Proceedings of the 1998 IEEE
International Conference on Robotics & Automation
Leuven, Belgium • May 1998

Autonomous Sensor Planning for 3D Reconstruction of Complex Objects from Range Images

Miguel Angel García[†] Susana Velázquez[‡] Angel Domingo Sappa[‡] Luis Basañez[‡]

[†]Department of Computer Science Engineering
Rovira i Virgili University
Ctra. Salou s/n. 43006 Tarragona, Spain
magarcia@etse.urv.es

[‡] Institute of Cybernetics
Polytechnic University of Catalonia
Diagonal 647, planta 11. 08028 Barcelona, Spain
{velazquez, sappa, basanez}@ic.upc.es

Abstract

This paper presents a new technique for determining a small set of views that allow the observation and acquisition of the surfaces of the objects present in a target scene through a range sensor that moves over a sphere containing that scene. No a priori knowledge about the shape of those objects is assumed. Instead of applying costly visibility analysis techniques from the beginning as in most previous approaches, a two-stage algorithm is proposed. The first stage is responsible for getting the majority of object surfaces through a voting scheme based on occlusion edges. Then, a second stage applies visibility analysis to fill holes left by the first stage due to self-occlusions.

1 Introduction

Range images are gaining popularity in the computer vision and robotics communities owing to the development of fast and low-cost range sensors that allow the acquisition of 3D information directly from a scene. On the other hand, in order that this kind of dense representations may be successfully used in fields with real time constraints, such as robotics, it is necessary to simplify its processing. In this line, previous work by the authors focused on the efficient extraction of triangular approximations from range images [2][5] with the aim of simplifying further processing algorithms, such as image segmentation [4].

This work focuses on the problem of determining the positions where a range sensor should be placed in order to determine the surfaces of the objects present in a scene, with the aim of obtaining a 3D model of that scene, such as the one described in [3]. This is known as the *next-best-view* problem. As in several previous proposals, it is assumed that the sensor moves over a sphere of a certain radius that is centered at the middle of the scene, and that all objects to be reconstructed are inside this sphere.

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).

The next-best-view problem has received much attention in the robotics and computer vision communities. The majority of proposals (e.g., [1][9][11][12]) resort to the modeling of the unseen areas/volume of the scene to be reconstructed and then compute the amount of such unseen space that is visible from each allowed position of the sensor. The sensor 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 then chosen among the different candidate sensor positions by some kind of optimization process that maximizes the amount of unseen space discovered by the new view from that position. An optimization technique is also proposed in [13] where the uncertainty of the superquadrics used to model the perceived objects is minimized, the sensor position being related to the parameters of the superquadrics.

Unfortunately, all these stages, from the modeling of unseen volumes in space, to the determination of visibility regions or the final optimization process, are computationally costly. The computational complexity is related to the fact that the core of the next-best-view problem is the generation of the minimum number of views that allow the reconstruction of the scene. However, that problem is known to be NP-complete [9]. Therefore in many cases, it would be faster to move the sensor to a new position through a simple search strategy and to acquire and process a few more range images than, as in many previous proposals, to apply costly processes to come up with a solution that is not even guaranteed to be the best.

Taking those considerations into account, we propose a different strategy based on two stages (voting and hole filling). The first stage finds out the new point of view as the result of a voting process that only considers the location of occlusion edges, avoiding thus the computation of visibility regions. Occlusion edges are known to be useful for this problem from previous work [8][10]. This stage is quite fast and sufficient to acquire a significative amount of the surfaces present in the scene. The second stage is responsible for covering 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 views. The different range images obtained during the acquisition process can be integrated together by using a zippering algorithm such as [12].

The paper is organized as follows. The proposed algorithm is described in section 2. Section 3 presents experimental results based on a 3D graphical simulator. Section 4 gives conclusions and open lines of research.

2 Next-Best-View Generation

This section presents an algorithm for determining a set of views that allow the acquisition of the 3D surfaces of a given scene by means of a range sensor. No restrictions about the shape or number of objects are imposed.

