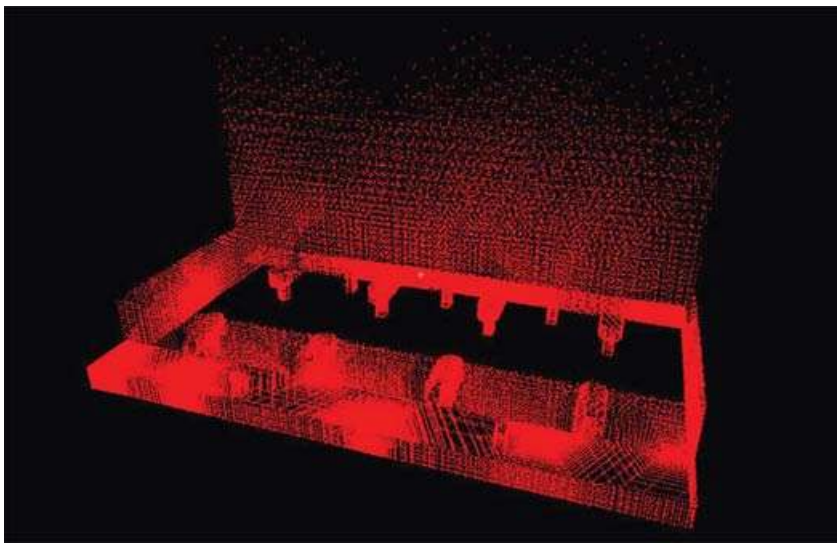

Figure 12. The final model of East Campus as taken by the scan simulator. This includes the four scans chosen by the Phase II planner along with the five scans chosen by the Phase I planner.

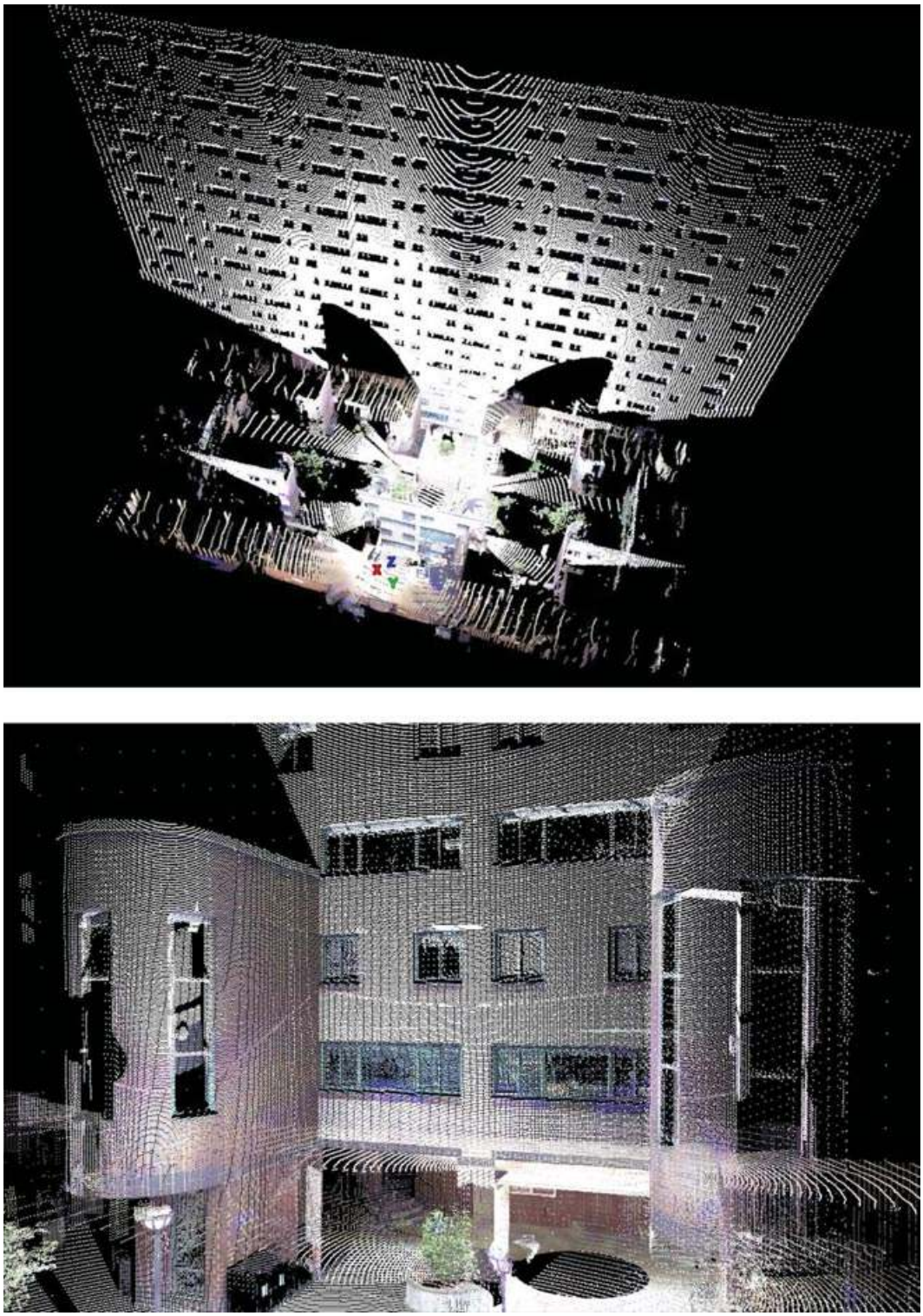

Figure 13. The point cloud of East Campus obtained after the Phase I scanning of the actual building. Top: An overview of the whole structure. Bottom: A close-up of a portion of the courtyard.

run through the Phase II 3-D planner and resulted in an NBV that was very similar in location to the first NBV chosen by the planner with the simulated data.

This points to another potential use for the simulator. If a rough 3-D model of the scene of interest can be found or quickly prototyped, running the Phase I 2-D planner along with several iterations of the Phase II 3-D planner on this synthetic 3-D model would result in an expanded set of precomputed preliminary views. These views, calculated before one gets on site, should cover more of the site when actually acquired than would be covered by using only those scans calculated from the Phase I planner. Further NBVs could then be computed from the real data to fill in fine details that were missed or that could not have been captured in the simple prototype model. The net result would be a larger set of precomputed scan locations, which would mean a decrease in the amount of computation time required on site to obtain the complete model.

5. EXPERIMENTAL RESULTS

In this section we present two complete experiments utilizing our two-stage planning pipeline. In the first experiment, we model the northern end of the Columbia University campus surrounding Uris hall (Blaer & Allen, 2006b). In the second, we construct a model of Fort Jay on Governors Island in the City of New York (Blaer & Allen, 2007).

