2014 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS 2014)
September 14-18, 2014, Chicago, IL, USA

View Planning for 3D Object Reconstruction with a Mobile Manipulator Robot

J. Irving Vasquez-Gomez*, L. Enrique Sucar*, Rafael Murrieta-Cid†
*National Institute for Astrophysics, Optics and Electronics (INAOE), Puebla, Mexico
Email: {ivasquez,esucar}@inaoep.mx
†Center for Mathematical Research (CIMAT), Guanajuato, Mexico
Email: murrieta@cimat.mx

***Abstract*— The task addressed in this paper is to plan iteratively a set views in order to reconstruct an object using a mobile manipulator robot with an "eye-in-hand" sensor. The proposed method plans views directly in the configuration space avoiding the need of inverse kinematics. It is based on a fast evaluation and rejection of a set of candidate configurations. The main contributions are: a utility function to rank the views and an evaluation strategy implemented as a series of filters. Given that the candidate views are configurations, motion planning is solved using a rapidly-exploring random tree. The system is experimentally evaluated in simulation, contrasting it with previous work. We also present experiments with a real mobile manipulator robot, demonstrating the effectiveness of our method.**

I. Introduction

Three-dimensional (3D) models from real objects have several applications in robotics. For example, collision detection, object recognition, pose estimation, etc. Therefore, a mobile robot must have the ability of building 3D models from the objects in its environment for interacting with them further. The task of building a 3D model of an object is known as automated 3D object reconstruction [1]. It is a cycling process of observing and deciding where to see next. First, a range sensor is placed by the robot at certain location where a scan is taken. Then, if there are scans taken from previous iterations, the new one is transformed to a global reference frame and registered with previous scans. After that, the robot has to compute the next sensor pose which increases the reconstructed surface based on the available information (called next-best-view). The reconstruction is finished when a termination criterion is satisfied.

The problem addressed in this paper is to plan the next-best-view (NBV) in order to reconstruct an object using a mobile manipulator robot with an "eye-in-hand" sensor. In this work, the NBV is a robot configuration that sees (covers) the greatest amount of unknown area while several constraints are satisfied (detailed in III-A). To find the minimum set that covers an area has been demonstrated to be NP-Hard [2]. Therefore, the main challenge is to iteratively determine a set of NBV configurations that collectively cover the object, and a set of paths to reach them in a short processing time.

In [3], we proposed a utility function for a free-flyer robot. In [4], we proposed a hierarchical ray tracing for fast visibility calculation. *In contrast, in this work, our main contribution is a fast novel method to determining the NBV with a mobile manipulator robot*. We propose a new a utility function, implemented by a series of filters, that efficiently finds the NBV. Our approach contrasts with related work where the candidates are generated in the workspace and inverse kinematics is required reach them. The drawback of those methods is that the robot might not be physically able to reach a planned view (e.g. to observe the top of a given object). We directly generate views in the configuration space avoiding inverse kinematics calculation and taking into account only feasible views.

Fig. 1. The proposed method is able to plan each robot configuration in order to reconstruct a real object. In our experiments we use a mobile manipulator of 8 degrees of freedom to reconstruct a chair. A Kinect sensor was mounted on the robot's end effector.

The proposed approach provides an effective and fast method for a mobile manipulator to build 3D models of unknown objects. Effective means that a large percentage of the object surface is reconstructed, in our experiments, it is in the order of 95%. Fast means that the processing time to plan the NBV and a path to reach it takes typically less than a minute. We present different experimental results. In simulation, several complex objects are reconstructed. We validate the effectiveness of our utility function, comparing it versus information gain. The proposed utility function covers the same surface's percentage in a shorter processing time. We also present experimental results with a real mobile manipulator robot (see Fig. 1) with 8 degrees of freedom (DOF), showing the effectiveness of the method to deal with real objects.

978-1-4799-6934-0/14/$31.00 ©2014 IEEE

II. Related Work

Since the 80's the next-best-view (NBV) problem has been addressed. For a detailed review of classic methods see [1]. According to [1], our algorithm is volumetric and search based, so we will mainly review similar methods in this section. The work of Connolly [5] was one of the first in this field, it represents the object with an octree and determines the NBV with one of two approaches. The first one determines the NBV as the sum of normals from unknown voxels. The second method, called planetary, determines the NBV by testing views from a set around the object.