The sensor is assumed to move over a sphere (the *observation sphere*) located at the center of the target scene and with a radius large enough to guarantee that all objects to be sensed are inside the sphere. Since the sensor is always aiming at the center of the sphere, its position is fully determined by a pair of orientation and elevation angles. For efficiency purposes, the observation sphere is discretized into a number of cells so 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 algorithm generates successive points of view until all surfaces of the scene are recovered. The process associated with each iteration can be summarized as follows. Given a new range image from the current point of view, a triangular mesh is generated and then integrated with the meshes obtained from previous views in order to obtain an estimation of the observed scene so far.

From this integrated mesh, occlusion edges are identified. *Occlusion edges* are triangle edges that belong to surface discontinuities in the triangular mesh (jump edges) and may occlude parts of the scene. For each occlusion edge, both a normal and a tangent vector are obtained. These vectors contribute with a vote in two *orientation histograms* (discrete approximations of Extended Gaussian Images [7]). After all occlusion edges have been considered, two situations may arise, leading to the two stages of the algorithm mentioned before.

The first situation occurs when a new direction is found in any of the histograms having a number of votes above a certain threshold. Moreover, this direction must correspond to a cell on the observation sphere that has not been visited yet. In that case, the next point of view is calculated based on that direction. In practice, this situation occurs during the first part of the process, leading to the recovery of the majority of surfaces of the scene.

The second situation occurs when all cells with a large number of votes have been visited, denoting occlusion edges that have not disappeared even when they were targeted with a specific viewpoint. In practice, this situation occurs at the end of the process when most of the surfaces have been recovered and only relatively small holes are left due to self-occlusions among objects. At this point, a hole filling strategy is applied. First, all separate holes are identified. Then the largest one is chosen as a target. The new point of view is selected by choosing the cell on the observation sphere that has not been visited yet and from which the largest amount of occlusion edges from the target hole are visible. This implies an analysis of visibility that takes into account the occlusions that may be produced by previously acquired surfaces.

Disregarding the stage, after the new point of view is determined, a new range image is acquired and a new iteration starts over. The whole algorithm will stop when no occlusion edges are found, indicating that no open surfaces are left, or when there are remaining holes but all cells on the observation sphere from where the holes should be visible have already been visited, and this implies either that it is not possible to observe those holes given the current observation sphere or that the objects in the scene contain open surfaces that cannot be closed. The latter situation could be detected at earlier stages, avoiding thus an exhaustive search over the observation sphere, by keeping track of the bounding box of each hole and marking as unfeasible those holes that cannot be reduced after a certain number of iterations.

The algorithm is fully described next.

2.1 Approximation of Range Images

The exploration starts with a predetermined point of view. Every time a new range image is acquired, it is approximated by a triangular mesh in order to speed-up further processing. This can be done in different ways, including the adaptive technique proposed in [5]. For the sake of conceptual simplicity though, we have opted for a simple uniform sampling of the range image, which already suffices our purposes. Each pixel selected from that sampling is converted to a 3D vertex referred to a local frame attached to the sensor. The coefficients of these transformation are obtained from the calibration of the range sensor. The sampling resolution needed for the next-best-view generation process should be, at least, half the minimum separation among objects.

Once we have an initial triangulation in space describing the surfaces of the scene, some triangles are removed in order to avoid that separate surfaces (e.g., surfaces belonging to different overlapped objects) may be joined.

Specifically, all triangles whose barycenters project onto the background of the range image are removed. Moreover, all triangles whose normal vectors have an angle with respect to the viewing direction lower than a certain threshold τ (100 degrees in this work) are also removed. This corresponds to a limit inclination (*breakdown angle* [11]) of $180 - \tau$ degrees. Finally, the mean μ and the standard deviation σ of the perimeter of all triangles are computed and those triangles whose inclination is higher than 60 degrees and whose perimeter is larger than $\mu + K\sigma$, $1 < K < 2$, are also removed. This heuristic sup-

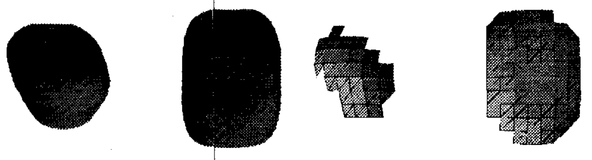
Figure 1: (*left*) Original 3D scene. (*right*) Triangular mesh approximating the range image obtained from the left scene.

presses triangles joining separate surfaces in the case where the vertices of the triangles do not lie right at an occlusion boundary but close to it. Those triangles might remain after applying the breakdown angle test.

