A Two-Stage Algorithm for Planning the Next View From Range Images

Miguel Angel García,
Department of Computer Science Engineering
Rovira i Virgili University
Ctra. Salou s/n. 43006 Tarragona, Spain
magarcia@etse.urv.es
Susana Velázquez and Angel Domingo Sappa
Institute of Cybernetics
Polytechnic University of Catalonia
Diagonal 647, planta 11, 08028 Barcelona, Spain
[velazquez|sappa]@ic.upc.es[1]

Abstract

A new technique is presented for determining the positions where a range sensor should be located to acquire the surfaces of a complex scene. The algorithm consists of two stages. The first stage applies a voting scheme that considers occlusion edges. Most of the surfaces of the scene are recovered through views computed in that way. Then, the second stage fills up remaining holes through a scheme based on visibility analysis. By leaving the more expensive visibility computations at the end of the exploration process, efficiency is increased.

1 Introduction

The automatic reconstruction of 3D objects (scenes in general) through range images is gaining popularity in computer vision and robotics owing to the variety of applications that can benefit from it, including world modeling [2], reverse engineering and object segmentation [3] or recognition. Two basic tasks must be addressed in order to solve that problem. First, an exploration process is necessary for determining the positions where the range sensor must be placed so that all the surfaces of the objects in the scene can be observed. Second, a multi-view integration process must combine the range images obtained from the different viewpoints to obtain a 3D model.

This paper focuses on the exploration task and presents a new technique for generating a small set of views necessary to reconstruct a complex scene. This issue is known in the literature as the *next-best-view* problem. The majority of previous proposals (e.g.,

This work has been partially supported by the Government of Spain under the CICYT project TAP96-0868. The third author has been supported by the Spanish Agency for International Cooperation and the National University of La Pampa (Argentina).

[1][8][10][11]) tackle this problem by modeling the unseen areas/volume of the scene being reconstructed and then computing the amount of unseen space that is visible from each allowed position of the sensor. The latter is supposed to move over a surface (cylinder or sphere) that is discretized for computational savings. A form of ray-tracing is usually necessary in some of the stages of the process.

The next-best-view position is chosen among the different candidate positions by an optimization process that maximizes the amount of unseen space observable from them. A different approach also based on optimization is proposed in [12]. There, the uncertainty of the superquadrics used to model the perceived objects is minimized, with the camera position being a function of the parameters of the superquadrics.

Unfortunately, the aforementioned stages are computationally expensive: from the modeling of unseen volumes in space, to the determination of visibility regions, or the final optimization process. In spite of this computational effort though, no proposals claim to reach an optimal solution capable of providing the minimum number of views, but a suboptimal solution giving a "small" number of them.

Taking that into account, we propose a different strategy based on two stages. The first stage avoids the costly computation of visibility regions and finds out the new point of view as a result of a voting process that only considers the location of occlusion edges. Occlusion edges have been shown to be useful for this problem ([7][9]). Experimental results show that this stage is sufficient to acquire the majority of surfaces present in the scene. Moreover, its outcome can be directly used to feed further algorithms, such as object recognizers or gross motion planners. The second stage applies visibility analysis to cover the holes left by the first stage due to self-occlusions among objects. These holes tend to be relatively small and can be filled up with a few further views.

In order to gain efficiency during the exploration process, a simple 3D model in which the different views are just appended is utilized, avoiding thus the costly integration of those views into a valid CAD model of the scene.

The proposed algorithm is described in Section 2. Section 3 presents experimental results based on a 3D graphical simulator. Conclusions and future lines are given in Section 4.

2 Next-Best-View Generation

Let us suppose a range sensor that is moving over a sphere (the *observation sphere*) centered at the target scene and at a minimum distance away from it such that all objects to be sensed are inside the sphere. For efficiency purposes, the observation sphere is discretized into a number of cells such that the sensor can only be positioned at the center of a cell. When a cell is visited, it is marked to prevent it from being chosen again. The sensor's position is identified by an orientation and an elevation angle relative to an arbitrary *observation frame* located at the center of the sphere. The sensor is always aiming at that center.

The first point of view is arbitrary (e.g., at the vertical of the observation sphere). From it, successive points of view are generated until all surfaces are recovered. Each iteration starts with a range image obtained from the current viewpoint and, based on it, the algorithm decides where to look next. This is done through the following steps.

