The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Integrated View and Path Planning for an Autonomous six-DOF Eye-in-hand Object Modeling System

Liila Torabi, 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 sensor mounted on a six-DOF manipulator arm and the task is to autonomously build 3D model of an object in-situ, i.e., the object may not be moved and must be scanned in its original location. Our system assumes no knowledge of either the object or the rest of the workspace of the robot. The overall planner integrates a next best view (NBV) algorithm along with a sensor-based roadmap planner. Our NBV algorithm while considering the key constraints such as FOV, viewing angle, overlap and occlusion, efficiently searches the five-dimensional view space to determine the best modeling view configuration. The sensor-based roadmap planner determines a collision-free path, to move the manipulator so that the wrist mounted scanner is at the view configuration. If the desired view configurations are not collision free, or there is no free path to reach them, the planner explores the workspace such that facilitates the modeling. This is repeated until the entire object is scanned. We have implemented the system and our results show that system is able to autonomously build a 3D model of an object in an unknown environment.**

I. 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, comprises three broad modules, as shown in Figure 1: i) 3D modeling, which concerns itself with the scanning and merging the scans to build the object model, and essentially deals with machine vision/graphics aspects; ii) view planning, i.e., to determine the sensor pose from where to scan the object, which has both vision and robotics aspects, and (iii) moving the sensor positioning mechanism (a robot or a turn table) to desired scanning poses, which brings in the path planning aspect.

The 3D model building cycle, shown in the left block in Figure 1, consists of three main phases: a) scan, b) register, and c) integrate. A scan is taken from a camera pose, acquired range images are registered using standard techniques such as Iterative Closest Point (ICP) method [1], [2]. Finally, registered images are combined into a single object model, a process commonly known as integration. The integration could be done as volume intersection[3], [4], mesh integration[5], [6], [7] or at the point cloud level. 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 hand held scanner, such as Polhemus

Lila Torabi, and Kamal Gupta are with Robotic Algorithms & Motion Planning (RAMP) Lab, School of Engineering Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada. Email: `ltorabi@sfu.ca`, `kamal@cs.sfu.ca`

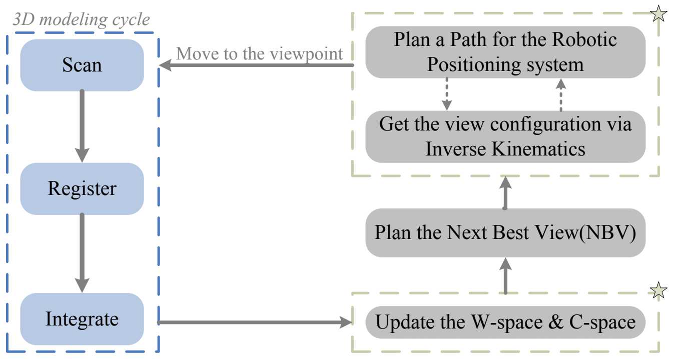

3D modeling cycle
Scan
Register
Integrate
Move to the viewpoint
Plan a Path for the Robotic Positioning system
Get the view configuration via Inverse Kinematics
Plan the Next Best View(NBV)
Update the W-space & C-space

Fig. 1. Components of a fully autonomous 3D object acquisition system (* represents path planning aspects)

FastSCAN [8], 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 is not autonomous.

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 the right blocks in Figure 1. The next best view (NBV) problem is to determine the best scanner pose to scan the object from [9]. The primary purpose of an NBV algorithm is to ensure that all scannable surfaces of an object will be scanned. The preference is to minimize the number of scans needed to build the model, since scanning, processing a scan, and integrating it into one object model is often quite time consuming. Most works in computer vision plan the next best view normally based on large and fixed set-ups that possess limited mobility, e.g., a turn table [1]. The viewpoint space is mostly limited to one or two degrees of freedom (DOFs), and at best provides limited coverage models for a restricted class of objects, also they all assume that the desired viewpoint is reachable. In order to acquire a complete model of an object, in general a six-DOF viewpoint space is requiered, so that the object can be viewed from an arbitrary viewpoint. 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 indeed such a goal is achievable with efficient NBV algorithms coupled with the high computing power of current standard PCs. The reachability of NBV, however, obviously depends on the positioning mechanism on which