Foissotte *et al.* [6] propose an optimization algorithm to maximize the amount of unknown data in the camera's field of view (FOV). However, optimization methods can easily fall into local minima.

In our previous work [3], we have proposed a search over set of views around the object and a utility function which measures surface, quality and distance. A limitation of that approach is that the predefined set of pointing views might not always feasible, i.e., the robot can be in collision or the sensor can be occluded. Krainin *et al.* [7] proposed a method in which the robot grasps the object and moves it inside the camera's FOV. However, the robot might not have the ability to grasp and move the object.

Few works have considered both, the problem of finding *good* views, and the problem of obtaining the paths to reach them. Torabi and Gupta [8] address both problems, but differently to the approach proposed in this work. In that work, the authors plan a NBV in the workspace and then they use inverse kinematics to obtain a configuration that matches the desired sensor location in the workspace. In this work, we select sensing configurations directly in the configuration space and we plan the controls to reach them, also we take into account the length of the collision free path. A short path is better in terms of spatial uncertainty and it spends less robot's energy. Another work that addresses both problems is the one presented by Kriegel *et al.* in [9]. The authors combine two approaches for determining the NBV, a surface based and a volumetric method. First, they compute a set of candidate paths over the triangular mesh border, then they evaluate the goodness on the volumetric representation. Two important differences with our approach are: (i) Kriegel *et al.* propose the use of information gain (IG) to evaluate views, while in this work, the unknown surface is measured to evaluate views, (ii) in this work we consider a mobile manipulator while in [9], the authors only consider a robot arm.

III. Preliminaries

A. Next-best-view constraints

The set of candidate views used to perform the next-best-view search is denoted by $V = \{v_0, v_1 \dots v_m\}$. Each view is a tuple of the form: $v_i = (q_i, x_i, g)$ where q_i is the robot configuration, x_i is the sensor pose $x_i \in \mathbb{R}^3 \times SO(3)$, and g is the utility of the view. Our goal is to select a view $v \in V$ with the following characteristics:

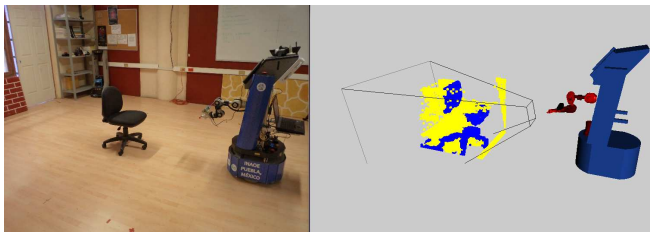


Fig. 2. Partial Model of the scene. To the left is the reconstruction scene. To the right is the partial model. The robot is represented by a triangular mesh. Unknown voxels are painted in yellow and occupied voxels are painted in blue (best seen in color).

1) New information. The NBV configuration must see unknown surfaces to completely observe the object.
2) Positioning constraint. The NBV must be collision free with the environment and the object.
3) Sensing constraint. Surfaces to be seen must be in the camera's field of view and depth of view. Also, the angle formed between the sensor's orientation and the surface normal must be smaller that a certain angle defined by the vision angle [10].
4) Registration constraint. The positioning error of the mobile robot causes that the current scan does not match with previous scans. Therefore, a registration step is required [11]. The NBV should guarantee that the scan will be registered.
5) Cost of the robot's motion. In a mobile manipulator, moving each degree of freedom (DOF) has a different cost, e.g., to move the arm instead of the base consumes less energy and causes lower positioning error. Therefore, a path where the degrees of freedom with the larger costs move less is desirable.