5.1. Uris Hall

As our first test of the complete two-stage view planning algorithm, we chose to model the northern end of the Columbia University campus, centered around the building Uris Hall shown on the left of Figure 14. For Phase I of the algorithm, we set the threshold such that the algorithm terminated if additional scans added less than 2% of the total boundaries of the target region. Our algorithm (see Algorithm 1) typically returned between 8 and 10 scanning locations for our test area (see the right of Figure 14), giving us a coverage of 95% of the region's obstacle boundary.

Once the initial set of viewpoints was chosen, the scans needed to be acquired. In this example, we are testing only the view planning algorithms; therefore, all scans were acquired manually and registration was performed by placing fiducial marks in the scene at known locations. In this particular run of the algorithm, Phase I planning generated nine scanning locations. We took scans at each of the nine locations chosen by the planner and registered them. Figure 15 shows the model being constructed from the 2-D plan. The resulting model from this algorithm consists of registered sets of point clouds.

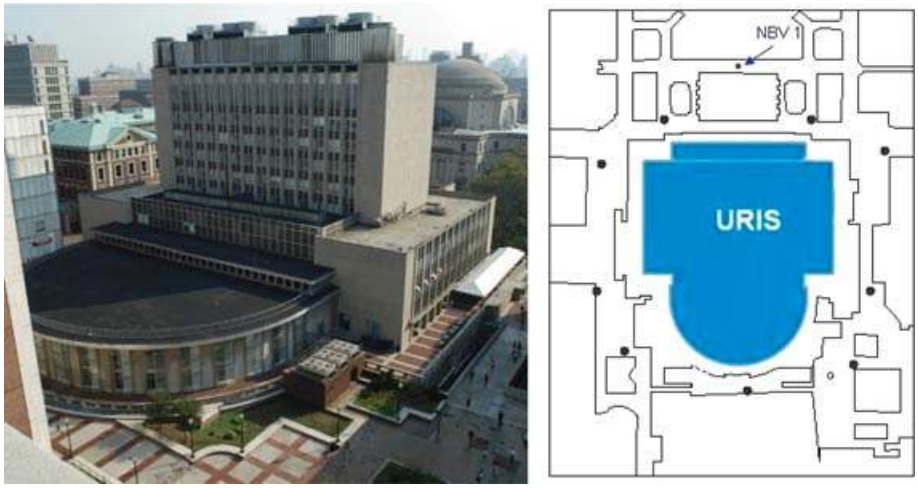

Figure 14. Left: A photograph of Uris Hall, the building at the center of our test region, taken from the roof of a neighboring building (picture courtesy of Alejandro Troccoli). Right: The 2-D map of the building footprints on the northern portion of the Columbia campus. Also shown are the nine scan locations (shown as black dots) determined by the Phase I view planner. The views from these locations cover 95% of the 2-D outline of Uris Hall. The rectangle is centered at the location of the first NBV computed by the Phase II planner.

Figure 15. The initial campus model being constructed sequentially from the 2-D plan. From top to bottom: The first scan, registered scans 1–4, and the complete initial model with all nine scans. The scans are texture mapped with images taken from the scanner's built-in camera. (These images are best viewed in color.)

Table I. Uris Hall model statistics.

At the end of	Seen-occupied	Total unseen	Boundary unseen	Visible to NBV
Phase I	76,941	114,372	21,765	1,025
NBV 1	78,212	111,216	18,540	972
NBV 2	79,005	109,571	15,327	322
NBV 3	79,243	109,011	15,156	122

A summary of the number of labeled voxels after each iteration of the 3-D planning algorithm. The last column indicates the number of *boundary unseen* voxels that are actually visible to the NBV.

The initial model was then converted to voxels and inserted into the voxel space. Table I summarizes the number of labeled voxels at each iteration of the algorithm. Voxels that contained data points from the initial model were labeled as *seen-occupied*. We then carved the *seen-empty* voxels by ray tracing from the positions of the scanning system. Our grid had a total of approximately 15.5 million voxels. After the initial model, 76,941 voxels were seen occupied and 114,372 voxels were labeled as *unseen*, with the rest labeled *seen-empty*. Next, we computed the set of *boundary unseen* voxels. It turned out that only a small number of the *unseen* voxels were in this set (21,756). This operation pruned the number of voxels whose visibility we needed to consider by an order of magnitude. At this point, we computed the potential viewing location that could see the largest number of those *boundary unseen* voxels. This location is shown as the rectangle on the right of Figure 14.

This view looked directly at the facade of the central building of our target region. The 2-D planner had computed views that saw this facade from the corners. The building, however, has two parts to it, a wide base and a thin tower. The 2-D planner took into account only the footprint of the building, and as a result the scans from the corners were unable to image much of the tower section of the building. The 3-D planner's first choice for the NBV was a head-on scan of the building that filled in the large *unseen* region of the tower's facade (see Figure 16).

For this experiment, we chose a threshold of 200 boundary unknown voxels as the cutoff for our algorithm. When it was estimated that the NBV would net fewer than this threshold number of voxels, we stopped scanning. In our test, we reached this threshold after the third NBV. Although there was still a sizable number of *boundary unseen* voxels (15,156) in the

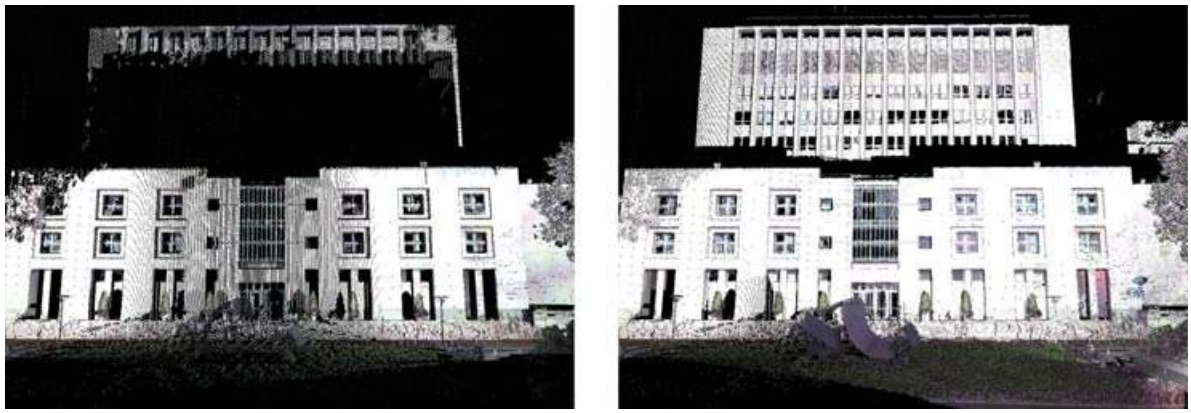

Figure 16. Left: The facade of Uris Hall, scanned from the views generated by the Phase I planner. Phase I generated views only at the corners of the building, causing much of the tower to be occluded by the base. Right: The facade of the same building after acquiring the first NBV.

scene, we were restricted by where we could place the robot. A view of the final model of Uris Hall can be seen in Figure 17. Portions of the roof are missing in the resulting model. This, however, is not a failing of the view planning algorithm. The roof is not visible from the ground level at all because ground-based scanners are not capable of imaging the roof.

