An autonomous six-DOF eye-in-hand system for in situ 3D object modeling
Lila Torabi and Kamal Gupta

The International Journal of Robotics Research 2012 31: 82 originally published online 31 October 2011
DOI: 10.1177/0278364911425836

The online version of this article can be found at:
http://ijr.sagepub.com/content/31/1/82

Published by:
SAGE
http://www.sagepublications.com

On behalf of:
Multimedia Archives

Additional services and information for The International Journal of Robotics Research can be found at:

Email Alerts: http://ijr.sagepub.com/cgi/alerts
Subscriptions: http://ijr.sagepub.com/subscriptions
Reprints: http://www.sagepub.com/journalsReprints.nav
Permissions: http://www.sagepub.com/journalsPermissions.nav
Citations: http://ijr.sagepub.com/content/31/1/82.refs.html

>> Version of Record - Jan 12, 2012

Proof - Oct 31, 2011

What is This?
An autonomous six-DOF eye-in-hand system for in situ 3D object modeling

Liila Torabi and Kamal Gupta

Abstract

We present an integrated and fully autonomous eye-in-hand system for 3D object modeling. The system hardware consists of a laser range scanner mounted on a six-DOF manipulator arm and the task is to autonomously build a 3D model of an object in situ where the object may not be moved and must be scanned in its original location. Our system assumes no knowledge of object shape or geometry other than that it is within a bounding box whose location and size are known a priori, and, furthermore, the environment is unknown. The overall planner integrates the three main algorithms in the system: one that finds the next best view (NBV) for modeling the object; one that finds the NBV for exploration, i.e. exploring the environment, so the arm can move to the modeling view pose; and finally a sensor-based path planner, that is able to find a collision-free path to the view configuration determined by either of the two view planners. Our modeling NBV algorithm efficiently searches the five-dimensional view space to determine the best modeling viewpoint, while considering key constraints such as field of view (FOV), overlap, and occlusion. If the determined viewpoint is reachable, the sensor-based path planner determines a collision-free path to move the manipulator to the desired view configuration, and a scan of the object is taken. Since the workspace is initially unknown, in some phases, the exploration view planner is used to increase information about the reachability and also the status of the modeling view configurations, since the view configuration may lie in an unknown workspace. This is repeated until the object modeling is complete or the planner deems that no further progress can be made, and the system stops. We have implemented the system with a six-DOF powercube arm and a wrist mounted Hokuyo URG-04LX laser scanner. Our results show that the system is able to autonomously build a 3D model of an object in situ in an unknown environment.

Keywords

Path planning for manipulators, range sensing

1. Introduction

Automated model acquisition of 3D objects is an important capability in a variety of application domains, including automatic inspection, manipulation, reverse engineering, and service robotics. Such an automated system, in our opinion, comprises three broad modules: (i) 3D model building which primarily concerns itself with the scanning and modeling of the object, and essentially deals with machine vision/graphics aspects; (ii) view planning, i.e. to determine the scanner pose from which to scan the object, which has both vision and robotics aspects; and (iii) moving the scanner positioning mechanism (a robot or a turntable) to desired scanning poses, which brings in the path planning aspect.

The pure vision/graphics aspects are encapsulated in the 3D model building cycle, shown in the left block in Figure 1, consisting of three main phases: a) scan, b) register, and c) integrate. A scan is taken from a camera pose, and acquired range images are registered using standard techniques such as the Iterative Closest Point (ICP) method (Besl and McKay 1992; Rusinkiewicz and Levoy 2001). Finally, registered images are combined into a single object model, a process commonly known as integration. The cycle is iterated until a complete model is built or some termination criteria are satisfied. An example of this would be a person using a handheld scanner, such as a Polhemus FastSCAN (Polhemus 2011), to take multiple scans, and the 3D model building cycle being carried out by the system. Note that steps (ii) and (iii) are carried out by the user and there is no autonomy here.

School of Engineering Science, Simon Fraser University, Burnaby, Canada

Corresponding author:
Liila Torabi, Robotic Algorithms & Motion Planning (RAMP) Lab, School of Engineering Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada. 
Email: ltorabi@sfu.ca
To automate this process, step (ii), i.e. view planning (also called next best view or NBV) and step (iii), i.e. path planning, are added as shown in Figure 1. The next best view problem is to determine the best scanner pose to scan the object from. The primary purpose of an NBV algorithm is to ensure that the entire surface of an object will be scanned. The preference is to minimize the number of scans taken to build the model, since scanning, processing the scans, and integrating them into one object model is often quite time consuming. This step, however, obviously depends on the positioning mechanism on which the scanner is mounted. Most published works in computer vision plan the next best view based on large and fixed setups that possess limited mobility, e.g. a turntable. The viewpoint space is mostly limited to one or two degrees of freedom (DOFs), and at best provides limited quality/coverage models for a restricted class of objects. They also all assume that the desired viewpoint is reachable. For a truly automated system, particularly for in situ objects, where the object must be scanned in its place (e.g. imagine an objet d’art in a museum that cannot be moved from its display location), we argue that there is a need for at least a six-DOF positioning mechanism, e.g. a manipulator arm, so that the object can be viewed from an arbitrary viewpoint, significantly increasing the likelihood that a complete model of an object could be obtained. The challenge, of course, is to keep the NBV computationally feasible, since the space of possible viewpoints increases tremendously. We show in this work that such a goal is achievable with efficient NBV algorithms coupled with the great increase in computing power of standard PCs in recent years. Furthermore, using a six-DOF (or even higher) manipulator as a scanner positioning device necessitates the use of path planning to move the robot while avoiding collisions with obstacles, which must also be sensed by the robot so that it can avoid them, since the environment is initially unknown. While such coupling has been explored in site modeling works (Sequeira et al. 1996; Blaer and Allen 2007), the “positioning mechanism” is a mobile robot and, in addition, the NBV problem has a different flavor in these works. Along another line, several works have proposed and implemented sensor-based path planners for eye-in-hand systems in unknown environments, but these works did not consider object modeling and the associated constraints such as overlap between images (Kruse et al. 1996; Renton et al. 1999; Yu and Gupta 2004). Such sensor-based path planners, as in Yu and Gupta (2004), use a different type of view planner, an exploration view planner, that plans views to explore the environment so that the robot can move around. For instance, in our context, the best view configuration determined for scanning the object may itself be unreachable because much of the environment is unknown to the robot or a part of the configuration itself may lie in unknown space. In such cases, another view must be planned to make more of the workspace known. (This exploration view planner is embedded within the path planner in Figure 1 and is not explicitly shown so as not to clutter the diagram.)