B. Working assumptions

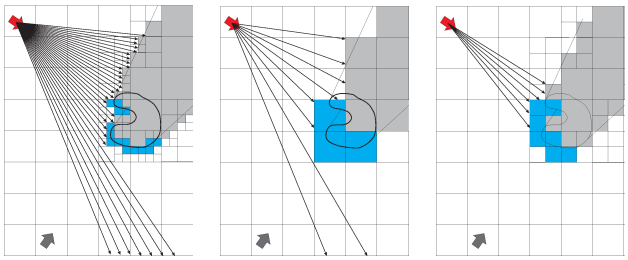
We take the following assumptions: i) the environment is known and it is represented by a 3D triangular mesh, ii) the position and maximum size of the object is known *a priori* to the robot and iii) the object is inside an object bounding box (OBB) marked as unknown volume.

IV. Partial Model

A. Probabilistic occupancy map

The partial model stores the information about the reconstruction scene, including the object and the environment. It shows what spaces have been sensed and what spaces remain unknown. In addition is also used detect collisions between the robot and the environment.

We use a probabilistic occupancy map based on the octomap structure [12], which is an octree with probabilistic occupancy estimation. See figure 2. *We stress the fact that we use a probabilistic octree, because it is able to deal with noise on the sensor readings*. From now on we refer to a probabilistic occupancy map as octree. Depending on the probability of been occupied, we classify each voxel with one of three possible labels: i) **occupied**, which represents surface points measured by the range sensor, ii) **free**, which



(a) Uniform ray tracing. All sensor rays are traced in the octree.

(b) Rays traced in a coarse octree.

(c) Ray tracing in a finer octree only for touched voxels.

Fig. 3. Examples of uniform ray tracing and hierarchical ray tracing.

represents free space and iii) **unknown**, whose space has not been seen by the sensor. Each label has a defined probability interval. In our implementation the unknown voxel label has the interval $[0.45, 0.55]$. To use an interval for unknown voxels, instead of a fixed probability of 0.5, increases the confidence of the occupied or empty volumes of the resultant octree.

In order to evaluate new information, sensing and registration constraints of a candidate view, we perform a visibility computation over the octree. Usually, this task is achieved with a uniform ray tracing, it traces a number of rays inside the map simulating a range sensor (Fig. 3(a)). However, such process can be highly expensive if the voxels' size is small. To reduce the processing time, we use a variant of the hierarchical ray tracing presented in [4].

B. Hierarchical ray tracing

In [4], we proposed a hierarchical ray tracing (HRT). It is based on tracing few rays in a rough resolution map; then, only when occupied voxels are touched by a ray, the resolution is increased for observing details (see Fig. 3). The coarsest resolution where the HRT starts is defined by a resolution parameter (a); when a is equal to 0 a uniform ray tracing is performed, for $a > 0$ the ray tracing starts in a octree resolution with a voxel size of 2^a times the original size. Such strategy typically reduces the processing time needed to evaluate a view in at least one order of magnitude. In our previous work we only increase the resolution when a occupied voxels is touched. In this work, we increase the resolution when occupied and unknown voxels are touch. This upgrade allow us to reduce the processing time but usually keeping the same coverage with the same number of required views. Section VII-B details several experiments where there is 60% of processing time reduction with only a loss of 1% of coverage.

V. Next-Best-View Selection

We propose a search based method to compute the NBV. Fig. 4 shows the flowchart of the proposed method. First, a set of candidate views is generated, then the views are evaluated and ranked with a utility function. The NBV is the one that maximizes the utility. The evaluation of the utility function for a large set of views can be a highly expensive process. To mitigate this problem, we propose an efficient evaluation scheme, in which if a candidate view does not pass a filter it is deleted from the set of candidate views.