5.2. Fort Jay

Fort Jay (see Figure 18, left) is a large fort located on Governors Island in the City of New York. With the kind permission of the National Park Service, we used this as our outdoor test bed for the full VuePlan system.

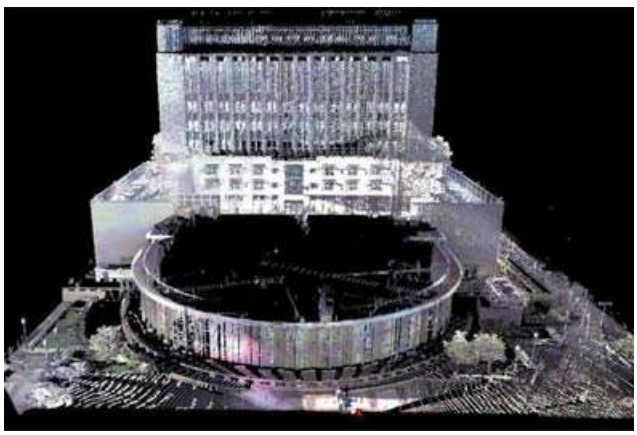

Figure 17. A view of the final model of Uris Hall after all 12 scans from Phases I and II were acquired.

5.2.1. Phase I: Building an Initial Model of Fort Jay

Initially we were given a floor plan of the fort's building by the Park Service. For the purposes of modeling, Fort Jay can be divided into three distinct sections: the inner courtyard, the outer courtyard, and the moat. We split the fort into these three sections because the transitions between them were not easily traversable by the robot. For the inner courtyard modeling, the mobile robot chose the scanning locations, planned the tour of those positions, and traversed the path, allowing us to acquire the inner model with the mobile robot.

To compute the path, the robot needs to be able to travel from any viewpoint to any other viewpoint. Such paths can be computed by using the underlying navigation system of our mobile robot, which utilizes a Voronoi diagram–based path planning module described in our earlier work (Blaer & Allen, 2003). The 2-D region in which the robot moves will contain buildings and other types of barriers, each of which can be represented by a polygonal obstacle. To find the generalized Voronoi diagram for this collection of polygons, we use an approximation based on the simpler problem of computing the Voronoi diagram for a set of discrete points. First, we approximate the boundaries of the polygonal obstacles with the large number of points that result from subdividing each side of the original polygon into small segments. Second, we compute the Voronoi diagram for this collection of approximating points using Fortune's (1987) sweepline algorithm. Once this Voronoi diagram is constructed, we then eliminate those Voronoi edges that have one or both end points lying inside any of

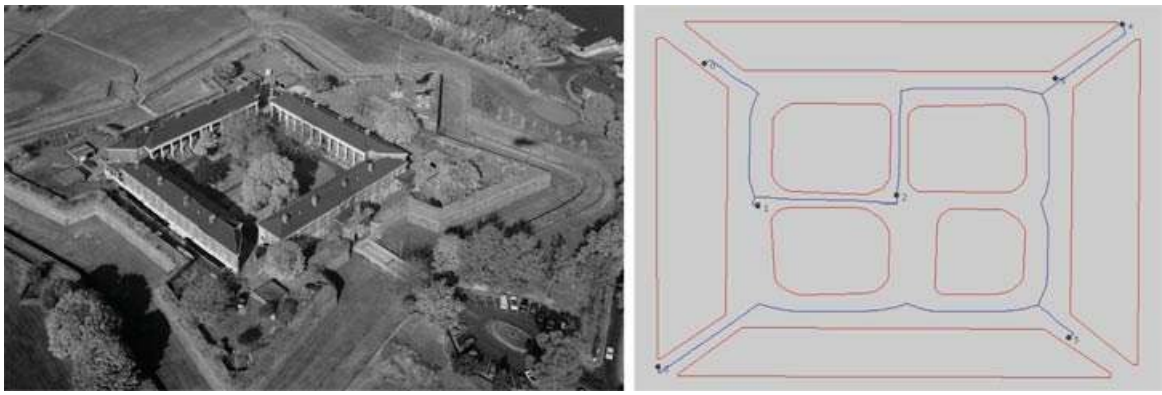

Figure 18. Left: Fort Jay, located on Governors Island in the City of New York. Right: An efficient tour of the seven viewpoints planned by the Phase I algorithm.

the obstacles. The remaining Voronoi edges form a good approximation of the generalized Voronoi diagram for the original obstacles in the map. Paths are then found by first moving the robot on to a node of the Voronoi graph and then computing the shortest path with Dijkstra's algorithm to the node closest to the destination location.

The problem of computing the robot's tour of viewpoints can be formulated as a variant of the traveling salesperson problem (TSP). The set of viewpoints calculated in the Phase I algorithm is the graph on which we are performing our TSP computation. The lengths of the paths generated by the path planner between two viewpoints are then used as the edge cost for our TSP implementation. Although this problem is nondeterministic polynomial-time (NP) hard, good approximation algorithms exist to compute a near optimal tour of the observation locations. With our tour of the initial viewpoints computed, we can then allow the underlying navigation system software to bring the robot to each scanning position and acquire the scans. Wang, Krishnamurti, and Gupta (2007) have also addressed this problem by merging view and travel costs into a single metric for selecting views.

Figure 18, right, shows the tour of the seven viewpoints in the inner courtyard as computed by the method described above. A video of the robot following this path can be seen at Modeling Fort Jay at Governors Island (2008). Because of severely sloped terrain, the robot could not operate in the outer courtyard or the moat. As a result, these two outer regions were acquired by using our planning algorithm to make the decisions and then placing the scanner manually at the scanning locations.

As with the previous experiment, we set our termination threshold to 2% of the total boundaries of the target region. Our algorithm produced 7 locations in the inner courtyard, 10 locations in the outer courtyard, and 9 locations in the moat, giving us a coverage of 98% of the region's obstacle boundary. On the top of Figure 19, one can see the planned locations for the outer courtyard. We acquired and registered each of these planned views with our scanner and built an initial model. As we did in the previous experiment, registration was performed by placing fiducial marks at known locations throughout the scene. Figure 19, bottom, shows the entire initial model acquired by Phase I of the algorithm.