More recently, motivated by the need for autonomy, several works in the humanoid and service robotics area have addressed integrating object recognition, path planning, and grasping, such as the DESIRE project of European Union (Kuehnle et al. 2009; DESIRE 2011). However, in these systems, usually the main task involves grasping or recognition, and thus they exploit pre-built object models to find a suitable viewpoint for either grasping or recognizing (Eidenberger et al. 2009). Furthermore, there is no tight coupling between moving the sensor and moving the arm, as is the case in our system, where the sensor is mounted on the arm. As a result there is no view planning for exploration involved, whereas it is an integral component of our overall system; in fact, this additional maneuverability is a key advantage of our system. Foissotte et al. (2010) presents an object modeling system via a head-mounted stereo range camera on a humanoid. Feasible postures of the humanoid are generated via a two-step NBV algorithm (see Section 2 for more on it). A key difference from our work is that the environment (other than the object to be modeled) is considered known and hence there is no view planning for exploration, and furthermore no path planning is integrated in the system.

In this paper we present integrated view planning and motion planning for a six-DOF eye-in-hand object modeling system. Two key contributions of our work are (i) a modeling NBV algorithm that efficiently searches the five-dimensional space (because of the symmetrical conical shape of the scanner field of view, scanner pose only needs 2-DOF in orientation, pitch and yaw – roll is immaterial) to determine the best modeling viewpoint, and (ii) integrating the modeling NBV with an exploration NBV (Torabi et al. 2007), along with a sensor-based motion planner (SBIC-PRM: Yu and Gupta 2004), to make the desired modeling
view pose reachable, if required, and plan a collision-free path to it. In our modeling NBV, the viewpoint space representation allows us to map the viewing directions for the target areas, simply by defining a projection function. Our resulting system is a fully autonomous system capable of modeling in situ objects without any human intervention. In addition, we develop and prove an automatic termination criterion that can determine if the object model is complete. We show that the system is able to autonomously scan an object in environments while avoiding collisions with unknown obstacles. Each iteration, including all view and path planning and workspace update, takes about 4 minutes with the current system on a relatively old machine. However, with several efficiencies incorporated in the algorithms, we are confident that these runtimes can be brought down significantly. To the best of our knowledge this is the first work which considers both view and path planning aspects for a six-DOF fully autonomous 3D object modeling system.

2. The NBV problem for object modeling: background and related work

The view planning problem for object modeling has been well studied and this section is a brief overview of the problem, summarized from existing literature, e.g. Scott et al. (2003). The object surface is analyzed to determine where to scan next, then a viewpoint which can scan that area is computed, and finally the positioning system should move the scanner to that pose, if possible. Thus, the problem involves reasoning about the state of knowledge of three spaces: object surface $S$, viewpoint space $V$, and scanner/imaging workspace $W$. The object surface, $S$, is a 2D parameterized space, analyzed to determine the target areas for the next scan. The viewpoint space, $V$, defines the set of possible viewpoints, which in the general case is 6 dimensional. The scanner/imaging workspace, $W$, is the 3D region in which the positioning device moves and the laser camera takes scans, and in which visibility and collision constraints are addressed.

The modeling NBV is essentially composed of the following two main subproblems (Pito 1999):

1) **What to scan**, i.e. determining the target areas to be scanned, and their corresponding viewing directions. The goal is to identify and scan previously unseen portions of the scanner/imaging workspace, which possibly contain parts of the object surface. A common approach is to use range discontinuities which indicate target surface areas not yet scanned, thus providing cues for viewing direction. The target area could be either a volumetric element (octree node or voxel) or a surface element. If the target area is a surface element, the viewing direction is usually calculated by some analysis of the target area, such as normal estimation, but when the target area is a volumetric element, the viewing direction essentially points to a scanner position in $V$.

2) **How to scan**, i.e. specifying which target areas are visible from each viewpoint in $V$, given the scanner model (often called the scanner constraint). A brute-force method for such visibility checks would be ray-tracing. Note that occlusion (visibility constraint) by other objects (or parts of the same object for non-convex surfaces) also needs to be taken into account. Additional sets of constraints such as overlap with already scanned parts between successive views (for registration) also need to be considered for viewpoint selection. These constraints involve some form of ray-tracing, and can be computationally intensive. Hence a key challenge, particularly for a high-dimensional viewpoint space, as in our case, is to keep the computation fast and efficient to retain the online nature of the system.

The existing view planning methods could be classified mainly as volumetric (Connolly 1985; Reed et al. 1997; Papadopoulos-Orfanos and Schmitt 1997; Massios and Fisher 1998; Banta and et al. 2000) or surface-based (Maver and Bajcsy 1993; Whaite and Ferrie 1997; Pito 1999; Chen and Li 2005; He and Li 2006; Chen et al. 2008). **Volumetric** methods select viewpoints by reasoning about the knowledge of $W$, whereas **surface-based** methods reason about knowledge of $S$. We also classify these methods based on degrees of freedom of the viewpoint space $V$, and whether the viewpoint is calculated by searching $V$ based on the entire set of target areas (global), or is computed for one target area at a time (local).

One of the earliest papers, Connolly (1985), used an octree representation of the imaging workspace as empty, occupied, or unseen. The viewpoint space $V$ is 2-DOF, the surface of a sphere with view direction normal to the sphere. They developed two algorithms, one using a global search of $V$ in which the viewpoint with the largest unseen volume is selected. This is determined via ray-tracing for all viewpoints on the sphere. Direct extension of this method to high-dimensional view spaces would be computationally expensive. The second algorithm is incremental, and carries out visibility analysis in a local manner by examining faces in the octree common to both unseen and empty voxels. Banta (Banta and et al. 2000) chooses the viewpoint with the greatest number of unknown voxels. Massios (Massios and Fisher 1998) adds a quality factor (angle between estimated local surface normal and viewing direction) to enhance the previous voxel occupancy methods.

Maver (Maver and Bajcsy 1993) presented an NBV algorithm for a 1-DOF viewpoint space (a turntable). Unclosed view angles for target regions were determined via ray-tracing and accumulated in a histogram, and the next best view was determined from histogram maxima. Pito’s work (Pito 1999) was a significant contribution to the NBV literature. He chose small rectangular patches attached to occluding edges of the seen surface as a cueing mechanism for a global search of viewpoint space $V$. He used an intermediate “positional space” (PS), placed between
the object and the scanner workspace. It facilitates a succinct representation of what must be scanned (via the use of observation rays) and what can be scanned (via the use of ranging rays) in a single data structure. For the definition of observation ray and ranging ray please refer to Section 3.3. He used a turntable (1-DOF viewpoint space) as a positioning system, with 72 possible viewpoints. In his work, the memory usage and processing time are both proportional to the product of the size of the intermediate positional space and the size of the viewpoint space. Thus, clearly, directly extending his method to a high-DOF viewpoint space would be computationally expensive both in memory usage and processing time.