[1]While some works have considered a six-DOF positioning mechanism [10], these do not search the viewpoint space globally, and furthermore do not consider the path planning aspect at all.

the scanner is mounted. Hence at least six-DOF positioning mechanism, e.g., a manipulator arm, is requiered. However, using a six-DOF manipulator as a sensor positioning device necessitates the use of path planning to move the robot while avoiding collisions with obstacles, which are also sensed by the scanner. While such coupling has been explored in site modeling works [11], [12], the positioning mechanism has been a mobile robot, and 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 environment [13], [14], [15], but these works didn't consider object modeling and the associated constraints such as overlap.

In this paper, we present an integrated view planning and motion planning for a six-DOF eye-in-hand object modeling system. Two key contributions of our work are (i) an NBV algorithm that efficiently searches the five dimensional (five because of symmetry of sensor field of view), and (ii) integrating the NBV with a sensor-based motion planner (SBIC-PRM [14]). Note that the interaction between the NBV module and the path planning module is non-trivial. For instance, the desired view configuration from NBV module may not be collision free, or it might be collision-free but not reachable within the currently known workspace. In both cases, the path planner needs to explore the workspace such that the desired NBV could be reached (details in Section 4). The resulting system is a fully autonomous system capable of modeling in-situ objects without any human intervention. We show that the system is able to autonomously scan an object in environments while avoiding collisions with unknown obstacles. To 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.

II. The NBV Problem for object modeling: background and related work

The view planning problem has been well studied and this section is a brief overview of the problem, summarized from existing literature, e.g., [16]. The NBV problem involves reasoning in three spaces, object surface S, viewpoint space (the set of possible viewpoints) V, and workspace W, the 3D region in which the positioning device moves and the camera takes scans.

The NBV is essentially composed of the following two main sub-problems: determining the target areas that need to be scanned and determining how to position the range scanner to scan them [17]. For the first sub-problem, a common approach is to use range discontinuities in the current range image, to identify unseen portions of the object, which indicate target surface areas, thus providing cues for viewing direction. The target area is represented by either a volumetric model (octree or voxel) [18], [19], [20], [21], [22] or a surface [23], [24], [10], [25], [17], [26].

Once the target areas to be scanned have been determined, the viewpoint space is searched to determine which target areas are visible from each viewpoint in V, given the sensor model (often called the sensor constraint). Additional set of constraints such as overlap between successive views(for registration), occlusion(visibility constraint) from other objects (or parts of the same object for non-convex surfaces) also need to be taken into account. Ray tracing is a key computation in determining the set of visible target areas for each viewpoint. However, a naive brute force application of ray tracing for all view points would be computationally expensive. Hence a key challenge, particularly for high dimensional viewpoint space, as in our case, is to keep the computation fast and efficient to retain the on-line nature of the system. Consequently, NBV algorithms can be classified based on degree of freedom of the viewpoint space V, whether V is searched locally or globally, and by the constraints they take into consideration.

In some of the earliest papers, Connolly [22] and Banta [21] used Octree/voxels representation, and labeled the cells as empty, occupied, or unseen. The viewpoint space V was the surface of a sphere with view direction normal to the sphere (2-DOF). The visible unseen volume for each viewpoint was determined via ray tracing, and the viewpoint which scans the largest unseen volume, was determined through a global search. Massios [20] added the viewing angle as a quality factor. Maver [26] introduced a target driven method, where the view angles (his V was 1-D) for target regions was determined, and the viewpoint (view angle) with the maximum number of targets, selected as the NBV. Pito [9], [17] chose rectangular patches attached to occluding edges of the seen surface (mesh representation) as a cuing mechanism for a global search of V. He used an intermediate 'positional space' (PS), to represent what must be scanned (via the use of '*observation rays* ' and what can be scanned (via use of *ranging rays*') in a single data structure. He used a turn table (1-DOF viewpoint space) as a positioning system. Clearly direct extension his method to a high-DOF viewpoint space would be computationally expensive both in memory usage and processing time.