2.1 Approximation of Range Images

For efficiency purposes, each range image is approximated by a triangular mesh upon which further processing is applied. This can be done in various ways, including the adaptive technique proposed in [4]. However, for conceptual simplicity, we select a set of pixels uniformly distributed in a rectangular grid. Then, each pixel is transformed to a 3D point referred to a Cartesian frame associated with the sensor. The corresponding transformation matrix is obtained from the sensor's calibration process. Finally the 3D points are triangulated by simply linking them along rows, columns and diagonals.

The resolution of the triangular mesh determines the minimum distance that can be distinguished during the exploration process. Holes or separations smaller than the mesh resolution will be discarded. The chosen resolution must be as low as possible in order to speed-up the process. We are currently working with meshes of 90 by 90 points.

From the previous mesh, triangles whose barycenters project onto a background pixel of the range image are discarded. Moreover, if μ is the average perimeter of all the triangles and σ the standard deviation, those triangles with larger inclination (e.g., more than 60 degrees) and a perimeter larger than $\mu + k\sigma$ (with $k = 1.2$ in our implementation) are also removed in considering that they are joining separate surfaces. This usually happens when an object partially occludes another.

After this removal step, a set of separate triangulated regions is obtained, each corresponding to different surfaces present in the scene. The edges of the triangles that delimit each region are labelled as *exterior edges*.

2.2 Determination of Occlusion Edges

Given a triangular mesh representing the current view, the objective now consists of merging it with previous views in order to determine what exterior edges become *occlusion edges*. In our context, an occlusion edge is the edge of a triangle that is susceptible to occlude surfaces of the scene. Occlusion edges are an indicative of scene regions that are pending to be explored, and will be used during the voting process (Section 2.4).

All triangular meshes obtained during the exploration process are stored in an *exploration model* as they are acquired. This integration does not require the generation of a valid CAD model or even a single triangular mesh though. This would be a time-consuming task and can be done off-line right after the exploration phase. Hence, the exploration model is just the concatenation of the different triangular meshes obtained for each view.

When a new mesh is added to the exploration model, all its points must be transformed from the sensor's local frame to the observation frame. This can be easily done as the position and orientation of the sensor are referred to that global frame as a result of the sensor's calibration. Then, the algorithm proceeds by determining what exterior edges from both the previous views and the current one become overlapped with any of the triangles of the exploration model. The exterior edges that are overlapped along their whole extent become interior edges, while those which are not fully overlapped become occlusion edges. According to that, occlusion edges are those edges that would delimit the boundaries of the regions in the exploration model if the latter was a real merging (zippering) of the different views into a single triangular mesh.

The problem of finding occlusion edges then becomes the detection of overlap between an edge and a triangle. To speed-up the process, each exterior edge is not tested against all the triangles of the exploration model, but against all the triangles that are in a neighborhood of the exterior edge. Specifically, we are discarding the triangles whose vertices are at a distance from the edge larger than 9 times the edge's length.

The detection of overlap is done by dividing each exterior edge into a number of *exterior points* (currently eight). Each point has associated the normal of the triangle to which its exterior edge belongs. An exterior edge is considered to be overlapped when all its exterior points are overlapped with any of the neighboring triangles.

In order to consider a triangle for overlap against an exterior point, that triangle must be close enough to the point and the angle between the normals of both the triangle and the exterior point must be lower than a certain threshold. In practice, we are considering a maximum distance equal to the average length of the triangle's edges and a threshold angle of 90 degrees. If the previous conditions are satisfied, the point will be considered to be overlapped with a triangle T if it is contained in the *overlap polytopes* of T.

A triangle T has two overlap polytopes. The *upper overlap polytope* of T is the volume defined by the plane that contains T and by the *overlap planes* associated with the edges of T. The overlap plane associated with an edge E of T is the plane that contains E and is orthogonal to the triangle adjacent to T along E. If T has no adjacent triangle along E, or T and its neighbor along E are non-convex, the overlap plane coincides with the plane orthogonal to T. Conversely, the *lower overlap polytope* of T is the symmetry of the upper overlap polytope with respect to T. Further details can be found in [5].