In our NBV algorithm, the observation rays of target areas are directly projected to the viewpoint space \( V \), without any use of the intermediate positional space. The visibility of the target areas over the whole viewpoint space are then efficiently computed through a two-level (position followed by orientation) projection computation. This technique allows us to efficiently determine the global maximum over the densely sampled five-DOF viewpoint space.

Several other works search the viewpoint space locally; as a result, the number of scans required could be quite large and hence the time taken to build the model could be quite large. For instance, Whaite (Whaite and Ferrie 1997) used the uncertainty in the modeled (using superellipsoids) object as a guide for taking the next view. Chen (Chen and Li 2005) predicts the curvature of the unknown area of the object and uses it to drive the exploration direction. His method works mostly for smooth surfaces, and has problems for objects with sharp boundaries. Although he uses a 6-DOF viewpoint space, since his algorithm is local there is no search involved, also no path planning is considered. In He and Li (2006), He selects the viewpoint which gives the maximum known partial modeling boundary integral value of the vector fields as the next best view position. In his work the viewpoints are located on the surface of a sphere pointing to the center.

More recently, as mentioned earlier, in Foissotte et al. (2010), the object is modeled via a head-mounted stereo range camera on a humanoid. The NBV algorithm is a two-step process. The first step takes into account several constraints due to the camera and the humanoid head. It minimizes a composite objective function that incorporates several constraints. A local quadratic approximation through a deterministic iterative sampling is carried out at a pre-selected sparse set of view positions distributed around the object, and the minimum over all these runs is selected as the best camera pose. The second step takes into account posture constraints to generate a suitable posture of the humanoid. Such local minimization techniques are prone to getting stuck in local minima, particularly for complex objective functions, and fixed sampling may outperform them (as the authors themselves point out).

3. Our modeling NBV algorithm

3.1. Notation

A brief explanation of our notation is as follows. Capital roman letters such as \( V, W, S \) represent sets or spaces, and lower case roman letters represent their corresponding elements. Generally, subscript \( o \) denotes orientation and \( p \) denotes position. So, for example, a viewpoint is denoted by \( v \) which consists of \( v_p \), the viewpoint position, and \( v_o \), the viewpoint orientation.

3.2. Representation of spaces \( W \) and \( S \)

We represent the workspace, \( W \), as an octree, primarily because it facilitates occlusion checks in view planning, and collision checks in path planning. The octree model is constructed from the range image, as in Yu and Gupta (1998). Briefly, the voxel hit by the ranging ray is categorized as obstacle, the intermediate voxels along the ranging ray will be considered as free, and the region behind the “hit voxel” stays as it was.

The seen part of the object’s surface, \( S \), is represented with a point cloud, which facilitates registration and also subsequent mesh generation to display the constructed object model. It also aids in determining surface normals, needed for determining valid viewing directions for target areas. Please note that since the object (whose model is to be constructed) is also part of the workspace, it is also represented in the workspace octree. This second representation is used for self-occlusion checks and to verify if a target point has been scanned before.

3.3. Target patches, laser scanner model, viewpoint space

Each target point, \( t_{pi} \), is a point from the unknown region bordering the seen part of the object surface. The small surface patch associated with each target point \( t_{pi} \), together with normal vector \( n_i \), is called the target patch, and is a primary estimation for an unseen part of the object which is desired to be viewed in the next scan. Although the best viewing direction for each target patch is the normal vector \( n_i \), the patch is also scannable from directions that form an angle less than or equal to the breakdown angle \( (\theta_b) \) with the normal vector of that surface patch. A ray or a viewing direction, \( o_t \), which emanates from \( t_{pi} \) with \( o_t \cdot n_i \leq \theta_b \), is an observation ray for \( t_{pi} \). The set of all observation rays for each target patch will be referred as its observation cone, and it represents the set of rays that can scan \( t_{pi} \). We can look at the range sensing process from either the “sensed surface” perspective or from the laser scanner perspective. From the scanner perspective, the range image obtained in each scan consists of the readings of a set of ranging rays\(^1\) emanating from the origin of the scanner frame, within the field of view (FOV) of the scanner. The
In a range image, the set of range discontinuities are key events, because the surface defined by these discontinuous ranges, known as occlusion surfaces, are the border between seen and unseen areas. These discontinuities could be caused by either (i) non-convex areas, or (ii) a surface edge in a non-smooth object, or (iii) a ranging ray tangential to the object surface. In Figure 2a, we illustrate a discontinuity as ray $R_i$, hits the object at $p_{1\text{near}}$, and a neighboring ray $R_{i+1}$ hits the object at $p_{1\text{far}}$. The line connecting $p_{1\text{near}}$ and $p_{1\text{far}}$ is an occlusion line, and a contiguous set of occlusion lines is called an occlusion surface, $S_{\text{occ}}$. Any point on $S_{\text{occ}}$ could be considered as a target point, but we only get the $k$ points with uniform resolution within distance $d_{\text{res}}$ from the scanned surface, such as $p_{1\text{near}}$ and $p_{1\text{far}}$, as shown in Figure 2b. Key reasons for this choice are: (i) since the object’s surface must continue into the unseen volume (from the boundary of the seen surface), this is a good place to look for more of the object’s surface; (ii) choosing target points close to a seen surface facilitates estimation of surface normals, needed for choosing viewing directions (explained later); and (iii) it automatically satisfies overlap constraints, essential for registration. Variations of this theme could use more points on the occlusion surface, $S_{\text{occ}}$; in this case the viewing direction will be normal to the occlusion surface. This will require additional explicit checks for overlap constraints.

After the list of target points is extracted, we need to estimate their corresponding surface normals, which are needed to determine their valid viewing directions. To estimate the surface normal for each target point, $p_{tp}$, first the neighbor points for $p_{tp}$, on the object point cloud and/or occlusion surface are computed through building a kdtree, and then the normal vector for the estimated surface of these neighbor points is calculated based on a principal component analysis (PCA) method (Hoppe and et al. 1992; Dey and et al. 2005).

The neighborhood points for surface normal estimation are chosen differently for $p_{1\text{near}}$ and $p_{1\text{far}}$ as follows. For $p_{1\text{near}}$, where a part of the object surface is not scanned because of occlusion, the unseen surface of the object close to the seen border could be estimated by continuing the seen surface. Therefore, the points from the object point cloud, which are in the neighborhood radius of a target point, form a local estimate for its target patch. To do this efficiently, we build a kdtree for the point cloud data, and search it to identify neighboring points for $p_{1\text{far}}$ (Atramentov and LaValle 2002).