In our NBV algorithm we take advantage of the conical sensor FOV, and directly project the observation rays to the V. This direct projection allows us to determine the set of viewpoints (which can potentially see a target point) without ray tracing for every viewpoint, leading to significant computational efficiency, as shown by our implementation. Like [17], we consider viewing angle, overlap, and sensor FOV constraints. We also exploit the structure of V, which is $R^3 \times SO(3)$ in our case. As a result, if a position is occluded, entire orientation sub-space can be eliminated without any occlusion checks (or ray tracing). Furthermore, note that in all works mentioned above no path planning is involved.

Several other works search the viewpoint space locally [24], [25], as a result the number of required scans and the time taken could be quite large. Chen [10] predicts the curvature of the unknown area of the object, to drive the viewing direction. Although he uses a six-DOF viewpoint space, since it's a local algorithm, there is no search involved. Orfanos presented a NBV for a very short range scanner mounted on the robotic arm [19], but his algorithm was mostly concentrated on collision avoidance with the object.

III. Our NBV Algorithm

A. Representation of spaces W and S

We represent W as an Octree, primarily because it facilitates occlusion checks in NBV, and collision checks in path planning. In the Octree model used in this work, each cell is categorized as free, obstacle or unknown [27].

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 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.

B. Laser Scanner Model, Viewpoint space, and Target points

The range image obtained in each scan consists of set of ranging rays within the field of view (FOV) of the scanner. We approximate the sensor field of view as a cone (with apex angle θ_{FOV}), hence the rotation around cone axis does not change the view, and viewpoint space V is therefore five dimensional.

V is discretized as follows: Concentric spherical surfaces (we use four) are used to represent the viewpoint position $v_p = (r_p, \theta_p, \phi_p)$, with each spherical surface (or, viewing sphere) discretized to 32×32 patches. At each viewpoint position, the set of viewpoint orientations is represented by a 2-tuple $v_o = (\theta_o, \phi_o)$, discretized to a 32×32 resolution. As a result V has $32 \times 32 \times 32 \times 32 \times 4 = 4.19 \times 10^6$ distinct viewpoints [2]. Thus every viewpoint is represented as a five tuple $v_i = (r_p, \theta_p, \phi_p, \theta_o, \phi_o)$This representation for V allows us to map the observation rays to V, simply by defining a projection function. Potentially, other tessellation methods could be used for viewing sphere [28].

Each target point represents a small surface patch in the unknown region. The surface patch is scannable from directions that form an angle of the *breakdown angle* (θ_b) or less with the normal vector of that surface patch [3]. Each of these directions, or *observation rays* represent a potential *ranging ray*[4]. The set of all observation rays for each point is its *observation cone*.

C. What to scan

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 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.

[2]We chose this resolution somewhat empirically. There ia trade off; higher resolution results in more viewpoints, but increases the algorithm run-time.

[3]The breakdown angle depends on the surface properties and the laser scanner. For details for laser sensor (Hokuyo URG-04LX) refer to [29].

[4]A ranging ray emanates from the sensor, and an observation ray emanates from the surface.

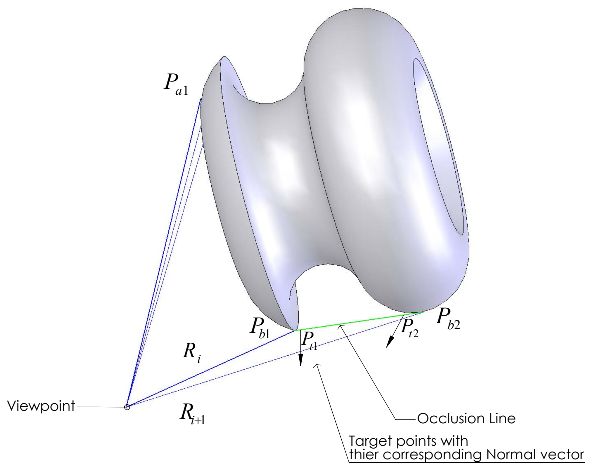


Fig. 2. The geometry of target points and viewing direction. See text for explanation.