As mentioned above, the exterior edges that do not have all their exterior points overlapped with nearby triangles become occlusion edges. Each occlusion edge has associated the normal of the triangle that contains the edge, and a vector orthogonal to both the edge and its normal, and pointing out of the surface. The latter vector will be referred to as the occlusion edge's tangent. The previous normal and tangent vectors indicate directions in space where further exploration is likely to be worth it. In order to determine predominant groups of directions, a voting process based on the use of *orientation histograms* is utilized (Section 2.4). Orientation histograms are described next.

2.3 Spherical Discretization Maps (SDMs)

Orientation histograms are utilized to find clusters of both tangent and normal vectors associated with each occlusion edge. Orientation histograms require a discretization of the unitary sphere into uniform size cells. Each cell represents a set of orientations in space (a solid angle). The simplest technique for discretizing a unitary sphere consists of dividing it into parallels and meridians [1][6]. Unfortunately, the cells obtained in this way do not have a uniform size. To have uniform cells, geodesic domes and tessellations based on regular polyhedra have been proposed [6], but the problem then becomes the efficient mapping of orientations to their corresponding cells.

We propose *spherical discretization maps* (SDMs) as simple representations that allow a relatively uniform discretization of a unitary sphere with a simple way of mapping orientations to cells. The idea consists of dividing the sphere into a fixed number of parallels P. Then each parallel is divided into a number of cells that is proportional to the area covered by that parallel, the latter being approximated by the length of circumference of the parallel. The aim is that the equator has the maximum number of cells while the poles have a single cell.

SDMs are defined by a predefined number (multiple of four) of cells along the equator, CE. From it, $P+1$ parallels are defined, with $P = CE/2$. Given a certain parallel p, its corresponding elevation angle is $\varphi(p) = \pi/2(4p/CE-1)$. From the latter, the number of cells that belong to a parallel p is $\zeta(p) = 1$ if $\cos(\varphi(p)) = 0$ (i.e., p is equal to 0 or P and corresponds to a pole) and $\zeta(p) = \lfloor CE \cos(\varphi(p)) \rfloor$ otherwise. It is easy to show that $\cos(\varphi(p))$ is the ratio between the length of circumference of the parallel at elevation $\varphi(p)$ and the length of circumference of the equator. The orientation angle of a given cell c that belongs to a parallel p is obtained as $\theta(p,c) = 2\pi c/\zeta(p)$.

Conversely, given an elevation angle φ, $-\pi/2 \le \varphi \le \pi/2$, and an orientation angle θ, $0 \le \theta \le 2\pi$, the corresponding parallel p is obtained as $p(\varphi) = \lfloor (\varphi + \pi/2) P/\pi \rfloor$, whereas the cell inside p is calculated as $c(\varphi,\theta) = \lfloor \zeta(p(\varphi))\theta/2\pi \rfloor$. According to that, the pole at $\varphi = \pi/2$ is mapped to parallel P, the equator at $\varphi = 0$ to parallel $P/2$ and the pole at $\varphi = -\pi/2$ to parallel 0.

The resolution at which the sphere is discretized only depends on the number of cells along the equator. In our implementation, all the SDMs are defined with 20 cells along the equator. This leads to a discretization of the whole sphere into 11 parallels and a total of 126 cells (see Fig. 1).

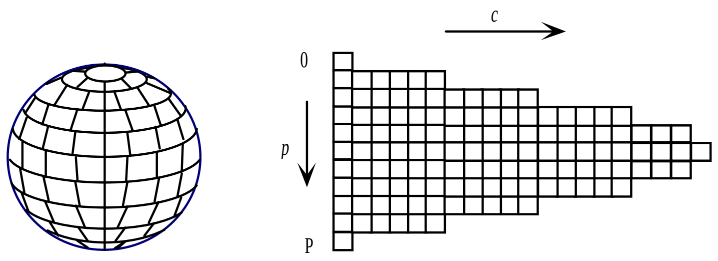

Figure 1. Example of a Spherical Discratization Map with 20 cells along the equator, leading to 11 parallels and a total of 126 cells.

The proposed algorithm requires three SDMs: one for representing the visited cells over the observation sphere and two more SDMs for the tangent and normal orientation histograms used during the voting process. The latter is described next.

2.4 Normal and Tangent Voting

At this point, each occlusion edge has associated a normal and a tangent vector (Section 2.2). Then, two orientation histograms represented as SDMs (Section 2.3) are considered: one for normals (*normal histogram*) and another for tangents (*tangent histogram*). Those histograms are the basis for a voting process that highlights predominant orientations of the occlusion edges.