For $p_{1\text{near}}$, as is shown in Figure 2b, it is less likely that we can accurately estimate the surface based on it’s neighbors in the point cloud model, since the seen points in the point cloud are not sufficient to calculate a good estimate for the unseen surface. Hence, for calculating the normal vector of the surface patch, the kdtree is built for the union of points on the point cloud model and the points on $S_{\text{occ}}$ (where the unknown occluded surface lies).
At the end of this step we will have a list of target patches, by storing each $tp_i$ and their corresponding normal direction $n_i$ in a list (see Algorithm 1).

**Algorithm 1**: Target patch list generator

**Input**: Last scanned range image, list of target patches

**Output**: The updated list of target patches

Extract the range discontinuities in the scanned image;

for every $i$'th discontinuity do

Determine $p_{near}$ and $p_{far}$;

if There is no $p_{far}$ then

Compute the occlusion line $O_{p_{near}}$;

Calculate $tp_{near}$;

else

Compute the occlusion line $p_{near}p_{far}$;

Calculate $tp_{near}$ and $tp_{far}$;

end

end

Store all $tp_i$ in a list;

Build a kdtree for the object pointcloud as $Kd_o$;

for every $tp_{near}$ in the list do

Compute the closest neighbors for $tp_{near}$ on $Kd_o$;

Estimate the patch normal $n_{near}$ using PCA method;

end

Build a kdtree for the union of object pointcloud and target points as $Kd_{tot}$;

for every $tp_{far}$ in the list do

Compute the closest neighbors for $tp_{far}$ on $Kd_{tot}$;

Estimate the patch normal $n_{far}$ using PCA method;

end

end

Add all the target patches ($tp_i, n_i$) to the existing list.;

3.5. How to scan

Having computed the set of target patches, for each patch ($tp_i, n_i$) in this set, the valid viewpoints from which $tp_i$ is visible will be computed. For a patch centered at $tp_i$ with normal $n_i$, the corresponding observation cone (recall that it is a cone with apex located at $tp_i$ and oriented along $n_i$, with an apex angle of $\theta_b$) defines the set of possible viewpoint positions $V_{tp_i}$, from which $tp_i$ may be observed. Then for each valid viewpoint position, the set of valid orientations, $V_{ori}$, are limited by the scanner field of view.

Note that the following calculations are for one viewpoint position sphere, $VPS$, and will be repeated for all $VPS_i, i = 1 \ldots m$.

First, a base planar surface patch is defined at the origin of the viewpoint space as $\{z = 0, -\epsilon < x < \epsilon, -\epsilon < y < \epsilon\}$, along with the corresponding base observation cone with apex located at the origin and oriented along the $z$-axis. As shown in Figure 3a, the intersection of this cone with $VPS$ defines a spherical cap, every point on which is a possible viewpoint position for the base surface patch. The silhouette of this spherical cap is a circle, and is stored as a list of points, $V_{sil}$. The set of possible viewpoint positions for any target patch ($tp_i, n_i$) is given by the intersection of its observation cone and $VPS$, again a spherical cap, as shown in
Figure 3b. The silhouette of this cap, $V_{sil}$, is easily calculated by applying a transformation to $V_{bg}$; translation by $t_p$, and rotation by the Euler rotation matrix for the $n$ vector from the z axis. To calculate the points inside the cap, all $v_{3ad} = (r, \theta, \phi) \in V_{3ad}$ (r being the radius of $VPS$) are first sorted according to their $\phi$ angle. For every $\phi$, the corresponding $\theta$ interval is discretized, thus creating a list of viewpoint positions $V_{pi}$ to scan target patch $tp_i$.

These potential viewpoint positions are now checked for occlusion, and only the non-occluded ones are kept. Note that this leads to significant computational efficiency, since as the occluded viewpoints are discarded, all associated view directions are also automatically discarded and need not be calculated.

For each non-occluded candidate viewpoint position $v_{pi} = (r_{pi}, \theta_{pi}, \phi_{pi})$ associated with the target patch $tp_i$, the ranging ray $r_{vi}$ is the line connecting $v_{pi}$ to $tp_i$. All viewpoints positioned at $v_{pi}$, oriented such that $r_{vi}$ lies inside the scanner FOV cone, are a viewpoint candidate for scanning $tp_i$. As is shown in Figure 3c, the set of valid orientations for each viewpoint position $v_{pi}$ is a cone with the apex located at $v_{pi}$, axis along $r_{vi}$ and apex angle of $\theta_{FOV}$, and is called the orientation cone.

To determine this set efficiently, we define a base gaze cone with apex angle of $\theta_{FOV}$, centered at the origin (similar to the observation cone), and the points on the silhouette of the corresponding spherical cap are stored as $V_{bgSil}$. The set of valid orientations is then easily determined by applying a transformation by rotating and then translating by $v_{pi}$ to $V_{bgSil}$. The rotation matrix is the one that rotates the z-axis to $r_{vi}$. In this manner the set of viewpoints which can scan $tp_i$ is determined.

After the set of candidate viewpoints for every target point is identified, the viewpoints are sorted based on the number of visible target points and stored in a list. The viewpoint which scans the maximum number of target points is chosen as the next viewpoint for modeling.

The overall algorithm for the modeling NBV algorithm at each iteration is summarized in Algorithm 2. A key aspect of our algorithm is that by projecting the observation cone for each specified target point on the viewpoint space, a large number of invalid viewpoints are automatically excluded from further analysis, hence we avoid ray-tracing for every viewpoint. This leads to significant computational efficiency.

4. Path planning: moving the sensor to the desired viewpoint

4.1. SBIC-PRM: sensor-based path planning for the 6DOF manipulator

Having obtained the desired modeling viewpoint, inverse kinematics can be used to obtain the corresponding viewpoint configurations (see the overall algorithm in Section 4.3). Here we assume that the viewpoint configuration has been computed and explain the path planning technique used. We have chosen SBIC-PRM as the path planning technique for moving the 6DOF manipulator to the modeling configurations. A brief description of SBIC-PRM is given here; for further details, please see Yu and Gupta (2004).

The planner places a set of random samples in the configuration space (C-space) of the manipulator, and determines their collision status as one of free/in-collision/unknown within the workspace octree, $W$. The status of a sample configuration is: (i) free if the robot body does not intersect any obstacle or unknown nodes of the workspace octree; (ii) in-collision if the robot body intersects an obstacle node of the workspace octree; and (iii) unknown if the robot body intersects any unknown node but no obstacle node of the workspace octree. The free samples are added as nodes to the roadmap, the status unknown samples are added to a list $L_u$, and the in-collision samples are discarded. Whenever a new scan is taken, the workspace octree model is updated, and the C-space roadmap is also updated by checking the collision status of all the samples in list $L_u$, the obstacle ones are removed from the list, and the free ones are made nodes in the roadmap and checked for connectivity with their neighboring nodes (using a discretized straight line local planner in C-space), and in this way the roadmap is incrementally expanded. A key computation is that of a collision check for the robot body at a given configuration with the workspace octree, which is a fairly standard geometric computation (Yu and Gupta 1998).
moves along this path and takes the exploration scan at \( q_{ve} \). This leads to another view configuration that yields the maximal information gain. Thus, the exploration view planner gives priority to those areas that increase information about the modeling view configurations, taking into account the robot’s physical size and shape, thereby facilitating reachability for further views.