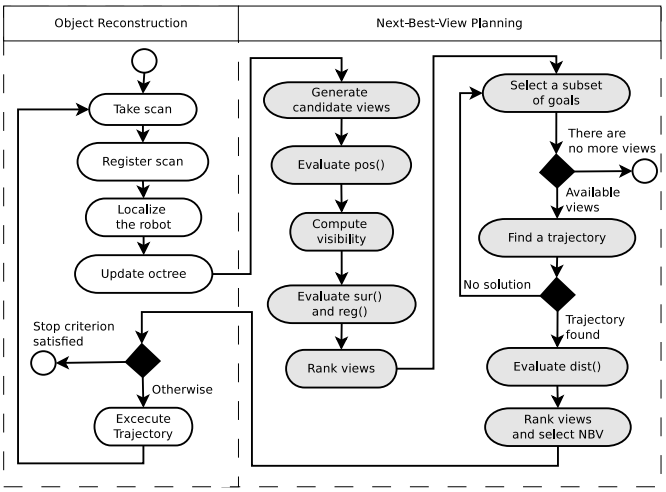


Fig. 4. 3D object reconstruction with next-best-view planning. The diagram shows the whole process of object reconstruction. The processes related with the NBV computation are filled in gray (see Section V-C for details).

A. Utility function

The utility function ranks the candidate views according to their goodness for the reconstruction process. We propose a utility function as a product of factors:

$$g(v) = pos(v) \cdot reg(v) \cdot sur(v) \cdot dist(v) \tag{1}$$

where each factor evaluates a constraint, below we detail each constraint.

1) Positioning: $pos(v)$ is 1 when a robot configuration is collision free, and a collision free path from the current configuration to the evaluated configuration is available; otherwise it is 0.

2) Registration: To register the new scan, previous works have proposed to assure a minimum amount of overlap [4] or consider all causes of failure [13]. A minimum overlap is a necessary but not sufficient condition to guarantee registration. However, it requires a small processing time. On the other hand, to measure all causes of failure guarantee a successful registration but is very expensive (as described in [14]). Therefore, in this work, we propose a simple factor that is fast for evaluation, $reg(v)$. It is 1 if a minimum percent of overlap with previous surfaces exist, and 0 otherwise. See equation (2).

$$reg(v) = \begin{cases} 1 & \text{if } \dfrac{oc_o(v)}{oc_o(v) + un_o(v)} > h \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $oc_o(v)$ indicates the amount of occupied voxels that are touched by the sensor and lie inside the object bounding box (OBB), $un_o(v)$ is the amount of unknown voxels in the OBB, and h is a threshold (in our experiments in the real robot this threshold is set to 55%). This factor allows us to

deal with a big amount of views and has been tested in the experiments with a real robot with good results.

3) New surface: $sur(v)$ evaluates a view depending on how much surface from the unknown volume is seen, i.e. the amount of visible unknown voxels. Such function returns values between 1 and 0. It is 1 when v sees all the unknown voxels in the OBB, see equation (3).

$$sur(v) = \frac{un_o(v)}{un_{total}} \tag{3}$$

where un_{total} is the total amount of unknown voxels inside the OBB.

4) Distance: Candidate views are also evaluated according to their distance to the current robot configuration. The function is shown in eq. (4):

$$dist(q_n) = \frac{1}{1 + \rho(q_0, q_n)} \tag{4}$$

where ρ is the summation of the weighted Euclidean distance between the nodes of the path $P = \{q_0, q_1 ... q_n\}$ between the current robot configuration q_0 and the candidate configuration q_n, as defined in equation (5).

$$\rho(q_0, q_n) = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{m} w_j (q_i(a_1, a_j, a_m) - q_{i-1}(a_1, a_j, a_m))^2} \tag{5}$$

where a_j is the j-th degree of freedom, w_j is a weight assigned that degree of freedom, and m is the number of degrees of freedom.

Unlike our previous approach, where a distance in the workspace was defined [3], this distance measures the path followed by the robot, which in most of the cases is not a straight line, i.e., the robot has to avoid obstacles or needs a trajectory different to a straight line due to non-holonomic constraints.

B. Candidate views

In this work, we directly generate views in the robot's configuration space. For each iteration a set of random samples using a uniformly distribution is generated. Then they are filtered though the evaluation strategy. Robot's configuration space could be very vast and many samples could be needed to get useful samples. However, our first filter is to check if the sensor points to the object, a constant time process that fast discards views. In our experiments, for each iteration, 10,000 random samples are generated. Usually for each 10,000 views few hundreds of views are useful. Only the useful views are kept for further evaluation.