In Figure2, we illustrate a discontinuity as the ray R_i hits the object at P_{b_1}, and a neighboring ray R_{i+1} hits the object at P_{b_2}. The line connecting P_{b_1} and P_{b_2} is an occlusion line. Any point on the occlusion line could be considered as a target point, but we only use points close to the scanned surface, P_{t_1}, and P_{t_2}, as shown in Figure 2. 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 object's surface, (ii) choosing target points close to seen surface facilitates estimation of surface normals, needed for choosing viewing directions (explained later), and (iii) it automatically satisfies overlap constraint, essential for registration.

After the list of target points are extracted, we need to estimate their corresponding surface normal to determine their valid viewing directions.

The points from the object point cloud which are in neighborhood radius of P_{t_2} form a local estimate for its surface patch. To do this efficiently, we use a Kdtree for the point cloud data and search it, to identify neighboring points for P_{t_2} [30]. The normal vector of the surface patch is calculated using these neighbor points based on a principal component analysis (PCA) method [31], [32].

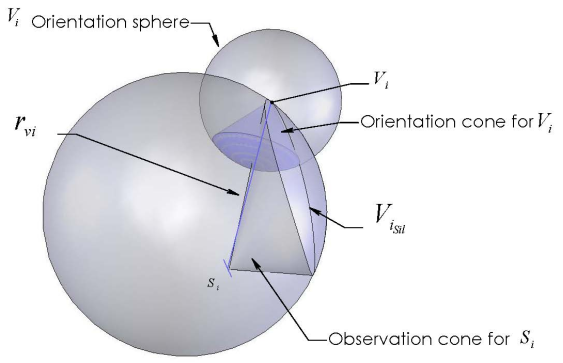


Fig. 3. The geometry of viewpoints that can see a surface patch s_i. See text for explanation.

The known surface near P_{t_1} is the occluding surface, hence it is either tangent to the surface patch in that area, or there is a sharp edge; either way the seen points in the point cloud are not a good estimate for viewing direction. Thus, as is shown in Figure 2, the viewing direction for P_{t_1} will be calculated in a similar technique as P_{t_2}, but the kdtree is built from union of the neighboring target points (unknown) and the (seen) points in the point cloud model.

At the end of this step we will have a list of target points, s_i and their corresponding normal direction n_i.

D. How to Scan

The geometry of viewpoints that can see a surface patch s_i is shown in Figure 3. We define a base planar surface patch at the origin of the viewpoint space as: $\{z = 0, -\epsilon < x < \epsilon, -\epsilon < y < \epsilon\}$ and its corresponding observation cone (apex angle of θ_b) with apex located at origin and oriented along z-axis. The intersection of this cone with each of the spheres in the viewpoint position space, defines a spherical cap. Every point on this cap is a possible viewpoint position for *base surface patch*. The silhouette of this spherical cap is a circle, and is stored as a list of points, $V_{b_{sil}}$.

A similar cone with apex angle of FOV (apex at origin) is defined as the gaze cone, and the points on the silhouette of the corresponding spherical cap are stored as $V_{g_{sil}}$.

For a surface patch centered at s_i with normal n_i, the set of possible viewpoint positions is calculated by applying a transformation matrix (corresponding to the rotation and translation that transforms the unit z vector at origin to unit vector at s_i) to $V_{b_{sil}}$. Figure 3 illustrates the viewing cone for s_i, and the silhouette of intersection of this cone with the viewpoint position sphere, $V_{i_{sil}}$. The $V_{i_{sil}}$ forms a circle, with every point on the associated spherical cap inside this circle, is a desired viewpoint position for surface patch s_i. To calculate the points inside this circle, all $v_{p_i} = (r_p, \theta_{p_i}, \phi_{p_i}) \in V_{i_{sil}}$are first sorted according to their ϕ angle. For every ϕ_{p_i}, the corresponding θ_p interval is discretized, thus creating a list of viewpoint positions v_{p_i} to scan surface s_i.