More specifically, the list of unknown modeling view configurations, \( Q_{vm} \), is passed to the exploration view planner. The exploration view planner selects the node (among all nodes in the reachable connected component of the roadmap) that yields the maximal information about \( Q_{vm} \) as the view configuration \( q_{ve} \) (see algorithm 3), and passes it on to the SBIC-PRM, which determines a collision-free path to it. The arm then moves along this path to reach \( q_{ve} \), and scans the unknown regions. Having taken (embedded) and a different type of NBV problem, i.e. NBV for exploration, and has been addressed mostly in the context of mobile robots, with criteria such as the Frontier Method (Yamauchi 1997; Freda and Oriolo 2005). A few works, including our own, have addressed it in the context of sensor-based planning for articulated arms.

Unlike these previous works, which used workspace-based criteria such as Maximize Physical space volume (MPV) (Kruse et al. 1996; Renton et al. 1999), our own work proposed a configuration space-based criterion, the Maximal Entropy Reduction (MER) criterion. It applies to the general class of robot-sensor systems for sensor-based planning and exploration (Yu and Gupta 2004; Wang and Gupta 2006), and formally poses the problem of view planning for exploration as that of maximizing the information (hence minimizing the entropy) gained about the status of C-space. The information is the collision status of the unknown configurations. Using probabilistic models of obstacle distribution, the concept of C-space entropy is then formally defined and the next best view for exploration is the one that yields Maximal expected C-space Entropy Reduction (MER).

Given a viewpoint, in Torabi et al. (2007) we derived an expression for expected C-space entropy reduction with the occupancy grid representation of workspace. It assumes a certain a priori probability assigned to unknown grid-cells in the workspace and incorporates occlusion constraints. We omit the details of this computation; it suffices to say here that the key underlying geometric computation is that of determining the expected volume of the visible unknown region (or equivalently, the number of unknown nodes in the workspace octree) occupied by the robot at a given configuration, and is easily computed. In general, the unknown robot configurations are selected from a random set of configurations in C-space, since the goal is to explore the whole C-space. However, here in the modeling task, the unknown configurations are the set of unknown modeling view configurations output by the modeling NBV planner.

An efficient way to determine the best view configuration for exploration is to view each node in the roadmap as a potential view configuration. We can easily determine the expected information gain (or entropy reduction) for the set of modeling view configurations for each node, and choose the one that yields the maximum information gain. Thus, the exploration view planner gives priority to those areas that increase information about the modeling view configurations, taking into account the robot’s physical size and shape, thereby facilitating reachability for further views.

4.2. Exploration NBV planner: configuration space-based exploration with occupancy grids

Since the workspace is initially unknown, the collision status of the best modeling view configuration(s) may not always be known, i.e. the view configuration may lie (partly or wholly) in the currently unknown workspace. Clearly the workspace should be explored (i.e. sensed) such that the status of the unknown workspace region occupied by these configurations is made known. This leads to another
the scan, the workspace octree and the C-space roadmap are both updated.

**Algorithm 3**: Exploration NBV planner

```plaintext
for every configuration on the roadmap q do
    I_q(Q_{um}) = 0;
    for every modeling configuration q_{vm} ∈ Q_{um} do
        determine Information Gain I_q(q_{vm});
        I_q(Q_{um}) = I_q(Q_{um}) + I_q(q_{vm});
    end
end
q_{vc} = \arg \max_{q∈roadmap} \{I_q(Q_{u})\};
```

4.3. Overall algorithm for 3D modeling by a 6DOF manipulator

The control flow of our overall algorithm for autonomous object modeling is shown in Figure 5. The robot starts in the initial configuration with an assumed small initial free area around it, with the workspace octree corresponding initialized. It assumes that the object to be scanned is located within a given bounding box, and that the very first scan in the initial configuration will see at least some part of the object. The new scan (range image) is used to update the octree model of the workspace, and the object point cloud model is initialized with the points (extracted from the range image) that lie inside the bounding box.

The modeling NBV is then invoked to determine a sorted list (ranked by the number of visible target patches) of viewpoints for modeling and determines the best viewpoint, v_m, which is passed to the path planner SBIC-PRM only if the number of target points visible from it is greater than a percentage of the total number of target points (15% in our experiments). SBIC-PRM invokes inverse kinematics to determine a finite set of manipulator configurations (modeling view configurations) Q_{vm} = \{q_{vm}^i, i = 1 \ldots n\} (n = 8 for our robot) corresponding to v_m. Each q_{vm}^i is checked for collision with the workspace octree (with unknown and obstacle nodes in the octree). If its status is unknown, it is saved in a list Q_{um}, and the planner goes to the next one. The first one found collision-free is deemed the goal configuration q_{vm}. SBIC-PRM then searches the roadmap for a collision-free path to q_{vm}, and if it finds one, q_{vm} is added to roadmap, and the robot uses this path to move to the desired configuration and proceeds with a new scan there. In this case, the octree model of the workspace is updated, and the new points within the bounding box in the new range image are registered with the current object point cloud model using ICP (Besl and McKay 1992; Rusinkiewicz and Levoy 2001) and then integrated into the object model. The list of target points is also updated by deleting the ones that no longer belong to the unknown area. If none of the q_{vm}^i’s are collision-free, SBIC-PRM gets the next viewpoint from the sorted list from the modeling NBV planner, and the process repeats, until there is no good modeling viewpoint in the list. At this point, the view planning switches from modeling phase to exploration phase, and SBIC-PRM calls the MER-based exploration view planner to explore free-space so that it can make the status of configurations in Q_{um} known. The exploration view planner uses the MER criterion to determine a reachable view configuration (a roadmap node) q_{ve} that gives maximum information about Q_{um}. The robot then moves to this configuration, proceeds with a scan, and the process repeats. If the exploration view planner is not able to find a view configuration that yields expected information gain greater than a threshold, the system deems that no further progress can be made toward modeling and exits with the current partial object model.

If at any iteration the set of the target points is empty, the object model is complete (see next section for proof), and the modeling process is terminated. The overall flowchart for our autonomous 3D modeling system is shown in Figure 5.