5.2.2. Phase II: Refinement of the Fort Jay Model

Next, we wished to refine our model using our 3-D planning algorithm (see Algorithm 2). First, the initial model was inserted into the voxel space (see Figure 20). Voxels that contained data points from the initial model were labeled as *seen-occupied*. We then computed the *seen-empty* voxels by ray tracing from each of the positions from which we took a scan. On the ground level, the site was approximately 300 $\times$ 300 m^2. No elevation measured was above 30 m. Using a resolution of 1 m^3 for our voxels, we had a grid of 2.7 million voxels. Of those voxels, most of them (2,359,321) were marked as *seen-empty*. Of the remaining voxels, 68,168 were marked as *seen-occupied*, leaving 272,511 *unseen* voxels. Of those *unseen* voxels, only 25,071 were actually *boundary unseen*.

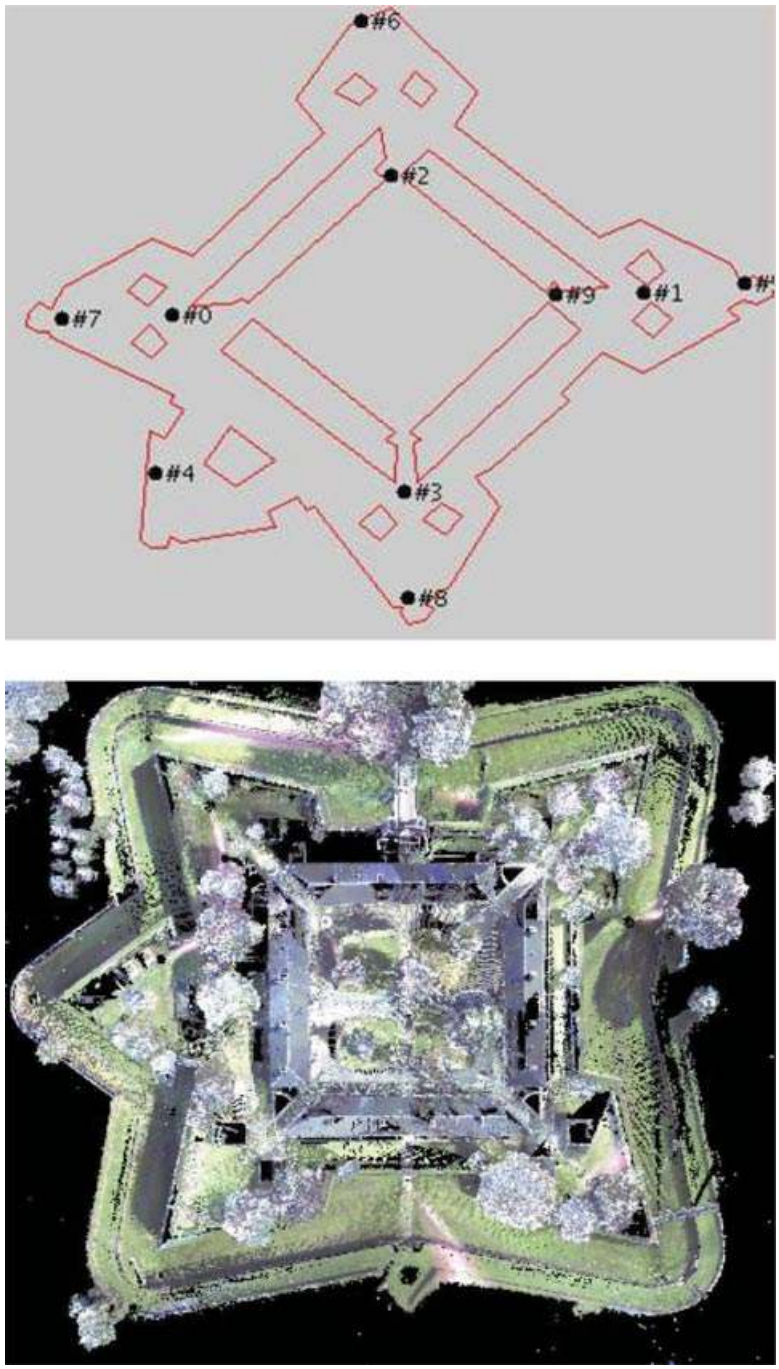

Figure 19. Top: The viewing locations chosen by the Phase I view planner for the outer courtyard of Fort Jay. Bottom: The entire model of Fort Jay (seen from above) generated by the initial planning stage for the inner and outer courtyards and the moat.

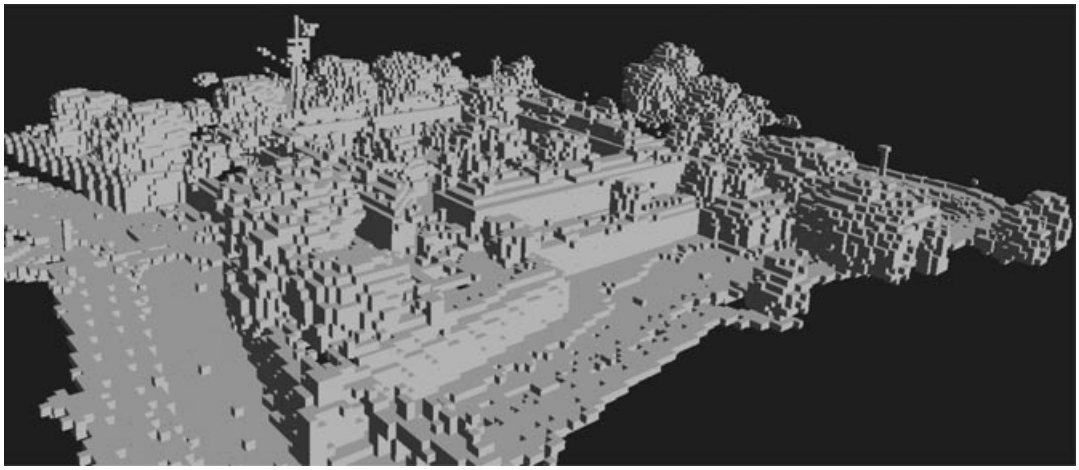

Figure 20. A representation of the *seen-occupied* cells of Fort Jay's voxel space. It was constructed at the end of Phase I from the point cloud acquired in Section 5.2.1.