C. Evaluation strategy

In order to evaluate the candidates efficiently we perform the evaluation trough several filters. If a candidate does not pass a filter it is deleted from the candidate view set. Fig. 4 shows how the filters are applied. First, we discard views if the sensor director's rays does not intersect a sphere that encapsulates the object. Then, we test the candidates with the positioning factor. This verifies for collisions between the candidate configuration and the environment (represented by an octree, in which occupied and unknown voxels are considered as obstacles). Then, we compute visibility with hierarchical ray tracing, as explained in section IV-B. After that, registration and surface factors are evaluated. Then, a subset of candidates is sent to the motion planner, the planner tries to find a trajectory to each candidate. If no solution was found a new subset is sent. Finally, the weighted length of the path is considered.

In the previous evaluation process, the processing time for finding collision free paths for several configurations might be too large, in particular if some of those are not reachable. An alternative is to estimate the distance as a straight line in the configuration space. Then, this estimation could be used to rank the set of views before motion planning and only the best one is given as goal to the RRT planner. If the selected configuration cannot be reached, the second best is given as a goal to the RRT, and so on. In the simulation experiments we compare both alternatives.

D. Stop Criterion

Establishing a good stop criterion for object reconstruction is a challenging problem. Previous works have proposed several criteria based on the partial model of the object [8], [9]. However, to consider only the object is not enough, i.e., the robot configuration can be in collision with the environment or an obstacle is blocking the way. In this work, the process is finished if the surface factor is lower than a threshold, or no path was found for any of the candidate views.

VI. Motion planning

The Rapidly Exploring Random Trees method (RRT) is a data structure and algorithm that is designed for efficiently searching no convex high-dimensional configurations spaces [15], [16]. RRT can be considered as a Monte Carlo way of biasing search into largest Voronoi regions. RRT and related methods have been used in a variety of applications, for instance motion planning for humanoid robots [17], or to plan navigation routes for a Mars rover that take into account dynamics [18]. In [19], a sensor-based RRT, called SRT, is used for exploration of 2D environments. In [20], the authors have extended RRT and other sampling-based motion planning algorithms to find optimal paths.

In this work, we adapt the RRT to plan robotic paths for building 3-D models of unknown objects. We use the RRT-EXT-EXT variant, which grows two balanced trees, one from the initial configuration and one from the goal.

VII. Experiments

In this section, we present experiments both in simulation and with a real robot. First, we validate our utility function comparing it with a state of the art utility function [9]. Second, we measure the processing time gain that we can achieve when the hierarchical ray tracing is used. Third, we show the effectiveness of the method by reconstructing

(a) Bunny (b) Teapot (c) Dragon

Fig. 5. Synthetic objects used for testing.

several complex objects. Fourth, we experimentally compare the two different ways to select the views for the motion planner. Fifth, the reconstruction of a office chair with a real robot is described, in order to show the effectiveness of the method in a real case.

Simulation experiments are performed in a scene where the objects are placed over a table; the object bounding box is over the table, the simulated sensor is a time of flight camera. The robot is the same as our real robot with 8 degrees of freedom (Fig. 1). Three complex objects have been tested, the Stanford Bunny, a Dragon and a Teapot, see figure 5. For the simulation experiments we have fixed a voxel resolution of 2 cm. The percentage of covered surface is computed as the ratio of correspondent points in the built model over the total number of points in the ground truth model. A point in the built 3-D model finds a correspondence in the ground truth model if it is closer than a threshold (3 mm). Average results are calculated after repeating the reconstruction 5 times.

A. *Utility function validation*

In this work, we estimate the goodness of a view measuring three main factors: i) unknown surface (amount of unknown voxels), ii) overlap and iii) c-space path distance. We call this utility function combined utility (CU). Another way to estimate the goodness is computing the Information Gain (IG) of a scan [9]. In this experiment we compare CU versus IG. The comparison was done by reconstructing 5 times the Bunny object using CU, and 5 times using IG.

Figures 6 and 7 show the average surface coverage and average unknown volume, respectively. In this experiment, IG at initial iterations gets a higher coverage than CU. This happens, because IG only takes into account the new information that is expected to see. Unlike CU that takes into account overlap and distance to reach a view configuration. After several scans both utilities converge to the same coverage. Thus, after several scans our utility function reaches the same coverage than IG, but assures overlap and measures the cost of the path (distance traveled).