5. Termination criterion for model completion

A complete autonomous 3D object modeling system requires a self-termination criterion; i.e. the modeling algorithm should autonomously recognize when the goal has been achieved and the object model is complete, or if no more progress can be made.
5.1. Background

In previous autonomous 3D modeling works, various termination criteria were used, but none are related to explicit requirements to be met by the reconstructed object model (Scott et al. 2003). There are methods that lie between a fixed number of predefined viewpoints, and those with some criteria based on the number and/or status of viewpoints. Pito calculates the number of unknown patches visible from each viewpoint in the viewpoint space (Pito 1999). If this number for all viewpoints is less than a threshold value the process is terminated, or in other words he terminates when most of the remaining target points are not visible from the viewpoints (Pito 1999). Banta and et al. (2000) examined several termination criteria, e.g. terminating when the size of either the surface model or the occluded model ceases to change by a significant amount, or when the ratio of the size of the surface model to that of the occluded model is large. None of these methods is related to explicit requirements to be met by the reconstructed object model, and they mostly reason either over the visibility of the viewpoints or the progress rate. These methods can thus be somewhat ad hoc, and they are not robust to object shape completeness. Li calculates the volume encompassed by data from available views and analyzes the variation in the volume due to each successive viewpoint (Li et al. 2005; He and Li 2006). Since during the intermediate stages the scanned surface model is not complete and hence does not form a closed shape, it is difficult to directly calculate the volume encompassed by the surface data. Therefore he computes the volume of the data cloud via computing the surface integral of the triangular meshes and uses Stokes’ theorem to derive the volume enclosed (Li et al. 2005; He and Li 2006). The main problem is, Stokes’ theorem applies only for a closed surface and, at least the way it is applied in the paper, is questionable to us. Also, mesh computation is required for this termination technique, which could be time consuming.

5.2. Proposed termination criterion for model completion

As different scans are taken from a set of viewpoints, the seen parts of the object surface are modeled. Note that, in every scan, due to occlusion or limited FOV, parts of the object will not be visible, and this causes a boundary between the seen surface and the unknown space. Intuitively, one can imagine that if there is no such boundary between the object surface and the unknown space, the constructed model is complete.

Termination criterion (continuous case): If the boundary set (points in 2D, curves in 3D) is empty, the constructed model is complete.

To formally prove this termination criterion, we need a formal geometric definition of an object and a formal model for the sensing action. The formal concept of a closed surface captures our intuitive idea of a geometrical object. Along with various formal definitions, we formally show in Athat whenever the set of boundary curves is empty, the 3D model is complete.

Note that in practice the laser scanner doesn’t scan continuously, instead, the scan is discretized at a certain resolution, and the boundary curve is saved as a set of boundary points. Note that these are precisely the unknown target points in our NBV algorithm.

Termination criterion (discrete case): If the set of boundary points is empty, the object model is complete within the given resolution.

We show the experimental behavior of the number of boundary (unknown target) points versus scan iterations in Section 6.

6. Experimental results

6.1. Experimental setup

The system hardware consists of a time-of-flight Hokuyo URG-04LX (Hokuyo 2011) line-scan range scanner with maximum scanning range of about 3 m and angular resolution of 0.3 degrees. The range scanner is mounted on the last joint of a 6-DOF manipulator composed of powercube modules (Schunk 2011). The scanner is mounted so that the last joint of the robot moves in a direction perpendicular to the scan line. This allows us to obtain a 256 × 256 range image by moving the last joint and hence treat our scanner as a full range image scanner with a conical FOV of about π/4 radians view angle and 3 m range. We have considered a 512 cm³ cube as the workspace, W, and its octree model is constructed from scanned images as in Yu and Gupta (1998) with the finest resolution of 2 cm. In our system, the viewpoint position space consists of 4 concentric spheres with radius 30, 40, 50, 60 cm, centered at the center of the bounding box. These radii are essentially selected based on the maximum reach of the arm, the size of the bounding box, and the distance between the bounding box and the base of the manipulator so that the spheres cover the reachable region around the object bounding box in a uniform manner.

6.2. Acquired models

We carried out a set of experiments to illustrate the performance of our automated 3D modeling system. Note that we impose no constraints on the size and the shape of the object to be modeled, and the only information required for our system is a bounding box where the object is located. Also, the workspace is unknown, and practically the experiments could be done in any environment with static obstacles. For the first test, we chose a chair to be modeled, which is a fairly complex object which could not be modeled by just simply moving the scanner around it, because scans from hard-to-reach areas such as underneath the chair are needed to model it fully. Figures 6 and 7 show how our system is able to build the object model, switching between modeling scans and exploration scans as needed. The robot gets the first scan at the home configuration and scans parts of
Fig. 6. Continued.
Fig. 6. The six-DOF eye-in-hand system autonomously builds a model of a chair. The snapshots show the modeling viewpoints from which the successive scans were taken; the corresponding point cloud model of the chair is shown as well. While modeling a chair, the workspace also needs to be explored for reachability, and several exploration views are taken as well. The corresponding octree model is also built as exploration scans are taken and is shown next to the exploration scans. Please note that we are showing only obstacle voxels for ease of visualization. Note that with our system the scanner is able to scan from under the chair, which requires full six-DOF maneuverability.
Fig. 7. Continued.
Fig. 7. Continued.
The six-DOF eye-in-hand system autonomously builds a model of the table, watering can, and the box. The snapshots show the modeling viewpoints from which the successive scans were taken; the corresponding point cloud model is shown as well. While modeling, the workspace also needs to be explored for reachability, and several exploration views are taken as well. The corresponding octree model is also built as exploration scans are taken and is shown next to the exploration scans. Please note that we are showing only obstacle voxels for ease of visualization. The powerful collision-free motion planning is starkly illustrated, particularly in this frame, where the robot gets under the table to take a scan.

The object point cloud corresponding to this scan is shown in Figure 6b. The system then plans for next best view for modeling; however, since most of the workspace is currently unknown, these configurations are not reachable, and the system therefore switches to exploration phase and, as is shown in Figure 6c, the robot moves to an exploring view configuration. One can also tell that the scanner is quite far from the chair and that the robot is scanning the workspace to gain some information about the workspace and subsequently about the status of desired modeling configurations. Figure 6d shows the updated octree model of the workspace after the scan is taken and integrated in the workspace octree (only obstacles are shown to avoid clutter). Some modeling view configurations become reachable, and the robot moves to the best modeling configuration to capture the front part, and to complete the seat area (Figures 6e, 6f). This is particularly notable for scan 7 in which the robot scans the area near the lower parts of the legs of the chair. Having explored this region, the planner then is able to reach a view configuration (scan 8) that exactly lies in this region! Further scans are taken to build the model of the chair, Figure 6 showing the snapshots of the scan, along with the evolving point cloud model of the chair, as more and more scans are taken. Because of the overlap constraint for registration, each modeling scan should cover a portion of the already seen area of the object. Thus one can see that the 3D model is constructed gradually. Also, since the environment is unknown, the manipulator can’t maneuver aggressively around the object for taking scans and, as is shown in Figure 6, it must scan to explore the unknown workspace (exploration phase) so that it can then scan the object (modeling phase). The octree model of the workspace is also gradually completed in the exploration phase.