Once we had the initial voxel space, we computed the potential viewing location that could see the largest number of those *boundary unseen* voxels. It turned out that this view was attempting to resolve part of a very large occlusion caused by the omission of a building on the 2-D map of Fort Jay. The footprint map that we were given did not have this building indicated, and we decided that it would be a more realistic experiment not to modify the 2-D map we were given. These kinds of errors are not uncommon, and a planning algorithm should be able to deal with them. One can see the resulting hole in the model on the left of Figure 21. The first NBV resolved only a portion of the occlusion. Our algorithm says that if a voxel is *unseen*, we have to assume that it could be occupied and that a potential viewpoint cannot see beyond such a voxel. Therefore, we can aim only at the frontier between the seen and the unseen portions of the space from a vantage point located in a *seen-empty* region. In this case, aiming at the frontier did not resolve all of the occlusion. The next NBV that was computed, however, did finally resolve the occlusion, because enough of the *unseen* region had been revealed as empty so as to safely place the scanner in a position

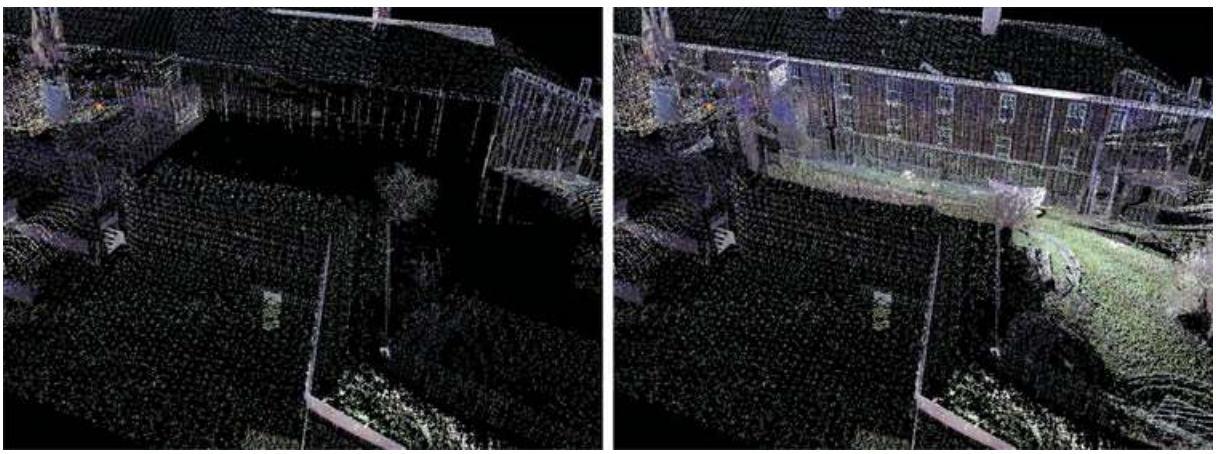

Figure 21. Left: A segment of the model of Fort Jay created during Phase I of the algorithm. The hole in the left-hand figure is caused by an unpredicted occlusion of one of the initial scanning locations. Right: The same section of the model after the first two NBVs from Phase II were computed and acquired.

Table II. Fort Jay model statistics: A summary of the number of labeled voxels after each iteration of the 3-D planning algorithm.

At the end of	Seen-occupied	Boundary unseen	Visible to NBV
Phase I	68,168	25,071	1,532
NBV 1	70,512	21,321	1,071
NBV 2	71,102	18,357	953
NBV 3	72,243	17,156	948
NBV 4	73,547	15,823	812
NBV 5	74,451	14,735	761
NBV 6	75,269	13,981	421
NBV 7	75,821	13,519	255
NBV 8	76,138	13,229	98

The second to last column indicates the number of *boundary unseen* voxels that are actually visible to the NBV.

that could see the entire occlusion. This portion of Fort Jay can be seen on the right-hand side of Figure 21 after the two NBVs were acquired.

For this experiment, we chose a threshold of 100 *boundary unseen* voxels visible to the NBV as the cutoff for our algorithm. Ultimately we needed to compute four NBVs in the outer courtyard, one NBV in the inner courtyard, and two additional NBVs in the moat. Table II summarizes the number of labeled voxels at each iteration of the algorithm. On average, each iteration takes approximately 10 min to complete without optimization. The algorithm is inherently parallel, and we are investigating using current multicore technology to implement speedups of the algorithm. This is an acceptable computation time because it is very short in comparison to the combined positioning and scan time for the mobile robot, which can take up to an hour per scan.

A view of the final model of Fort Jay can be seen on the bottom of Figure 22, and detailed meshes of two sections of the model can be seen in Figure 23. A video of the entire model can be seen at Modeling Fort Jay at Governors Island (2008).

6. CONCLUSION

There are many different approaches to the exploration and mapping of large-scale sites by a mobile robot. Our main interest has been the construction of dense and detailed models of such sites, and fast low-resolution scans taken by a moving vehicle do not satisfy our requirements. With such a moving vehicle the scanning locations are restricted to streets, occlusions are not explicitly dealt with (leaving a possibly incomplete model), and the resulting scans may not be high enough resolution for our purposes. Our scanning equipment maximizes detail and therefore requires very long scan times. As a result, using an essentially random location for each fixed-position scan would require too many such scans for complete coverage of a large site and would become prohibitively time-consuming. The view planning algorithms we have developed eliminate unnecessary scans and can significantly reduce the total time needed for detailed model building.

Much of the existing view planning literature is aimed at constructing models of single small-scale objects rather than large complex sites. View planning techniques must be extended to cover significantly larger and more complicated environments. In this paper we have presented a systematic method for automated view planning in the construction of highly detailed 3-D models of large indoor and outdoor environments. We have also integrated this algorithm into our mobile robot system, allowing us to acquire those views automatically. Models have been acquired using this method for buildings on the Campus of Columbia University and Governors Island and for cathedrals in the Bourbonnais region of France (Romanesque Churches of the Bourbonnais, 2007). In all these cases our view planning made acquiring these complex models more efficient and complete.

The procedure started from a 2-D ground map of the environment, which defined the open space available to the robot. The method then progressed through two distinct stages. In Phase I, a preliminary 3-D model was constructed by having the robot compute and acquire a set of initial scans using our 2-D view planning algorithm. In Phase II, this initial model was then refined by a 3-D view planning phase that made use of a voxel representation of the environment and that successively planned NBVs in order for the robot to acquire a complete model.