Each tangent vector obtained above contributes with a vote in the tangent histogram and each normal vector with another vote in the normal histogram. Besides keeping a number of votes, every cell of the normal histogram also keeps the sum of the tangents associated with the normals that voted for that cell. Similarly, each tangent cell also keeps the sum of the normals associated with the tangents that voted there.

After all occlusion edges have been considered and their corresponding normals and tangents voted, the algorithm proceeds by looking for the cell that has received the maximum number of votes in either histogram. If that cell was already chosen for a previous point of view, the cell is discarded and a new maximum cell is found. A third SDM indicating visited cells over the observation sphere is kept.

At this point, two situations may arise corresponding to the two stages of the algorithm mentioned earlier in the introduction: (*a*) a non-visited cell is found with a number of votes above a certain *significance threshold* (24 votes in our implementation), or (*b*) all cells with a significative number of votes are marked as visited and the remaining cells have either no votes or a small number of votes below the significance threshold. The second case usually corresponds to the situation where only holes left by self-occlusion remain. Its associated processing is described in Section 2.5.

In the first case, the next view is computed as the result of the voting process as follows. If the maximum cell is found at the normal histogram, a distinctive set of occlusion edges is likely to belong to a surface with similar orientation. For example, this is the case of a hole in the middle of a planar or low curvature surface. In that situation, the next point of view should tend to fill up the hole by aligning along the normal associated with the winning cell. Conversely, if the maximum cell is found at the tangent histogram, a distinctive set of occlusion segments are likely to belong to a surface with changing orientation but similar tangents. This would be the case of a cylinder with one of its planar faces missing. In that case, the next point of view should tend to observe the missing "lid" by aligning along the tangent vector associated with the winning cell.

However, as pointed out in [10], a certain amount of overlap must be enforced to facilitate the registration of the new view with previously acquired ones. In our case this is done by utilizing the directions stored in the histogram cells to slightly modify the direction corresponding to the winning cell. Specifically, the new point of view is computed as the weighted average of unitary vectors such that the direction corresponding to the winning cell receives a weight of α while the direction stored in that cell receives a weight of $1-\alpha$. The value of α has been set to 0.7 experimentally.

2.5 Hole Filling

At this point, all the winning cells that have not yet been visited in both the tangent and normal histogram have a number of votes below the significance threshold. This typically corresponds to a situation in which all major surfaces have been acquired and only small holes due to self-occlusions are left.

The objective then becomes the determination of the next view that closes the largest hole. This is done in three steps. First, the different holes present in the exploration model are identified and the largest hole chosen as the target. Then, the surfaces from the exploration model susceptible to block the given hole are identified. Finally, the non-visited cell from where the target hole is visible at its largest extent is chosen as the next point of view. These steps are described in further detail next.

2.5.1 Determination of the Largest Hole

Given the current exploration model and the set of identified occlusion edges, the objective is to find out the group of occlusion edges that delimit the largest. This implies the determination of all holes present in the model and from them the largest one.

In order to separate the different holes, a *k*-nearest neighbor classifier (with $k = 2$) is applied to cluster the *extremes* of the occlusion edges contained in the exploration model. This classifier starts by assigning each extreme to a different cluster. Then, an iterative process finds out the *k* extremes closest to each extreme ξ and joins the clusters corresponding to both ξ and its *k* neighbors into a single cluster. In this way, all the extremes are partitioned into a collection of clusters, with each cluster corresponding to a hole or set of adjacent holes in the exploration model.

To prevent holes belonging to different parts of the scene from being incorrectly joined, two extremes are excluded from being considered as neighbors by the classifier if their distance is larger than a certain threshold and the angle between their associated normals is larger than 90 degrees. The normal of an extreme is the same normal vector associated with the occlusion edge to which the extreme belongs (which, in turn, is the normal of the triangle that contains that edge). The threshold δ has been set to twice the average length of all occlusion edges contained in the exploration model.

Once all holes have been determined, the one that is bounded by the largest number of occlusion edges is selected as the *target hole* to be filled out. The target hole is represented by the set of extremes corresponding to its occlusion edges. The average of those extremes is taken as the *hole's centroid*, and the sum of the normals associated with those extremes as the *hole's normal*.

