Noname manuscript No.
(will be inserted by the editor)

View/State Planning for Three-dimensional Object Reconstruction under Uncertainty

J. Irving Vasquez-Gomez · L. Enrique Sucar · Rafael Murrieta-Cid

the date of receipt and acceptance should be inserted later

Abstract We propose a holistic approach for three-dimensional (3D) object reconstruction with a mobile manipulator robot with an eye-in-hand sensor; considering the plan to reach the desired view/state, and the uncertainty in both observations and controls. This is one of the first methods that determines the next best view/state in the state space, following a methodology in which a set of candidate views/states is directly generated in the state space, and later only a subset of these views is kept by filtering the original set. It also determines the controls that yield a collision free trajectory to reach a state using Rapidly-Exploring Random Trees (RRTs). To decrease the processing time we propose an efficient evaluation strategy based on filters, and a 3D visibility calculation with hierarchical ray tracing. The next best view/state is selected based on the expected utility, generating samples in the control space based on an error distribution according to the dynamics of the robot. This makes the method robust to positioning error, significantly reducing the collision rate and increasing the coverage, as shown in the experiments. Several experiments in simulation and with a real mobile manipulator robot with 8 degrees of freedom show that the proposed method provides an effective and fast method for a mobile manipulator to build 3D models of unknown objects. To our knowledge, this is one of the first works that demonstrates the reconstruction of complex objects with a real mobile manipulator considering uncertainty in the controls.

Keywords Next best view · motion planning · object reconstruction · uncertainty

J. Irving Vasquez-Gomez and L. Enrique Sucar
Instituto Nacional de Astrofísica Óptica y Electrónica (INAOE), Puebla, México
E-mail: ivasquez@ccc.inaoep.mx esucar@inaoep.mx

Rafael Murrieta-Cid
Centro de Investigación en Matemáticas (CIMAT), Guanajuato, México
E-mail: murrieta@cimat.mx

1 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 of 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 [25]. Given that the object is unknown, the reconstruction is a cycling process of observing and deciding where to see next. First, a range sensor is placed by the robot at a 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, such sensor pose is called next best view. The reconstruction is finished when a termination criterion is satisfied.

A Next Best View (NBV) is a sensor pose, position and orientation, that sees (covers) the greatest amount of unknown area while several constraints are kept. We assume that the sensor is mounted on the end effector of a mobile manipulator robot. Therefore, to plan the NBV requires to compute the robot state and to determine a trajectory to reach the NBV. In this paper we propose a method that plans the Next Best View/State (NBVS) and a trajectory to reach it, instead of using

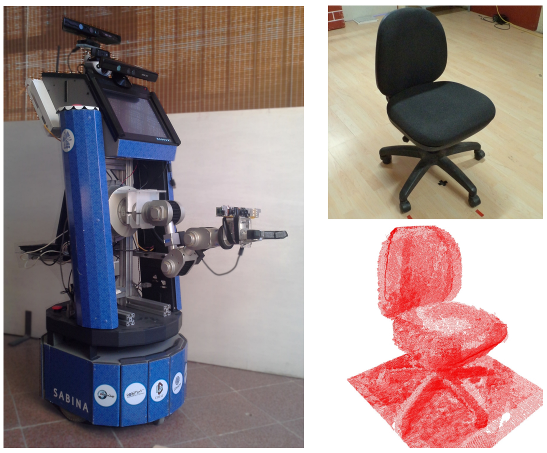


Fig. 1 The proposed method is able to plan each robot view/state in order to reconstruct a real object. In our experiments we use a mobile manipulator of 8 degrees of freedom to reconstruct several objects. A Kinect sensor was mounted on the robot's end effector.

only the sensor pose. A view/state is a collision free robot state that satisfies the constraints of the next best view. When the robot moves it does not correct its pose continuously; the robot is re-localized after each scan using the overlap between the partial model of the object and the new scanned surface. The re-localization is done with respect to a reference frame defined by the object.

A critical problem is that the reached robot's state differs from the planned one, due to the fact that the result from an applied control is uncertain [26]. The difference between the planned sensor pose and the reached sensor pose is called positioning system error [24]. This error reduces the measurement precision, sampling density, visibility and coverage. Furthermore, during the execution of a trajectory the robot can collide with the environment or the object.