Our system for view planning and site modeling could benefit from improvements that would decrease the computation time required on site. In our current system, the Phase II calculations of voxel visibility are still computationally expensive. To mitigate this, we try to fill in as much of the model as we can before embarking on Phase II and we then limit the number of *unseen* voxels that actually need to be considered during the second phase. One should note

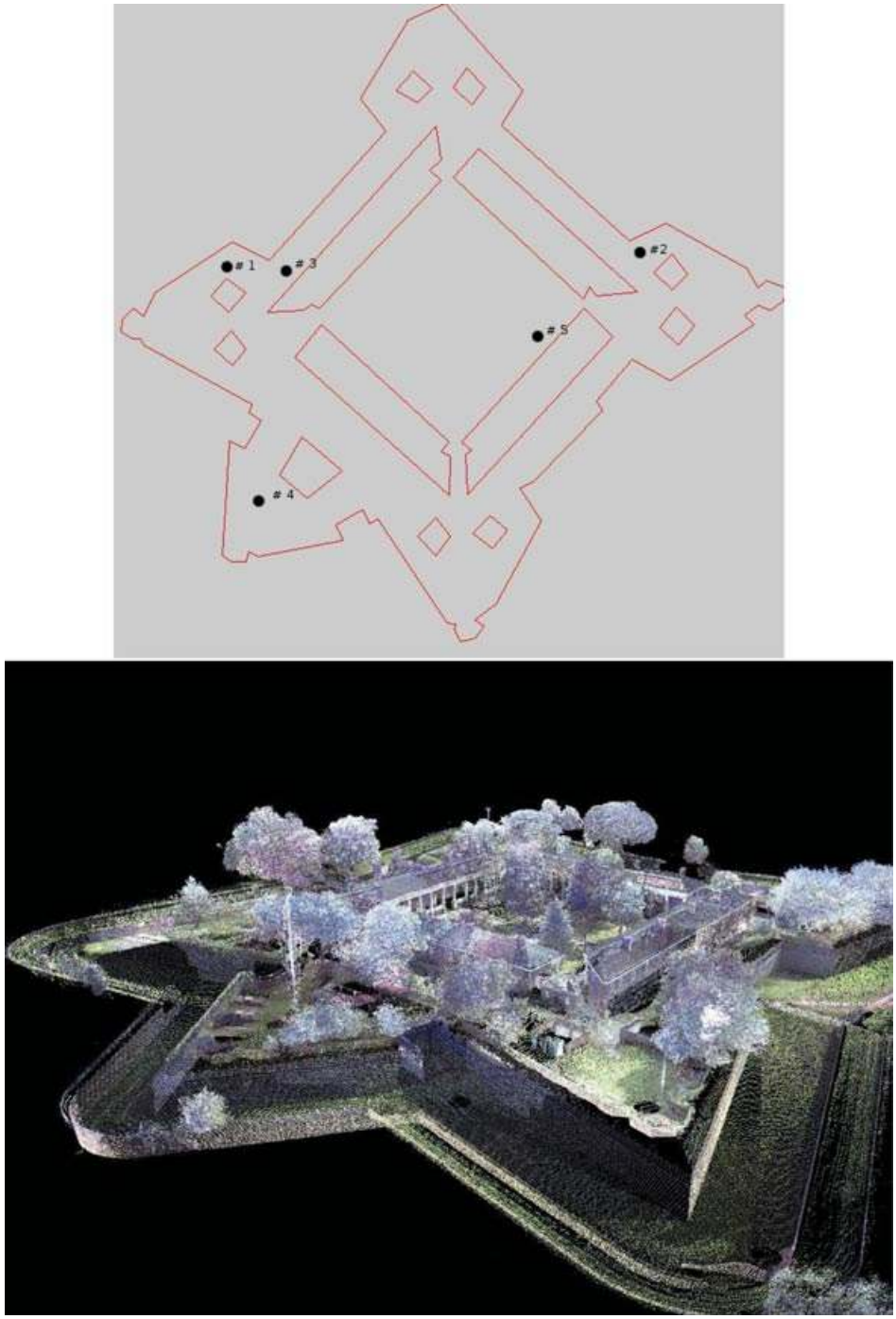

Figure 22. Top: The viewing locations chosen by the Phase II view planner for the inner and outer courtyards of Fort Jay (two more NBVs were chosen for the moat; these are not shown). Bottom: A view of the final model of Fort Jay after all 33 scans from Phases I and II were acquired.

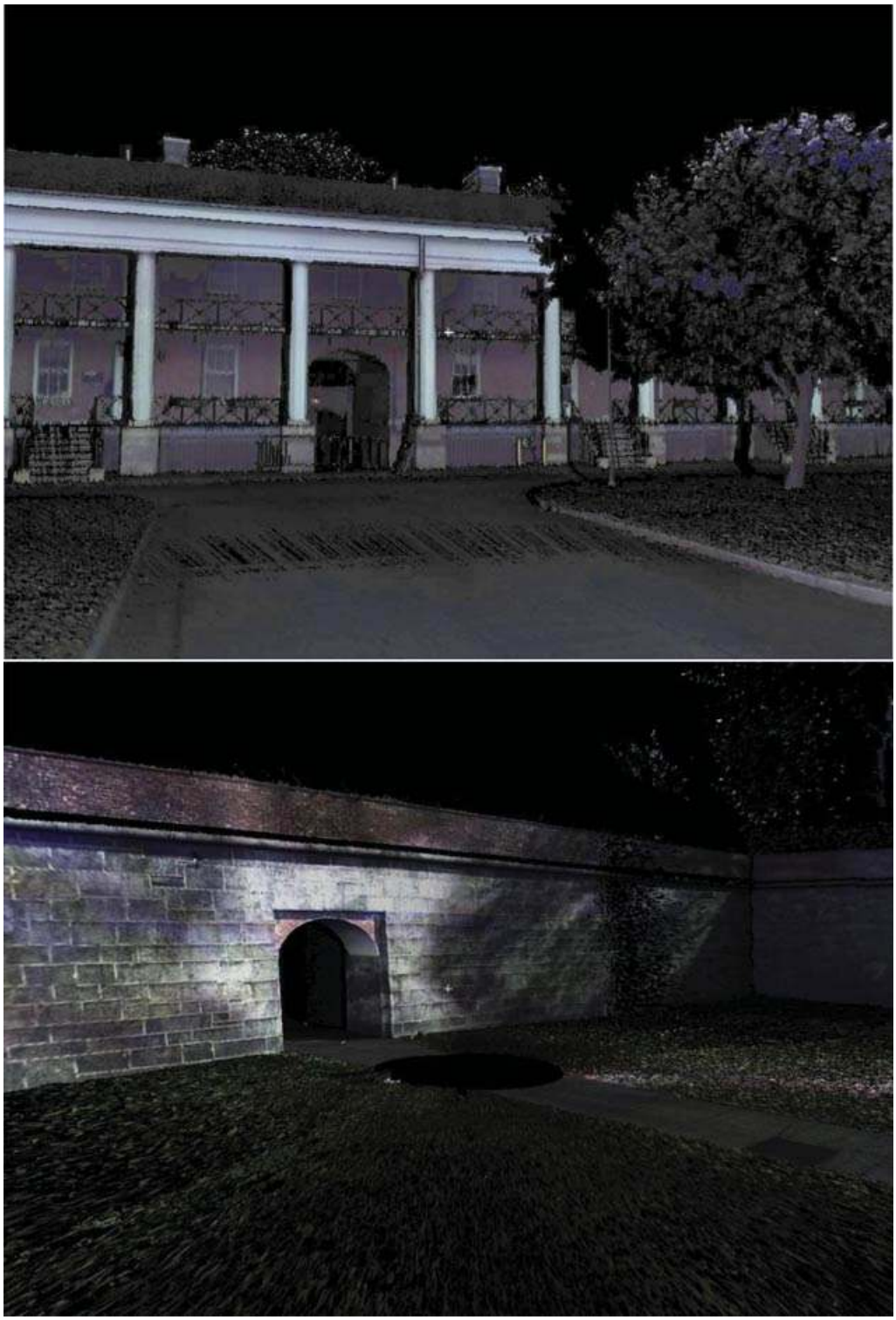

Figure 23. Detailed views of portions of the final model of Fort Jay.