2.5.2 Culling of the Exploration Model

In order to speed-up the process that determines the cells from where the target hole is visible, the exploration model is simplified in such a way that only the triangles susceptible to block the hole are considered for ray casting computations. This is done by going over all the triangles contained in the exploration model and selecting those whose normals have an angle lower than 90 degrees with respect to the hole's normal and whose vertex coordinates are above the hole's centroid along the direction of the hole's normal. Those triangles constitute the *culled model*.

2.5.3 Visibility Analysis

The last step of the hole filling stage selects a non-visited cell on the observation sphere from where the target hole is mostly visible, taking into account the possible overlaps produced by the surfaces already acquired during the exploration process. These surfaces are the ones contained in the culled model.

Usually, there will be more than one cell from where the target hole is fully visible. In that case it is necessary to chose one of them according to some optimality criterium. This criterium has been defined as follows. Let the *observation segment* of a cell be the 3D segment that connects the center of the observation sphere with the center of that cell. The optimality criterium consists of choosing, from all the cells from where the target hole is visible at the same extent, the one whose observation segment has the smallest angle (*observation angle*) with respect to the hole's normal. This corresponds to a cell from where the hole is observed with the lowest inclination and, thus, from where its maximum area can be appreciated.

However, when there are surfaces above the hole, this criterium may lead to views in which the hole is very close to one of the boundaries of those surfaces, and this usually produces vertical walls in the new acquired range image, which are removed after the approximation stage described in Section 2.1. This removal usually produces a new hole that will have to be targeted in further iterations. To prevent this behavior from happening, the target hole is dilated and the visibility analysis then applied to the dilated hole. The dilation process consists of computing for every hole's extreme a new point (*dilated point*) along the line that connects that extreme to the hole's centroid and at a distance from the original extreme half the distance between the centroid and the extreme. The dilation process is illustrated in Fig. 2.

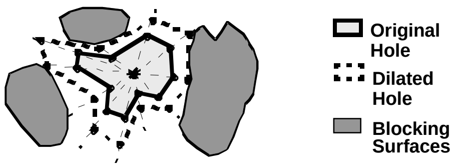

Figure 2. Example of the dilation of a hole and its effect on the selection of the point of view.

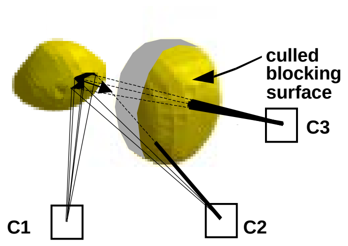

Figure 3. Visibility analysis. Cell C1 is the only one from where the hole is entirely visible considering the blocking surfaces of the culled model.

The visibility process traverses all cells on the observation sphere (126 cells in our implementation) discarding those which have already been visited or whose observation angle is larger than 90 degrees. For each of the remaining cells, the one from where the maximum number of dilated points is visible and whose observation angle is lowest is chosen as the next best view to fill up the target hole. To check if a dilated point is visible, the straight line between the cell's center and the dilated point does not have to intersect any of the triangles contained in the culled model (see Fig. 3).

3 Experimental Results

The proposed algorithm has been tested with ranges images obtained from a simulation tool developed for this project. This tool allows the set-up of 3D scenes with arbitrary objects imported from CAD (VRML and Robmod) and from real range images. The system allows the definition of camera positions over an observation sphere and the computation of dense range images. The simulator also provides the necessary transformation matrices between the local coordinate frames attached to the camera and the global observation frame.

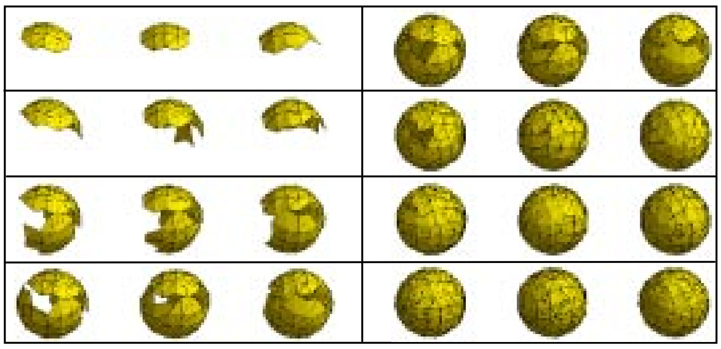

Figure 4. Partial sequence of the exploration of a scene containing three spheres. The whole sequence consists of 20 views, from which the last 6 views correspond to the hole filling stage. The result after the voting stage is shown at the third image in the right column.