Processing time for evaluating both function is quite similar. In our implementation, to evaluate CU for all candidate views takes an average time of 428 s., in contrast, IG factor takes 484 s., that is 13% more than CU. It is worth to say that, the evaluation processing time of CU can be significant reduced using the hierarchical tray tracing. To the best of our knowledge there is no speed up technique for IG computation. Next section specifies the time required for evaluating a single view using HRT.

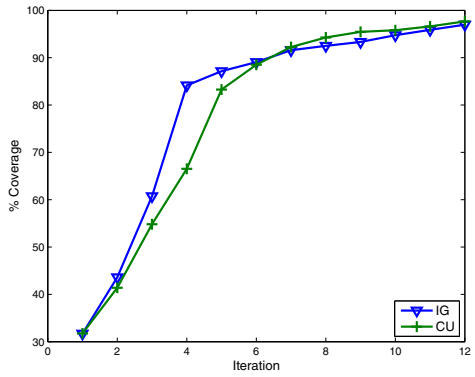


Fig. 6. Comparison of the average surface coverage using information gain (IG) and unknown surface (CU). Both methods converge to the same coverage. Note that coverage does not reach 100% given that the base of the object is occluded by the table.

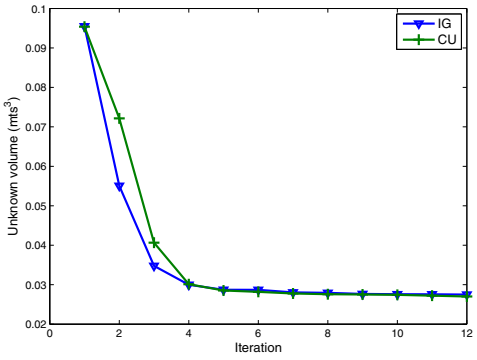
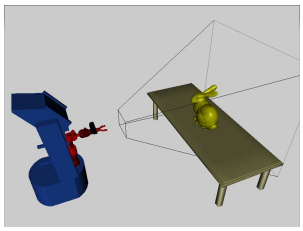


Fig. 7. Comparison of the unknown volume in the octree using information gain (IG) and unknown surface (CU). Both methods leave the same unknown volume in the octree.

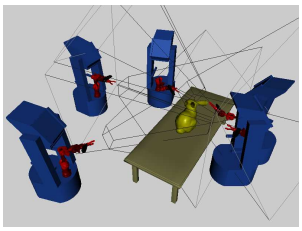
B. *Speed up of the utility function evaluation*

The proposed Hierarchical Ray Tracing (HRT) reduces the visibility computation time, depending on the resolution parameter (section IV-B). We have tested different resolution parameters for the reconstruction of the Bunny object. The results of the required processing time are shown in table I. The first column shows the resolution parameter a used for the reconstruction. Remember that a equal to zero is equivalent to a uniform ray tracing. The second column shows the average time required to evaluate a single view that points to the object. The third column shows the voxel size at the roughest resolution, in which HRT starts. The fourth column shows the coverage percentage after 12 scans of the Bunny. A reduction of 60% of the processing time is gained with $a = 1$. For higher resolution parameters there is a further reduction in processing time, until $a = 4$. Larger resolution parameters do not imply time reduction, given that the overhead of the ray tracing structure increases the processing time.

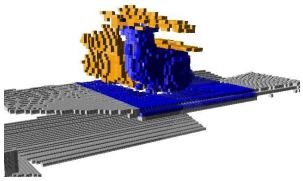
Using the HRT allow us to evaluate a large set of views in a short time, making possible that even a naive set of random



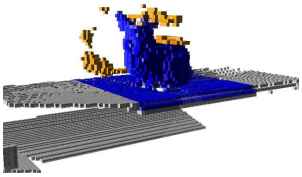
(a) Initial robot configuration.



(b) First five next-best-views computed to scan the object.



(c) Octree at a given iteration.



(d) Updated octree after scan.