that the evaluation of a voxel's visibility by potential viewpoints is inherently independent of the evaluation for another voxel. Therefore, these calculations could be done in parallel and would benefit from multicore processors to speed up computation in the field. Further calculational time saving could be accomplished in Phase II by forming regions of *boundary unseen* voxels and regions of potential viewpoints so that one would need only to use a single representative voxel and a single representative viewpoint for each region. One would thereby dramatically reduce the number of voxels and viewpoints that had to be considered.

Determining ground truth for our experiments is another difficult problem. There are no existing accurate 3-D models to compare against. Even renderings based on architectural plans are not that useful because most structures are not built exactly to specifications. One option is to add noise to the scan simulator and compare the resulting simulated scan to the 3-D model on which we are running the simulation; however, this approach could never reproduce all of the environmental conditions that could affect our model construction in the field.

Another avenue of exploration involves using the scan simulator in the actual planning process. Simple 3-D models of important sites are becoming more prevalent, and a number of simple, inexpensive tools for prototyping 3-D models are becoming available. If a rudimentary 3-D model of a scene can be found or constructed prior to the scanning process, we could utilize this model to run both the Phase I planner and several iterations of the Phase II planner before we are on site to perform the scanning. This would give a larger set of precomputed scan locations, which would in turn decrease the amount of computation time required in the field to form a complete model.

A particular problem for our mobile robot is its inability to view roofs and other structures from a ground-based viewpoint. To alleviate this problem, our future work will consider the integration of additional sensing modalities, such as aerial LIDAR and imagery, into our system. New kinds of data not obtainable by a ground-based scanner not only would allow us to fill in gaps in our models but would also provide extra information to aid in the planning process itself. Our algorithm could also be generalized to accommodate viewpoints anywhere in the voxel space. We could then use viewpoints above the ground plane, with sensors mounted on autonomous aerial vehicles. However, for such an expanded system, we would have to address the calculational expense associated with the increased voxel search space.

A unique problem that affects and complicates most stages of any outdoor 3-D scanning process is the presence of foliage. Our method is not immune to these complications, and both Phase I and Phase II of the planning algorithm are affected. Most 2-D maps do not include these kinds of occlusions, so the Phase I algorithm cannot take into account the presence of foliage. The Phase II algorithm is also complicated by foliage because the voxel grid is rather coarse and the sparse data coming from the scanning of foliage can fill too much of the space with *seen-occupied* voxels. This overfilled region of space can block legitimate potential viewpoints from being considered. Our future research will have to address this important issue.

ACKNOWLEDGMENTS

This work was funded by National Science Foundation grants IIS-0121239 and ANI-00-99184 and a grant from the Andrew Mellon Foundation. Thanks to the National Park Service for granting us access to Governors Island and to the Governors Island staff Linda Neal, Ed Lorenzini, Mike Shaver, Ilyse Goldman, and Dena Saslaw for their invaluable assistance on the island. Thanks also to the scanning team of Hrvoje Benko, Matei Ciocarlie, Atanas Georgiev, Corey Goldfeder, Chase Hensel, Tie Hu, and Claire Lackner.

REFERENCES

Akbarzadeh, A., Frahm, J.-M., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Merrell, P., Phelps, M., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewenius, H., Yang, R., Welch, G., Towles, H., Nister, D., & Pollefeys, M. (2006, June). Towards urban 3D reconstruction from video. In 3DPVT, Chapel Hill, NC.

Allen, P. K., Stamos, I., Gueorguiev, A., Gold, E., & Blaer, P. (2001, May). AVENUE: Automated site modeling in urban environments. In 3DIM, Quebec City, Canada (pp. 357–364).

Banta, J. E., Zhien, Y., Wang, X. Z., Zhang, G., Smith, M. T., & Abidi, M. A. (1995). A "best-next-view" algorithm for three-dimensional scene reconstruction using range images. Proceedings of SPIE, 2588, 418–429.

Besl, P. J., & McKay, N. D. (1992). A method for registration of 3-D shapes. Transactions on Pattern Analysis and Machine Intelligence, 14(2), 239–256.

Blaer, P., & Allen, P. K. (2003, September). Topbot: Automated network topology detection with a mobile robot. In IEEE ICRA, Taipei, Taiwan (pp. 1582–1587).

Blaer, P. S., & Allen, P. K. (2006a, June). Two stage view planning for large-scale site modeling. In 3DPVT, Chapel Hill, NC.

Blaer, P. S., & Allen, P. K. (2006b, May). View planning for automated site modeling. In IEEE ICRA, Orlando, FL.

Blaer, P. S., & Allen, P. K. (2007, October). Data acquisition and view planning for 3-D modeling tasks. In IEEE/RSJ IROS, San Diego, CA.

Chen, C. S., Hung, Y. P., & Cheung, J. B. (1999). RANSAC-based DARCES: A new approach to fast automatic registration of partially overlapping range images. Transactions on Pattern Analysis and Machine Intelligence, 21(11), 1229–1234.

Connolly, C. I. (1985, March). The determination of next best views. In IEEE ICRA, St. Louis, MO (pp. 432–435).

Cowan, C. K., & Kovesi, P. D. (1988). Automatic sensor placement from vision task requirements. Transactions on Pattern Analysis and Machine Intelligence, 10(3), 407–416.

Curless, B., & Levoy, M. (1996, August). A volumetric method for building complex models from range images. In SIGGRAPH, New Orleans, LA (pp. 303–312).

Danner, T., & Kavraki, L. E. (2000, April). Randomized planning for short inspection paths. In IEEE ICRA, San Francisco, CA (pp. 971–978).

Dellaert, F. (2005). 4D cities spatio–temporal reconstruction from images. http://www.cc.gatech.edu/4d-cities/. Accessed June 27, 2009.