Before calculating the valid orientations for the set of potential viewpoint positions, these viewpoint positions are checked for occlusion, and only the non-occluded ones are kept. Note that this leads to significant computational efficiency, since if an occluded view position is discarded, all associated view directions are also discarded without doing any occlusion check for them. For each non-occluded candidate viewpoint position, v_{p_i}, associated with the surface patch s_i, the ranging ray r_{v_i} is the line connecting s_i and v_{p_i}. All viewpoints positioned at v_{p_i}, for which r_{v_i} lies inside the sensor FOV cone, are a viewpoint candidate for scanning s_i. As is shown in Figure 3 the set of valid orientations for each viewpoint position v_{p_i}, is a cone with the apex located at v_{p_i}, axis along r_{v_i} and apex angle of θ_{FOV}. This cone, called orientation cone is calculated by first translating the gaze cone by v_{p_i} and then by rotating it (corresponding to the rotation that transforms the unit z vector at origin to r_{v_i}).

Once the set of viewpoints that can scan s_i is determined, their number of visible targets is incremented by one. 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, L_{vp}. The overall scheme for the NBV algorithm at each iteration is summarized in Algorithm 1.

Algorithm 1: NBV planner

```
Input: last scanned range image
Output: the sorted list of the viewpoints for next scan
Extract the range discontinues in the scanned image;
Determine the target points for each discontinuity;
for every target point $P_{t_i}$ in the list do
    Estimate the surface patch $s_i$, and normal $n_i$ from its
    closest neighbors on kdtree;
    for each viewpoint position sphere do
        Compute the $V_{i_{sil}}$ of the viewing cone for $s_i$;
        Extract the set of candidate viewpoint positions;
        for every candidate viewpoint position, $v_{p_i}$ do
            Determine the ranging ray $\overline{s_i v_{p_i}}$;
            if $\overline{s_i v_{p_i}}$ is occlusion free then
                Compute the orientation cone;
                Extract set of valid orientations for $v_{p_i}$;
                for every valid orientation, $v_{o_i}$ do
                    Increment the number visible target
                    points for $v_i = (v_{p_i}, v_{o_i})$;
                end
            end
        end
    end
end
Sort viewpoints by the number of visible target points
```

IV. Path Planning: Moving the sensor to the desired view configuration

The output of the NBV algorithm is a sorted list of view poses. We use SBIC-PRM (Sensor Based Incremental Construction of Probabilistic Roadmap) planner to determine a collision-free path for the manipulator as follows.

The best viewpoint pose (top of the sorted list) called v_g is mapped to a finite set of manipulator configurations, $Q_g = \{q_{g_i}, i = 1, \dots m\}$ (m=8 in our case) via closed form inverse kinematics[5]. Each q_{g_i} is checked for collision with the workspace Octree (unknown and obstacle areas), and the first one found collision-free is deemed the goal configuration q_g, and passed to SBIC-PRM[6]. The roadmap is searched for a path to this configuration, and if a collision-free path is found, it is executed,. If none of the q_{g_i}'s are collision-free, or there is no collision-free path to the collision-free ones, the next view pose in the sorted list is selected, and the process repeats.

[5]If this were a redundant manipulator, one could directly plan a path for the desired pose without explicit inverse kinematics [33].

[6]We give a brief description of SBIC-PRM here, for details, please see [14]. The planner places a set of random samples in the configuration space (C-space) of the manipulator, and determines their collision status within the workspace Octree as one of free/in-collision/unknown. The free ones are added as a node to the roadmap, the status unknown ones are added to a list $\mathcal{L}$, and the in-collision ones are discarded.