Fig. 1(*right*) shows an example of the triangular mesh extracted from a range image of the scene shown in Fig. 1(*left*). The previous removal criteria ensure that the next-best-view generation process only considers existing object surfaces. This implies that areas with relatively high inclination are suppressed, such as the boundaries of curved objects. The result is that more views are necessary for recovering a whole scene. For example, a single sphere would not be recovered in only two views since only a small cap would be obtained from each view. The triangular mesh that approximates a range image will typically contain separate regions corresponding to different object surfaces (see Fig. 1).

At this point, the topology of the mesh is obtained by computing the list of vertices adjacent to every vertex. Two vertices are adjacent if they are the extremes of the edge of a triangle. From the mesh topology, the vertices that lie at the boundary of the regions contained in the triangular mesh are identified. They are referred to as *exterior vertices*. A vertex is exterior when its list of adjacent vertices is open, in the sense that the last vertex of the list does not coincide with the first one. Two adjacent exterior vertices constitute an *exterior edge*. The triangle that contains an exterior edge will be referred to as an *exterior triangle*.

2.2 Determination of Occlusion Edges

Given a new range image, a 3D triangular mesh is obtained as described above and all its exterior edges are identified. The vertices of this mesh are referred to a local Cartesian frame attached to the range sensor.

The triangular mesh obtained from the current point of view is integrated with the meshes obtained from the previous views in order to update a model (the *exploration model*) representing the knowledge about the real scene which is available to the system so far.

The aforementioned integration requires that the 3D coordinates of the triangular meshes obtained from each view, which are originally referred to a local coordinate frame, are referred to a unique global coordinate frame. Therefore, it is necessary to know the homogeneous transformation matrix that relates both the local and global frames. This transformation can be determined either from a calibration process or, in the typical case in which the sensor is mounted on a robotic arm, from the location and orientation of the terminal element of the robot.

It is important to emphasize that the aim of this integration is not the reconstruction of the scene, in the sense of the generation of a CAD model. That would require much more computational effort than what is necessary just to plan the next point of view. Thus, the proposed approach uncouples the reconstruction stage from the exploration one in order to speed-up the latter. Once all necessary range images have been acquired, the reconstruction process can be run off-line. Therefore, the proposed exploration model is just the concatenation of the different triangular meshes, with no merging involved.

The objective now consists of determining the location of occlusion edges. When the first triangular mesh corresponding to the first view is available, all its exterior edges are also occlusion edges. However, as new views are acquired and new meshes available, there can be overlaps between meshes, and many edges that were originally exterior may become interior. Thus, the problem of detecting occlusion edges becomes the problem of finding what exterior edges from the triangular meshes obtained so far are not overlapped when a new triangular mesh arrives. Specifically, once the last view has been integrated, the algorithm detects the possible overlap between every exterior edge and its nearby triangles from the exploration model. In order to do that, the distance between the centroid of the current exterior triangle (the triangle that contains the exterior edge) and the centroids of all the other triangles from the model are compared. Two triangles are candidate to be overlapped if the previous distance is below a certain threshold. This threshold is twice the sum of the radius of both triangles, with the radius of a triangle being the distance between its centroid and one of its vertices.

As all triangles of the exploration model are oriented counter-clockwise, there is an implicit ordering of the vertices of all exterior edges. Thus, an edge can be identified by its first vertex. Since a large exterior edge may be overlapped along a small section of its extent—that leads to partial occlusion edges—and that situation must be detected, a small number of equidistant points over each exterior edge are selected. These points (seven in the current implementation), jointly with the first edge vertex, constitute the *exterior points* of the exterior edge. Each exterior point is tested for overlap against its nearby triangles.