Fortune, S. J. (1987). A sweepline algorithm for Voronoi diagrams. Algorithmica, 2(1–4), 153–174.

Frueh, C., & Zakhor, A. (2003, June). Constructing 3D city models by merging ground-based and airborne views. In IEEE CVPR, Madison, WI (pp. 562–569).

Genermont, M., & Pradel, P. (1938). Les Eglises de France. Allier.

Georgiev, A., & Allen, P. K. (2004). Localization methods for a mobile robot in urban environments. IEEE Transactions on Robotics, 20(5), 851–864.

González-Banos, H., & Latombe, J. C. (2001, June). A randomized art-gallery algorithm for sensor placement. In Proceedings of the 17th Annual Symposium on Computational Geometry, Medford, MA (pp. 232–240).

González-Banos, H., Mao, E., Latombe, J. C., Murali, T. M., & Efrat, A. (1999, October). Planning robot motion strategies for efficient model construction. In International Symposium of Robotics Research, Snowbird, UT (pp. 345–352).

Goodman, J. E., & O'Rourke, J. (1997). Handbook of discrete and computational geometry. Boca Raton, FL: CRC Press.

Google Sketchup (2008). http://sketchup.google.com/. Accessed June 27, 2009.

Google 3D Warehouse (2008). http://sketchup.google.com/3dwh/. Accessed June 27, 2009.

Grabowski, R., Khosla, P., & Choset, H. (2003, October). Autonomous exploration via regions of interest. In IEEE IROS, Las Vegas, NV (pp. 27–31).

Ikeuchi, K., & Miyazaki, D. (2007). Digitally archiving cultural objects. New York: Springer.

Ikeuchi, K., Nakazawa, A., Hasegawa, K., & Ohishi, T. (2003, October). The Great Buddha project: Modeling cultural heritage for VR systems through observation. In ISMAR '03: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality, Tokyo, Japan (p. 7). IEEE Computer Society.

Jensen, B., Weingarten, J., Kolski, S., & Siegwart, R. (2005, September). Laser range imaging using mobile robots: From pose estimation to 3D-models. In Proceedings 1st Range Imaging Research Day, Zurich, Switzerland (pp. 129–144).

Lamon, P., Kolski, S., & Siegwart, R. (2006, September). The SmartTer—A vehicle for fully autonomous navigation and mapping in outdoor environments. In Proceedings of CLAWAR 2006, Brussels, Belgium.

Low, K.-L., & Lastra, A. (2006, June). Efficient constraint evaluation algorithms for hierarchical next-best-view planning. In 3DVPT, Chapel Hill, NC.

Massios, N. A., & Fisher, R. B. (1998, September). A best next view selection algorithm incorporating a quality criterion. In British Machine Vision Conference, Southampton, UK.

Maver, J., & Bajcsy, R. (1993). Occlusions as a guide for planning the next view. Transactions on Pattern Analysis and Machine Intelligence, 15(5), 417–433.

Modeling Fort Jay at Governors Island (2008). http://www.cs.columbia.edu/~pblaer/fort-jay/. Accessed June 27, 2009.

Nüchter, A., Surmann, H., & Hertzberg, J. (2003, June). Planning robot motion for 3D digitalization of indoor environments. In ICAR, Coimbra, Portugal (pp. 222–227).

Ohno, K., Tsubouchi, T., & Yuta, S. (2004, April). Outdoor map building based on odometry and RTK-GPS position fusion. In IEEE ICRA, New Orleans, LA (pp. 684–690).

Pfaff, P., Triebel, R., Stachniss, C., Lamon, P., Burgard, W., & Siegwart, R. (2007, April). Towards mapping of cities, In ICRA 2007, Rome, Italy.

Pito, R. (1999). A solution to the next best view problem for automated surface acquisition. Transactions on Pattern Analysis and Machine Intelligence, 21(10), 1016–1030.

Reed, M. K., & Allen, P. K. (2000). Constraint-based sensor planning for scene modeling. Transactions on Pattern Analysis and Machine Intelligence, 22(12), 1460–1467.

Romanesque Churches of the Bourbonnais (2007). http://www.learn.columbia.edu/bourbonnais/. Accessed June 27, 2009.

Scott, W. R., Roth, G., & Rivest, J.-F. (2001, June). View planning for multi-stage object reconstruction. In International Conference on Vision Interface, Ottawa, Canada.

Scott, W. R., Roth, G., & Rivest, J.-F. (2003). View planning for automated three-dimensional object reconstruction and inspection. ACM Computing Surveys, 35(1), 64–96.

Sequeira, V., & Gonçalves, J. (2002, June). 3D reality modelling: Photo-realistic 3D models of real world scenes. In 3DPVT, Padova, Italy.

Soucy, G., Callari, F. G., & Ferrie, F. P. (1998, October). Uniform and complete surface coverage with a robot-mounted laser rangefinder. In IEEE/RSJ IROS, Victoria, Canada (pp. 1682–1688).

Stamos, I., & Allen, P. K. (2002). Geometry and texture recovery of scenes of large scale. Computer Vision and Image Understanding, 88(2), 94–188.

Tarabanis, K. A., Allen, P. K., & Tsai, R. Y. (1995). A survey of sensor planning in computer vision. IEEE Transactions on Robotics and Automation, 11(1), 86–104.

Tarabanis, K. A., Tsai, R. Y., & Allen, P. K. (1995). The MVP sensor planning system for robotic vision tasks. IEEE Transactions on Robotics and Automation, 11(1), 72–85.

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

Teller, S. (1997, May). Automatic acquisition of hierarchical, textured 3D geometric models of urban environments: Project plan. In Proceedings of the Image Understanding Workshop, New Orleans, LA.

Thrun, S., Hähnel, D., Ferguson, D., Montemerlo, M., Triebel, R., Burgard, W., Baker, C., Omohundro, Z., Thayer, S., & Whittaker, W. (2003, September). A system for volumetric robotic mapping of abandoned mines. In IEEE ICRA, Taipei, Taiwan.

Turk, G., & Levoy, M. (1994, July). Zippered polygon meshes from range images. In Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, Orlando, FL (pp. 311–318).

Wang, P., Krishnamurti, R., & Gupta, K. (2007, April). Metric view planning problem with combined view and traveling cost. In IEEE ICRA, Rome, Italy (pp. 711–716).

Whaite, P., & Ferrie, F. P. (1997). Autonomous exploration: Driven by uncertainty. Transactions on Pattern Analysis and Machine Intelligence, 19(3), 417–433.

Xie, S., Calvert, T. W., & Bhatacharya, B. K. (1986, August). Planning views for the incremental construction of body models. In ICPR, Paris, France (pp. 154–157).