Fig. 8. Illustrations from different stages of the reconstruction of the Bunny object. Unknown voxels are shown in orange, occupied voxels belonging to the object are displayed in blue and the occupied voxels belonging to the known environment are displayed in gray (best seen in color).

Fig. 9. Final representations (point clouds) from the reconstructed 3D synthetic objects.

views could be useful to determine the NBV. A drawback of this method is that there is no finner ray tracing for obstacles outside the bounding box. Therefore, the best performance of the HRT is when the object has a clear space around it.

TABLE I

VIEW EVALUATION TIME WITH HIERARCHICAL RAY TRACING.

Res. param.	Time (s)	Voxel size	% Coverage
0	0.185	0.02 m	97.66
1	0.063	0.04 m	96.35
2	0.035	0.16 m	96.26
3	0.024	0.32 m	96.25

C. Reconstruction of complex objects

In this experiment, we test the method using the Bunny, the Dragon and the Teapot. We present quantitative results to evaluate the performance and efficiency of the proposed approach. Fig. 8 depicts several stages in the reconstruction of the Bunny. Fig. 9 shows the final representation of the reconstructed objects. We use a resolution parameter $a = 2$ and 4 goals were sent to the motion planner.

Table II presents average results for the reconstruction of the objects in terms of the number of views needed to reconstruct the 3-D model, the visibility computation time (Vis.), the motion planning processing time (M.P.) and the percentage of covered surface. The results show that the method is able to plan each NBV for 3D object reconstruction with a mobile manipulator with eight DOF.

TABLE II

RECONSTRUCTION RESULTS FOR EACH OBJECT.

Object	Views	Vis.	M.P.	Coverage.
Teapot	12	48.92 s	79.05 s	93.90 %
Bunny	12	33.81 s	114.07 s	95.51 %
Dragon	12	32.45 s	95.60 s	87.26 %

D. Evaluation strategies

In this experiment, we compare the two evaluation strategies described in Section V-C, they differ on how many goals are sent to the RRT. These strategies are related with the navigation distance estimation. In the experiment we also compare the performance of the method if no navigation distance is used. Table III presents, average results for the reconstruction of the Bunny object for each strategy in terms of the visibility processing time (Vis.), the motion planning processing time (M.P.), the percentage of coverage and the distance traveled to visit all the sensing configurations. The compared strategies are: no navigation distance factor, single goal (distance is estimated as a straight line in the configuration space) and subset of n goals (distance is calculated with equation (5) and n goals are sent to the Motion Planner). The results show that, in general, the traveled distance is decreased whether a navigation factor is used, also, they show that the strategy of sending a subset of goals to the RRT covers more surface with a shorter path with respect to one single goal. However, such strategy requires more processing time.

TABLE III

RECONSTRUCTION RESULTS FOR EACH STRATEGY.

Strategy	Vis.	M.P.	Coverage	Dist.
No distance	33.85 s	40.21 s	96.21 %	89.24
Single goal	35.68 s	15.81 s	95.40 %	72.26
Subset of 5 goals	32.89 s	117.53 s	97.23 %	72.31
Subset of 10 goals	35.90 s	182.87 s	96.32 %	62.45

E. Real case reconstruction

This experiment performs the reconstruction of an office chair with a Microsoft Kinect sensor mounted on a mobile manipulator. See Figure 2. The objective is to show experimentally that the method can deal with a real environment in acceptable processing times. The implemented registration process uses Iterative Closest Point Algorithm (ICP) [11], the registration factor was set to 55% and the size of the voxels was 3 cm, the HRT resolution parameter was set to 2.

The proposed method provides a reliable and fast solution for finding the NBV according to the robot capabilities. Fig. 10 shows the acquired model of the chair. The object was seen from different locations and most of the surface was reconstructed. Table IV shows the average times for each

iteration of the reconstruction. Average times are the mean of the spent time per iteration.

TABLE IV
RESULTS OF THE REAL RECONSTRUCTION.

Number of scans	11
Traveled distance	76.06
Visibility evaluation	10.74 s
Motion planning	8.22 s
Octree update	6.61 s

Fig. 10. Qualitative comparison of two views of the reconstructed object versus the real object. Almost all surface was seen, except the low part of the seat where the robot cannot reach due to the size of the arm.