Each exterior point has an associated normal vector corresponding to the triangle to which the exterior edge belongs. Therefore, the detection of overlap between an edge and a triangle finally becomes the detection of overlap between a 3D point (with its corresponding normal) and a triangle. There are two special cases in which no overlap is decided in considering that the point and the triangle belong to separate surfaces. First, when the normal associated with the point has an angle of more than 90 degrees with respect to the normal of the triangle. Second, when the distance between the point and the triangle's

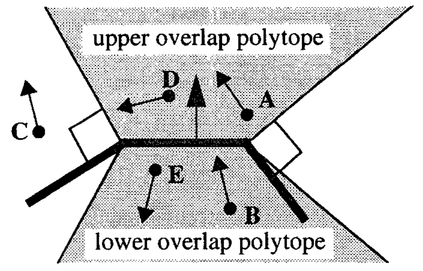

Figure 2: 2D illustration of the determination of overlap between a point (with associated normal) and a triangle. Overlap polytopes are shown in dark. Points **A** and **B** are overlapped while points **C**, **D** and **E** are not.

plane is above a certain threshold (in the implementation, half the average length of the edges of the triangle).

If the previous conditions are not satisfied, the point will be considered overlapped with the triangle if it belongs to either the upper or lower *overlap polytopes* of the triangle. The upper overlap polytope of a triangle is defined as the intersection between the positive half-space supported by the triangle's plane and the positive half-spaces supported by three *upper overlap planes* associated with the edges of the triangle. Given an edge E of a triangle T, the upper overlap plane associated with E is the plane that contains E and is orthogonal to the plane of the triangle T_E adjacent to T along E, if that neighbor exists, or orthogonal to the plane of T if it does not or if triangles T_E and T form a non-convex surface. Conversely, the lower overlap polytope is the symmetry of the upper overlap polytope with respect to the plane defined by triangle T.

Fig. 2 illustrates the concept of overlap polytopes and overlap detection considering a 2D section of three triangles shown with a thickened polyline. In that example, points **A** and **B** are considered overlapped with the central triangle T while points **C**, **D** and **E** are not. Points **D** and **E** do not pass the normal orientation test whereas point **C** is outside the upper and lower overlap polytopes of T.

Those exterior points that are not overlapped with any nearby triangles will be considered to be *occlusion points*. An exterior edge whose exterior points are overlapped is relabeled as an interior edge and not considered for further overlaps. Exterior edges that contain occlusion points are considered to be occlusion edges.

2.3 Spherical Discretization Maps (SDMs)

As pointed out in the initial description of the algorithm at the beginning of Section 2, orientation histograms are utilized throughout the process. Orientation histograms require a discretization of the unitary sphere into cells of uniform area, with each cell representing a set of orientations in space (a solid angle). The simplest technique for dividing a sphere defines parallels and meridians [1][7]. However, the obtained cells do not have uniform area. To have uniform cells, geodesic domes and tessellations based on regular polyhedra have been proposed [7], but the problem then becomes how to map orientations to their corresponding cells efficiently.

We propose *spherical discretization maps* (SDMs) as a simple representation that allows a relatively uniform discretization of a sphere with a simple way of mapping orientations to cells. SDMs are obtained by dividing the sphere into a fixed number P of strips. Each strip is associated with a parallel of the sphere and is divided into a number of cells that is proportional to the area covered by the strip. The aim is that the equator has the maximum number of cells while the poles have a single cell.

SDMs are defined by a certain number of cells CE along the equator, with CE being a multiple of four. From it, $P+1$ parallels are defined with $P = CE/2$. Given a certain parallel p, its corresponding elevation angle is $\varphi_p = \pi/2\,(4p/CE-1)$. The number of cells that belong to a parallel p is defined as $\zeta_p = 1$ if $\cos(\varphi_p) = 0$ (i.e., p is a pole equal to 0 or P) 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 φ_p and the length of circumference of the equator. Then, 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, $0 \le \theta < 2\pi$, the corresponding parallel p is obtained as $p_\varphi = \lfloor(\varphi+\pi/2)P/\pi\rfloor$, whereas the cell inside p is calculated as $c_{\varphi,\theta} = \lfloor\zeta(p_\varphi)\,\theta/2\pi\rfloor$.