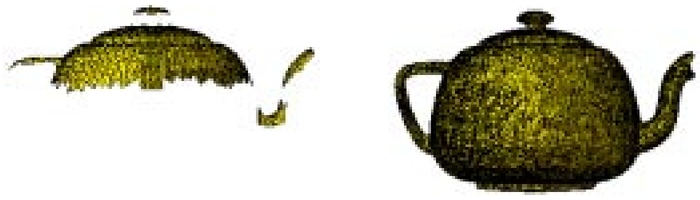

Figure 5. Exploration of a teapot. (*left*) Exploration model after the first view. (*right*) Exploration model at the end of the process in 31 views.

The first example corresponds to a simple scene containing three separate spheres. Fig. 4 shows a partial sequence of the exploration models from the first view to the last one. The whole sequence consists of 20 views. The CPU time to compute the 20 views was 63 sec. on a SGI Indigo II with a 175MHz R10000. The 99% of this time was spent by the occlusion edge detection process. The processing associated with the hole filling stage greatly depends on the location of the holes and the blocking surfaces. In this example, the maximum CPU time spent in that stage was 0.63 sec.

Fig. 5 shows the initial view and the result of the exploration of a teapot. 31 views are necessary from which 26 correspond to the voting stage and the others to the hole filling stage that closes 12 holes.

4 Conclusions and Future Lines

A two-stage technique has been presented for computing the positions where a range sensor should be positioned in order to acquire the surfaces of a 3D scene. The first stage determines the next view based on a voting scheme that takes into account the orientation of occlusion edges. The second stage finds out holes left by the first stage and applies visibility analysis to determine the views necessary to close them. Spherical discretization maps have been introduced as an efficient tool for implementing orientation histograms. Immediate work will consist of the development of space partition strategies necessary to accelerate the detection of occlusion edges. Future work will consist of the introduction of constraints in the exploration process, such as unreachable viewpoints.

References

[1] C. J. Connolly, The determination of next best views. *IEEE Int. Conf. on Robotics and* Automation, 1985, 432-435.

[2] M. A. García and L. Basañez, Efficient free-form surface modeling with uncertainty. *IEEE Int. Conf. on Robotics and Automation*, Minneapolis, USA, April 1996, 1825-1830.

[3] M. A. García and L. Basañez, Fast extraction of surface primitives from range images. *13th IAPR Int. Conf. on Pattern Recognition, Vol. III: Applications and Robotic Systems*, Vienna, Austria, August 1996, 568-572.

[4] M. A. García, A. D. Sappa and L. Basañez, Efficient approximation of range images through data dependent adaptive triangulations. *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, Puerto Rico, June 1997, 628-633.

[5] M. A. García, S. Velazquez, A. D. Sappa and L. Basañez, Autonomous sensor planning for 3D reconstruction of complex objects from range images. *IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, May 1998, 3085-3090.

[6] B. K. P. Horn, Extended Gaussian Images. *Proceedings of the IEEE*, vol.72, no.12, December 1984, 1671-1686.

[7] K. N. Kutulakos, C. R. Dyer and V. J. Lumelsky, Provable strategies for vision-guided exploration in three-dimensions. *IEEE Int. Conf. on Robotics and Automation*, 1994, 1365-1372.

[8] E. Marchand and F. Chaumette, Active sensor placement for complete scene reconstruction and exploration. *IEEE Int. Conf. on Robotics and Automation*, Albuquerque, USA, April 1997, 743-750.

[9] J. Maver and R. Bajcsy, Occlusions and the next view planning. *IEEE Int. Conf. on Robotics and Automation*, Nice, France, May 1992, 2043-2048.

[10] R. Pito, A sensor based solution to the next best view problem. *13th IAPR Int. Conf. on Pattern Recognition*, Vienna, Austria, August 1996, 941-945.

[11] M. K. Reed, P. Allen and I. Stamos, Automated model acquisition from range images with view planning. *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, June 1997, 72-77.

[12] P. Whaite and F. P. Ferrie, Autonomous exploration driven by uncertainty. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 3, March 1997, 193-204.