We also tested our system on a more complicated object, a table with a watering can on top and a box underneath. Figure 7 shows how the robot builds the model through several iterations. As one may have noticed, in this experiment a rather different scenario happens, i.e. after the first scan, which always is a modeling scan, our system continues with several exploration scans, and when enough information is gained over the workspace area the system switches back to modeling. As is illustrated in Figure 7, the robot reaches for different modeling viewpoints to capture the watering can, all around table, and even the box underneath the table, as far as its length can reach. The attached video of this experiment illustrates how our autonomous modeling system plans a spot-on modeling configuration, performs
collision-free motion planning, explores the workspace, and seamlessly integrates view planning for modeling, exploration and collision-free motion planning to build a complete model of the object of interest. Again, particularly note near the end of the video where the robot gets well under the table to take a scan. This modeling scan No. 14 is also shown in Figure 7q. The rendered video of this experiment is available in Extension 1.

In order to evaluate our termination criterion, we tracked the number of target points versus scan iterations in a simple experiment, as shown in Figure 8. We used a small object to ensure that the arm is able to reach around it. In the first scan, only a small part of the object is visible, hence the number of target points is relatively small. For the next few scans the number of target points increases. This is because, although some target points become visible and are removed, many more new ones are added. Therefore the plot has several peaks and valleys, but as the model gets more complete the number of target points converges to zero (a small number in practice).

Our overall planner was implemented on a 64-bit Dell Studio XPS 8100 (processor: i5 760 @ 2.8 GHz, 8 GB RAM). The overall run-time is about 4 minutes, and comprises run-times of a set of sub-modules: Modeling (Scanning/registration), View planning for Modeling, View planning for Exploration and Path planning (including updating octree and Roadmap). As one can see in Table 1, our NBV (modeling phase) planner takes only 2% of the total time in each iteration, and path planning takes a significant part (more than 50%) of the total run time.

Note that the efficiency of updating octree and Roadmap, and registration, could be improved. For example, Roadmap update times could be significantly improved via more efficient techniques for collision checking; similarly, registration could be significantly faster with fewer sample points, fewer iterations, and more overlap. The octree model could be double resolution, a finer one for modeling the object, and a coarser one for modeling the environment, improving both memory and runtime efficiency.

7. Conclusion and future works

We have implemented a fully autonomous 3D object modeling system with a six-DOF eye-in-hand system. The system is able to build a complete 3D model of an unknown object in situ while avoiding collisions in an unknown workspace. Our overall planner integrates three main algorithms in the system: a novel next best view (NBV) planner for modeling the object, an NBV planner for exploration, i.e. exploring the environment, so the arm can move to the modeling view pose, and finally a sensor-based path planner, that is able to find a collision-free path to the view configuration determined by either of the the two view planners. Our modeling NBV algorithm efficiently searches the five-dimensional point space globally, and furthermore do not consider the path planning aspect at all.

In this work, it is assumed that the bounding box encloses completely and only the object(s) to be modeled. If this assumption is not satisfied, either the bounding box does not enclose the entire object, or additional objects are located inside the bounding box. In the former case, only the portion of the object inside the bounding box is modeled, and in the latter case, all objects enclosed by the bounding box will be modeled, and further image processing (e.g. segmentation) would be needed to derive the model of the desired object.

A fixed-base manipulator, although it affords great maneuverability to the sensor, nevertheless has limited reachability, e.g. it is unable to reach across a relatively large object such as the chair in our test example. Hence, we are currently extending our autonomous 3D modeling system and the corresponding planning framework to a mobile manipulator with nine degrees of freedom, three for the mobile base and six for the arm mounted on it.

Notes

1. While some works have considered a six-DOF positioning mechanism (Chen and Li 2005), these do not search the viewpoint space globally, and furthermore do not consider the path planning aspect at all.
2. The breakdown angle depends on the surface properties and the laser scanner. For more details of our laser scanner (Hokuyo URG-04LX), refer to Okubo et al. (2009).
3. A ranging ray emanates from the scanner, and an observation ray emanates from the surface.
4. If this were a redundant manipulator, one could directly plan a path for the desired pose without explicit inverse kinematics (Yao and Gupta 2005).
5. The integration could be done as volume intersection (Curless and Levoy 1996; Rocchini et al. 2004), or mesh integration (Turk and Levoy 1994; Zhou et al. 2006).
6. Jordan curve: A plane curve which is topologically equivalent to (a homeomorphic image of) the unit circle, i.e. it is simple and closed.

Funding
This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

Conflict of interest
None declared.

References
Appendix

A. Proof for termination criterion

We now present a formal proof for the termination criterion (continuous case) stated in Section 5. For simplicity and ease of understanding, we will first show it for the 2D case, and then for the 3D case. In both scenarios, we first give formal definitions for objects, and our model of sensing action is continuous and is detailed further in each case.

A.1. Some definitions

An \( n \)-manifold (without boundary) will mean a topological manifold such that every point has a neighborhood homeomorphic to \( \mathbb{R}^n \). Let \( M \) be a manifold with boundary. Then the interior of \( M \), denoted \( \text{Int} (M) \), is the set of points in \( M \) which have neighborhoods homeomorphic to an open subset of \( \mathbb{R}^n \), and the boundary of \( M \), denoted \( \partial M \), is a manifold of \( \mathbb{R}^{n-1} \).

Intuitively we know that in the plane a closed curve is a curve with no boundary points and which completely encloses an area, and a simple curve is a curve that does not cross itself. In the following we present a more mathematical definition. Let \( I = [a, b] \subset \mathbb{R} \) be a closed interval of a real line. A curve (also called a 1-manifold) is a continuous mapping \( \gamma : I \to X \) taking values in topological space \( X \). We say that \( \gamma \) is a closed curve whenever \( \gamma (a) = \gamma (b) \), otherwise \( \gamma \) is an open curve and \( a, b \) are the endpoints. Also, \( \gamma \) is a simple curve if the mapping \( \gamma \) is injective.

Let \( A \subset \mathbb{R}^n \) be any subset, let \( p \in A \) be a point, and let \( r > 0 \) be a number. The open ball in \( A \) of radius \( r \) centered at \( p \) is the set \( O_r (p, A) \) defined by \( O_r (p, A) = \{ x \in A : \| x - p \| < r \} \).