Algorithm 2: Overall Planner for automated 3D object modeling with a 6-DOF eye-in-hand system.

```
Initialize Roadmap;
$q_c$ = initial robot configuration;
while Object model is not complete do
    Scan;
    Update workspace Octree;
    Update Roadmap;
    if iteration >1 then
        Update target points;
        Apply registration;
        Update the object 3D point cloud model;
    end
    Call the NBV planner (for modeling);
    repeat
        Get viewpoint $v_g$ from the top of the sorted list;
        $Q_g = \{q_{g_i}, i = 1, \dots m\}$ = InvKin($v_g$);
    until $q_{g_i}$ is collision-free;
    $q_g = q_{g_i}$;
    if $q_g$ is a valid solution then
        Add $q_g$ to the Roadmap;
    else
        $q_g$=Plan a view for exploration;
    end
    plan a path for $(q_c, q_g)$ through SBIC-PRM;
    Move manipulator to $q_g$;
    $q_c = q_g$;
end
```

If all view pose configurations have been exhausted, because none of them are reachable, the planner switches from modeling phase to exploring phase, i.e., the next view(s) is planned to explore free-space so that the manipulator can indeed manoeuver to get to the desired view pose. We have incorporated a slightly modified version of C-space entropy criterion based view planner [14], [34] in our overall planner. It tries to gain information about the view configurations that are unknown. Details are omitted here for lack of space

Whenever a new scan is taken, whether for modeling phase or for exploring phase, the workspace octree model is updated [27]. The C-space roadmap is also updated by checking collision status of all the samples in list L, 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, and in this way, the roadmap is incrementally expanded. The new range image is registered using ICP [1], [2] before updating the point cloud model. The list of target points is also updated by deleting the ones that have been visited.

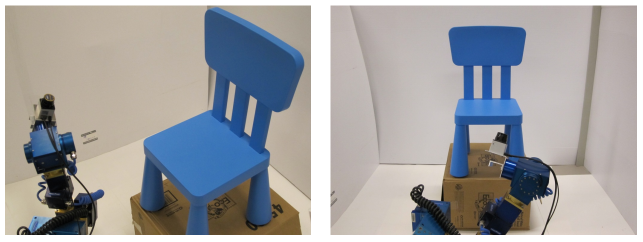
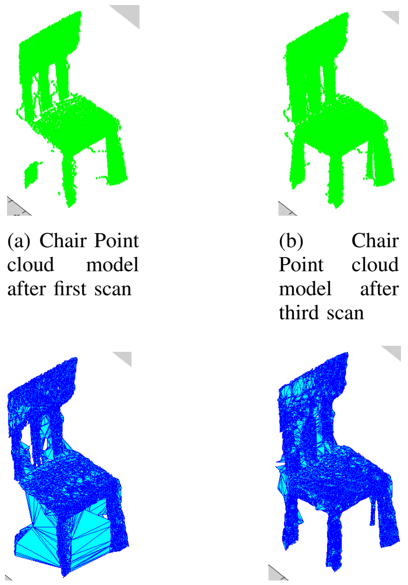

(a) First scan (b) Third scan

Fig. 4. Six-DOF Eye-in-hand system. Two snap shots show the viewpoints from which scans were taken. The corresponding integrated mesh models of the chair are shown in Figure 5. Note that with our system the scanner is able to scan from under the chair, which requires full 6-DOF manoeuvrability.


(a) Chair Point cloud model after first scan

(b) Chair Point cloud model after third scan

(c) Mesh model created after first scan

(d) Mesh model created after third scan

Fig. 5. Chair point cloud model and mesh triangulation.

V. Experimental Result

A. Experimental setup

The system hardware consists of a time-of-flight Hokuyo URG-04LX line-scan range sensor mounted on the last joint of a 6-DOF manipulator configured with powercube modules [35], and is shown in Figure 4. The maximum scanning range is about 3 meters. The sensor 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 area scan range image sensor with a conical FOV of about $\pi/4$ view angle and 3 meter range. We have considered a cube of side 256 cm as the imaging work space and its Octree model is constructed from scanned images as in [27] with minimum resolution of $1cm$.