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

2.4 Normal and Tangent Voting

Two *orientation histograms* represented as SDMs (see Section 2.3) are considered now, one for normals (*normal histogram*) and another for tangents (*tangent histogram*). Those histograms are the basis for a voting process intended to highlight predominant orientations of the occlusion points as summarized below.

For each occlusion point (see Section 2.2), two unitary vectors are obtained. First, the normal of the triangle to which the occlusion point belongs. Second, a tangent vector obtained as the cross product between the previous normal and the corresponding occlusion edge. That tangent is pointing out of the surface.

Each tangent vector obtained above contributes with a vote to the tangent histogram and each normal vector with another vote to the normal histogram. Besides keeping a number of votes, every cell of the normal histogram also

$\lfloor x\rfloor$ the greatest integral value less than or equal to x.

keeps the resultant of the vector addition of all the tangents associated with the normals that voted for that cell. Similarly, each tangent cell also keeps the resultant of the normals associated with the tangents that voted there.

After all occlusion points have been considered and their corresponding normals and tangents voted, the algorithm proceeds by looking for the cell with the maximum number of votes in both histograms. If that cell was already considered for a previous point of view—a SDM with visited cells over the observation sphere is kept—the cell is discarded and a new maximum cell is found.

At this point, two situations may arise which correspond to the two stages of the algorithm mentioned earlier in the introduction: (*a*) a non-visited cell is finally found with a number of votes above a certain threshold and (*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 aforementioned threshold. The second case corresponds to the situation where only holes left by self-occlusion remain and 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 points are likely to belong to a surface with similar orientation. For example, this is the case of a hole in the middle of a 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 points 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 [11], a certain amount of overlap must be enforced to facilitate the registration of the new view with previously acquired ones. In this work, this is done by utilizing the directions stored in the histogram cells. Specifically, the new point of view is defined by a weighted average of unitary vectors in which the direction corresponding to the winning cell receives a certain weight α while the direction stored in that cell receives a weight of $1-\alpha$. In the current implementation, α was set to 0.7. For example, if the cell in the tangent histogram wins, the new point of view will be aligned along the winning tangent with a 30% deviation towards the resultant of the normals of the occlusion points that voted for that cell.

2.5 Hole Filling

At this point, the winning cells that have not been visited yet in both the tangent and normal histograms have received a number of votes below a certain threshold (24 votes in our case). This basically corresponds to a situation in which all major surfaces have been acquired and only small holes due to self-occlusion are left. The objective then becomes the determination of the next view that closes the largest hole.

The algorithm proceeds by segmenting all found occlusion edges into isolated components through a k-nearest neighbors classifier which is fed with the central 3D coordinates of each occlusion edge. The class with the largest number of occlusion edges is chosen as the target hole. A centroid for that hole is computed by averaging the centers of its occlusion edges. A hole's normal is also determined by adding the normals of the triangles that contain the hole's occlusion edges.

The objective now is to find out all non-visited cells of the observation sphere from where the largest amount of occlusion edges are visible. The sensor position associated with each non-visited cell is computed. From each position a ray (straight line) is sent to the extremes of the target occlusion edges. This ray is tested for intersection against all the triangles from the exploration model whose normals have an angle lower than 80 degrees with respect to the hole's normal and whose vertex coordinates are above the hole's centroid in the direction of the hole's normal (this is a culling process applied to the triangles of the model in order to increase efficiency).

The next point of view is computed from the cell from where the largest number of occlusion edges is visible and whose associated viewing direction forms an angle with the hole's normal closer to 180 degrees.

It is important to realize that it is not necessary to fill up each hole in a single view. After the next view is acquired, the hole is guaranteed to become smaller, if not to disappear. Then the algorithm is started over, and will proceed with the remaining holes until the termination conditions (pointed out at the initial description in Section 2) are satisfied.

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 sensor 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 sensor and a unique global frame attached to the scene.