VIII. CONCLUSIONS AND FUTURE WORK

We have presented a method for next-best-view planning for 3D object reconstruction. This is one of the first methods that determines the NBV directly in the configuration space, following a methodology in which a set of candidate views is directly generated in the configuration space, and later only a subset of these views is kept by filtering the original set. The method proposes a utility function that integrates several relevant aspects of the problem and an efficient strategy to evaluate the candidate views. This method avoids the problems of inverse kinematics and unreachable poses.

We compare the proposed approach with related works both qualitatively and quantitatively. Qualitatively, this approach measures the goodness of the path in terms of unknown surface, overlap and the cost of each degree of freedom and also performs an efficient evaluation of the candidate views. Quantitatively, this method achieves the same coverage but with smaller processing time compared with previous works. In our experiments we have used a mobile manipulator of 8 DOF with an eye-in-hand sensor. To our knowledge, this is one of the first works in which a method to reconstruct a 3-D object is implemented in a real mobile manipulator robot.

As future work, we want to deal with spatial uncertainty, which decreases the real goodness of a planned view or causes collision between the robot and the object.

REFERENCES

[1] W. R. Scott, G. Roth, and J.-F. Rivest, "View planning for automated three-dimensional object reconstruction and inspection," *ACM Comput. Surv.*, vol. 35, pp. 64–96, March 2003.

[2] J. O'Rourke, *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[3] J. I. Vasquez, E. Lopez-Damian, and L. E. Sucar, "View Planning for 3D Object Reconstruction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS09)*, 2009, pp. 4015–4020.

[4] J. I. Vasquez, L. E. Sucar, and R. Murrieta-Cid, "Hierarchical Ray Tracing for Fast Volumetric Next-Best-View Planning," in *Tenth Canadian Conference on Computer and Robot Vision 2013*, 2013.

[5] C. Connolly, "The determination of next best views," in *IEEE International Conference on Robotics and Automation 1985, ICRA85*, vol. 2, 1985, pp. 432–435.

[6] T. Foissotte, O. Stasse, A. Escande, P.-B. Wieber, and A. Kheddar, "A two-steps next-best-view algorithm for autonomous 3d object modeling by a humanoid robot," in *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ser. ICRA'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 1078–1083.

[7] M. Krainin, B. Curless, and D. Fox, "Autonomous generation of complete 3d object models using next best view manipulation planning," in *ICRA*, 2011, pp. 5031–5037.

[8] L. Torabi and K. Gupta, "An autonomous six-dof eye-in-hand system for in situ 3d object modeling," *The International Journal of Robotics Research*, vol. 31, no. 1, pp. 82–100, 2012.

[9] S. Kriegel, C. Rink, T. Bodenmuller, A. Narr, M. Suppa, and G. Hirzinger, "Next-best-scan planning for autonomous 3d modeling," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct., pp. 2850–2856.

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

[11] P. Besl and N. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 239–256, 1992.

[12] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, AK, USA, May 2010.

[13] K. Low and A. Lastra, "An adaptive hierarchical next-best-view algorithm for 3d reconstruction of indoor scenes," Proceedings of 14th Pacific Conference on Computer Graphics and Applications (Pacific Graphics, Tech. Rep., 2006.

[14] K.-L. Low and A. Lastra, "Predetermination of icp registration errors and its application to view planning," in *Proceedings of the International Conference on 3D Digital Imaging and Modeling*, 2007, pp. 73–80.

[15] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[16] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[17] J. J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Motion planning for humanoid robots," in *ISRR*, 2003, pp. 365–374.

[18] C. Urmson and R. Simmons, "Approaches for heuristically biasing rrt growth," in *IEEE/RSJ IROS 2003*, October 2003.

[19] G. Oriolo, M. Vendittelli, L. Freda, and G. Troso, "The srt method: Randomized strategies for exploration," in *IEEE International Conference on Robotics and Automation (ICRA-01)*, 2001, pp. 4688–4694.

[20] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.