B. Acquired Models

We impose no constrains on the size and the shape of the object to be modeled and also assume that the workspace is unknown. We have chosen a chair as the object to show how the manipulator (six-dof manoeuvrability) as it avoids the obstacles in the environment, takes a scan under the chair. Figure 4 shows our system is able to scan legs, go in between them and take a scan from underneath the chair. while other limited DOF systems in literature cant do this. Figure 5 shows the evolving point cloud model of the chair, as more and more scans are taken. Note that since the environment is unknown, the manipulator can't manoeuver aggressively around the object for taking scans, and must first scan to

TABLE I

RUN TIMES FOR AUTONOMOUS 3D OBJECT MODELING

Average Processing Time (seconds)		
Modeling cycle	Scanning	25
	Registration/Integration	115
View planning	NBV Algorithm	6
	Update targets	3
Path planning	Update Octree model	129
	Update Roadmap	109
	Path searching	2
Complete iteration		455

explore the unknown workspace (explore phase) so that it can then scan the object (modeling phase). In the modeling phase, because of overlap constraint for registration, each scan should cover a portion of the already seen area of the object. Thus, the 3D model is constructed gradually.

Our overall planner was implemented on a Pentium 4 with 3 GHz clock and the run-times for various sub-planners are shown in Table.I NBV (modeling phase) planner takes only 2% of the total time in each iteration. This time is based on the average number of 150 target points.

Note that the efficiency of the other modules, could be improved e.g., using more efficient techniques for collision checking will reduce the processing time for the roadmap update.

VI. 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. A key part of the overall planner is an view planning algorithm that efficiently searches a five-dimensional viewpoint space (4×10^6 viewpoints) to determine the best view pose for the scanner. This allows the system to construct an accurate model with a small number of scans. The view planner is integrated with a sensor-based roadmap planner (SBIC-PRM) to find a collision-free path to move the manipulator (hence the scanner) to the desired view pose. Initial experiments with the system show that the system is able to autonomously scan an object in environments while avoiding collisions with obstacles, while each iteration, including all motion planning techniques and workspace updates, takes about 7-8 minutes.

REFERENCES

[1] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Trans PAMI*, vol. 14, pp. 239–256, 1992.

[2] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," *Third International Conference on 3-D Digital Imaging and Modeling*, pp. 145–152, 2001.

[3] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *SIGGRAPH-96,*, pp. 303–312, 1996.

[4] C. Rocchini, P. Cignoni, F. Ganovelli, C. Montani, P. Pingi, and R. Scopigno, "The marching intersections algorithm for merging range images," *The Visual Computer*, vol. 20, no. 2, pp. 149–164, 2004.

[5] G. Turk and M. Levoy, "Zippered polygon meshes from range images," in *SIGGRAPH : Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pp. 311–318, 1994.

[6] A. Hilton and et al., "Reliable surface reconstruction from multiple range images," *Computer Vision*, pp. 117–126, 1996.

[7] H. Zhou, Y. H. Liu, and L. Z. Li, "Incremental mesh-based integration of registered range images: Robust to registration error and scanning noise," in *ACCV06*, pp. 958–968, 2006.

[8] "http://polhemus.com/."

[9] R. A. Pito, *Automated surface acquisition using range cameras*. PhD thesis, University of Pennsylvania, 1997.

[10] S. Y. Chen and Y. F. Li, "Vision sensor planning for 3-d model acquisition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 35, no. 5, pp. 894–904, 2005.

[11] V. Sequeira, J. Goncalves, and M. I. Ribeiro, "Active view selection for efficient 3d scene reconstruction," *ICPR'96*, vol. 1, 1996.

[12] P. S. Blaer and P. K. Allen, "Data acquisition and view planning for 3-d modeling tasks," in *Proc. of IROS*, pp. 417–422, 2007.