A.2. Termination criterion for modeling a 2D object

We assume that our 2D object, \( M \), is a simple and closed curve (Jordan Curve).\(^6\) Now consider an ideal laser scanner moving around the object, taking scans, and in the \( k \)th scan the border of the object, \( M \), is sensed as shown in Figure A1. It consists of a set of open curves \( \mathcal{C}^k \) such that \( \mathcal{C}^k \subset M \). See Figure A1 for an illustration. Note that although point \( \partial C_{22} \) is not scanned since it is occluded, every open neighborhood of \( \partial C_{22} \) contains points which are scanned. Hence, in our sensing model we assume that \( \partial C_{22} \) is also sensed. In special cases, an open curve may degenerate into a single point, in which case it is denoted by \( p^k \). We use the word segment to denote a single scanned (maximal) open curve.
(or point). Multiple segments in the same scan are caused by a limited field of view, occlusion, or when the ranging ray is tangent to the object M. For each scanned segment, C, there will be two boundary points, \( \partial C_i, \partial C_j \). Let \( C = \cup C_i \), and let \( Bp \) be the set of all boundary points of \( C \) (isolated sensed points \( p \) are also included in this list). In each iteration, after the scan is performed, \( C \) is updated with the set of new scanned segments, and correspondingly the set of new boundary points and possible single points are added to \( Bp \). A merge/glue process is used so that \( Bp \) contains only those points which have not been sensed in any of the scans, i.e., those points that are sensed in other scans are eliminated from \( Bp \). Formally, \( p \notin Bp \) if \( \exists r > 0, O_r(p, R) \subset C \).

**Theorem 1.** The set of endpoints \( Bp \) is empty at any iteration \( k > 1 \) if and only if the object model is complete, or, formally speaking, \( Bp = \emptyset \) if and only if \( C = M \).

**Proof.**

**Part 1:** If \( Bp = \emptyset \) then \( C = M \):

The proof will be in two steps: a) if \( Bp = \emptyset \) then \( C \) is a closed and simple curve; b) if \( C \) is a closed and simple curve, then \( C = M \).

a) Assume \( Bp = \emptyset \). If \( C \) is not closed, then \( C \) will at least have one boundary point \( p \) such that \( p \in \partial C \), therefore \( \exists r > 0, O_r(p, R) \subset C \). But since \( C \) is continuous, there exists a \( C_j \subset C \) such that \( p \in \partial C_j \). Since \( \exists r > 0, O_r(p, R) \subset C \), and \( p \in \partial C_j \), by the definition of \( Bp \), \( p \in Bp \). But, \( Bp = \emptyset \), thus it leads to a contradiction, hence \( C \) is closed.

b) \( C \subset M \), and \( M \) is simple, thus \( C \) is simple. As we know \( M \) is a simple closed curve, i.e., only includes one closed curve, and \( C \) is closed and \( C \subset M \), hence \( M - C \) is vacuous as desired, and \( M = C \) (Ayres 1929).

**Part 2:** If \( C = M \) then \( Bp = \emptyset \):

If \( Bp \neq \emptyset \) then \( \exists p \) such that \( p \in Bp \), and \( p \) is a boundary point for a \( C_j \) and \( \exists r > 0, O_r(p, R) \subset C \). Since \( p \in C_j \), then \( p \in C \), and this means \( C \) is not closed. Now because \( M \) is a closed curve and \( M = C \), \( C \) is closed and \( C \subset M \), hence \( C = M \)

Thus, in the modeling process whenever the set of endpoints is empty, the model is complete. Note this proof could be extended to a set of non-intersecting closed curves. In such scenarios, if at any iteration the set of boundary points is empty, every object/curve in the scene which has been scanned at least once will have a completed model.

**A.3. Termination criterion for modeling a 3D object**

A surface is a 2-manifold, a closed surface is a 2-manifold (without boundary), an open surface is a 2-manifold with boundary (open manifold), and the boundaries are sets of closed curves (1-manifold).

We assume that the 3D object, \( M \), is a closed and simple 2-manifold. As the laser scanner moves around the object, and takes scans of the surface of the object, \( M \), it generates a set of open 2-manifolds. More precisely, at the 4th iteration, a set of open 2-manifolds \( S^k \), such that \( S^k \subset M \), and/or a set of disjoint open curves \( C \) (in special degenerate cases due to alignment with respect to view direction), are generated. Each \( S_j \in S^k \) is bounded by a close curve, \( \partial S_j \). Also at the 4th iteration the union of the set of scanned areas is denoted by \( S \), i.e., \( S = \bigcup_{j=1}^{k} S_j \).

In each iteration, after the scan is performed, \( S \) is updated with the set of new scanned surfaces, and correspondingly the set of new boundary curves and sensed open curves (if any) are added to \( Bc \). A merge/glue process is used so that \( Bc \) contains only those curves which have not been sensed in any of the scans, i.e., portions of curves that are sensed in other scans are eliminated from \( Bc \).

**Theorem 2.** The set of boundary curves \( Bc \) is empty at any iteration \( k > 1 \) if and only if the 3D object model is complete, or, formally speaking, \( Bc = \emptyset \) if and only if \( S = M \).

**Proof.**

**Part 1:** If \( Bc = \emptyset \) then \( S = M \).

The proof will be in two steps: a) if \( Bc = \emptyset \) then \( S \) is a closed surface; b) if \( S \) is a closed surface, then \( S = M \).

a) If \( S \) is not a closed surface, then \( \exists \) at least one closed curve \( C \in \partial S \). Since the \( S \) are continuous, except at the boundary curves, \( C \subset (\bigcup \partial S) \). By definition, for every point \( p \in C \) there is no open neighborhood in \( S \) homomorphic to \( R^2 \), i.e., \( \exists r > 0, O_r(p, R^2) \subset S, \forall p \in C \). However, by definition of \( Bc \), \( C \) must belong to \( Bc \). Since \( Bc = \emptyset \), it is a contradiction. Hence \( S \) must be closed.

b) \( S \) is closed and we know that \( S \subset M \), a simple closed surface, \( S = M \).

**Part 2:** If \( S = M \) then \( Bc = \emptyset \).

If \( Bc \neq \emptyset \) then \( \exists \) a curve \( C \subset Bc \). By definition of \( Bc \), \( C \subset \partial S \). But \( S = M \) and thus \( S \) is a closed manifold and \( \partial S = \emptyset \). This leads to a contradiction, hence \( Bc = \emptyset \). □

**B. Index to Multimedia Extensions**

The multimedia extension page is found at http://www.ijrr.org.

**Table of Multimedia Extensions**

<table>
<thead>
<tr>
<th>Extension</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Video</td>
<td>Integrated object modeling and motion planning</td>
</tr>
</tbody>
</table>