The first example corresponds to a scene containing two separate rock-like objects. Fig. 3 shows a partial sequence of the updated exploration model of the scene from the first view to the last one. The whole sequence consists of 19 views. The first 16 views were obtained by applying the voting scheme. From them, 10 views were obtained as a result of maxima at the normal histogram and 5 views of maxima at the tangent histogram.

The result after the voting scheme, shown at the bottom-left image in Fig. 3, contained 4 separate holes that were closed in three more views. Fig. 4(*left*) shows a

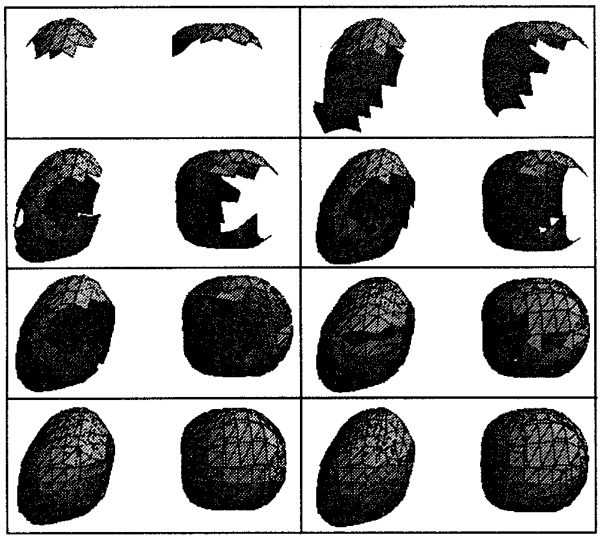
Figure 3: Sequence of the exploration of a scene containing two rock-like objects. The whole sequence consists of 19 views, from which the last 3 views correspond to the hole filling stage. The result after the voting stage is shown at the bottom-left image.

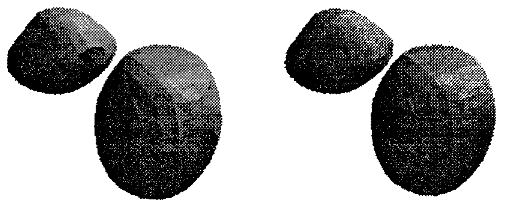
Figure 4: Application of the hole filling stage (*left*) Exploration model after the voting stage. (*right*) Exploration model after the largest hole is closed in one more view.

detail of the largest hole left by the voting scheme. That hole is closed in just one view leading to the result shown in Fig. 4(*right*). The CPU time to compute the 19 views was 150 sec. on a SGI Indigo II with a 175MHz R10000. Fig. 5 shows the initial view and the result of the exploration of a RX-90 robot. 36 views are necessary from which 29 correspond to the voting stage and the others to the hole filling stage that closes 16 holes.

4 Conclusions and Future Lines

A two-stage technique has been presented for computing a small set of views necessary to obtain all the surfaces of a 3D scene from range images. The first stage determines the next view based on a voting scheme that takes into account the orientation of occlusion edges. The second stage determines 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. Future work will consist of the introduction of constraints in the exploration process, such as unreachable viewpoints [9].

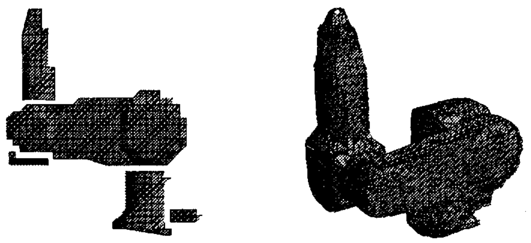
Figure 5: Exploration of a RX-90 arm. (*left*) Exploration model after the first view. (*right*) Exploration model at the end of the process in 36 views.

5 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, Fast approximation of range images by triangular meshes generated through adaptive randomized sampling. *IEEE Int. Conf. on Robotics and Automation*, Nagoya, Japan, May 1995, 2043-2048.

[3] 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.

[4] 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.

[5] 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.

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

[7] K. Kutulakos, C. Dyer and V. 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] G. Turk and M. Levoy, Zippered Polygon Meshes from Range Images. *SIGGRAPH '94*. 311-318.

[13] 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.