[13] P. Renton, M. Greenspan, H. A. ElMaraghy, and H. Zghal, "Plan-n-scan: A robotic system for collision-free autonomous exploration and workspace mapping," *Journal of Intelligent and Robotic Systems*, vol. 24, no. 3, pp. 207–234, 1999.

[14] Y. Yu and K. Gupta, "C-space entropy: A measure for view planning and exploration for general robot-sensor systems in unknown environments," *International Journal of Robotics Research*, pp. 1197–1223, 2004.

[15] E. Kruse, R. Gutsche, and F. Wahl, "Efficient, iterative, sensor based 3-d map building using rating functions in configuration space," in *Proc. of ICRA*, pp. 1067–1072, 1996.

[16] W. R. Scott, G. Roth, and J.-F. Rivest, "View planning for automated three-dimensional object reconstruction and inspection," *ACM Comput.Surv.*, vol. 35, no. 1, pp. 64–96, 2003.

[17] R. Pito, "A solution to the next best view problem for automated surface acquisition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 1016–1030, 1999.

[18] M. K. Reed, P. K. Allen, and I. Stamos, "Automated model acquisition from range images with view planning," in *In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 72–77, 1997.

[19] D. Papadopoulos-Orfanos and F. Schmitt, "Automatic 3-d digitization using a laser rangefinder with a small field of view," in *Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, IEEE Computer Society, 1997.

[20] N. A. Massios and R. B. Fisher, "A best next view selection algorithm incorporating a quality criterion," in *Proc. British Machine Vision Conference BMVC98, Southampton*, pp. 780–789, 1998.

[21] J. E. Banta and et al., "A next-best-view system for autonomous 3-d object reconstruction," *IEEE Transactions on Systems, Man and Cybernetics, Part A.*, vol. 30, no. 5, pp. 589–598, 2000.

[22] C. Connolly, "The determination of next best views," in *Proc. of ICRA*, vol. 2, pp. 432–435, 1985.

[23] S. Chen, Y. Li, J. Zhang, and W. Wang, *Active Sensor Planning for Multiview Vision Tasks*. Springer, 2008.

[24] B. W. He and Y. F. Li, "A next-best-view method with self-termination in active modeling of 3d objects," in *Proc. of IROS*, pp. 5345–5350, 2006.

[25] P. Whaite and F. P. Ferrie, "Autonomous exploration: Driven by uncertainty," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 3, pp. 193–205, 1997.

[26] J. Maver and R. Bajcsy, "Occlusions as a guide for planning the next view," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 417–433, May 1993.

[27] Y. Yu and K. Gupta, "An efficient online algorithm for direct octree construction from range images," in *Proc. of ICRA*, vol. 4, pp. 3079–3084, 1998.

[28] K. Morooka and et al., "Computations on a spherical view space for efficient planning of viewpoints in 3-d object modeling," *International Conference on 3D Digital Imaging and Modeling*, pp. 138–147, 1999.

[29] Y. Okubo, C. Ye, and J. Borenstein, "Characterization of the hokuyo urg-04lx laser rangefinder for mobile robot obstacle negotiation," *Unmanned Systems Technology XI*, vol. 7332, 2009.

[30] A. Atramentov and S. M. LaValle, "Efficient nearest neighbor searching for motion planning," in *Proc. of ICRA*, vol. 1, pp. 632–637, 2002.

[31] H. Hoppe and et al., "Surface reconstruction from unorganized points," *Computer Graphics*, vol. 26, no. 2, pp. 71–78, 1992.

[32] T. K. Dey and et al., "Normal estimation for point clouds: a comparison study for a voronoi based method," in *Eurographics/IEEE VGTC Symposium Proceedings on Point-Based Graphics*, pp. 39–46, 2005.

[33] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[34] L. Torabi, M. Kazemi, and K. Gupta, "Configuration space based efficient view planning and exploration with occupancy grids," in *Proc. of IROS*, pp. 2827–2832, 2007.

[35] "http://www.schunk-modular-robotics.com/."