In the early state of our research in this topic, we have proposed an approach [33,36] that works for a *freeflyer sensor. That approach does not consider the robot constraints.* Later, in [34] we have proposed a hierarchical ray tracing that efficiently computes approximated robots visibility.

A preliminary version of some parts of this work have been presented in [35]. In [35], we have proposed a deterministic utility function that integrates several relevant aspects of the problem and an efficient strategy to evaluate the candidate views. We have also integrated the hierarchical ray tracing [34] to the computation of the deterministic utility function. The main differences between the work presented here and the one presented in [35] are the following:

1. In this work, we have proposed a planning method that directly generates the robots controls yielding trajectories to reach a view/state and an expected utility function that reduces the collision rate during the execution phase.
2. We have also tested the proposed approach under several conditions. In [35] only one experiment in the real robotic system is presented, in this work we present two additional experiments that have not been presented in [35]. One experiment is made in an environment with obstacles and the other presents the reconstruction of a complex object. In this work we also present simulations comparing the proposed approach versus the one presented in [12], as well as simulations under motion uncertainty. These simulation results have not been presented in [35].
3. We have included new simulation experiments comparing the expected utility versus a deterministic one in a cluttered environment. These simulation experiments show that the expected utility significantly reduces the collision rate and increased the object's coverage.
4. We have also integrated in this work the efficient ray tracing [34] and the efficient evaluation strategy [35], which makes it possible to compute the expected utility in a reasonable amount of processing time.

In more detail, here we propose a method that determines a NBVS robust to positioning error. The proposed search-based method ranks a set of candidate views/states with the novel concept of expected utility. The expected utility is the most likely utility that a view/state will have under positioning error conditions. To determine the expected utility we generate several samples on the control space, based on the error distribution, then we transform those samples to the utility space and finally we compute the expected utility by associating a probability to each sample.

One of the contributions of this work is the incorporation of the expected utility, which is suitable for finding collision free trajectories under control errors at the execution time. Note that even if a motion planning method does find a collision free trajectory at the planning time, due to imperfect controls that trajectory might produce a collision with the obstacles at the execution time due to noise on the controls. The analysis presented in Subsection 7.2 estimates the probability of finding collision free trajectories at execution time and it is based on statistics over the trajectories generated with control errors. In that analysis we compute the number of samples required to guarantee with certain confidence that a trajectory is collision free.

We present experimental results in a simulated environment. We also present experiments with a real robot where several objects are reconstructed. In those experiments the expected utility method significantly reduces the collision rate and increases the reconstruction coverage.

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 for deterministic utility and the processing time remains in the order of minutes for expected utility. We present different experimental results. 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.

2 Related Work

Since the 80's the next-best-view (NBV) problem has been addressed. For a detailed review of classical methods see [25]. According to [25], our algorithm is volumetric and search-based, so we will mainly review similar methods in this section.

The work of Connolly [2] 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.

A pioneering work in the field of 3D scene reconstruction is the one presented in [20]. In that work the authors combine visual servoing and Bayes nets to generate motion strategies for 3D scene reconstruction. Our work proposes different techniques such as sampling based motion planning algorithms in the state space (configuration plus velocities), and an expected utility to deal with controls errors at execution time.

Foissotte *et al.* [4] 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 minimum. Krainin *et al.* [11] proposed a method in which the robot grasps the object and moves it inside the camera's FOV. In [1], the authors propose a method in which a robot manipulator actively manipulates a target object and generates a complete model by accumulation and registration of partial views. However, the robot might not have the ability to grasp and move the object.

2.1 Path Planning for Next Best View

Few works have considered the problem of finding *good* views and the problem of obtaining the robot paths to reach them. In [29], the proposed method plans a NBV in the workspace and then inverse kinematics is calculated to obtain a configuration that matches the desired sensor location in the workspace. In this work, we select views/states directly in the state space and we plan the controls to reach them. In [12], the authors combine two approaches for determining the NBV, surface based and volumetric based methods. First, they compute a set of candidate paths over the border of the reconstructed triangular mesh, 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 (In section 8.1.4, we present a comparison of both approaches), (ii) in this work we consider a mobile manipulator, while in [12], the authors consider a robot arm with perfect positioning.

In [28, 27] a mobile manipulator robot is used to reconstruct an object. In that work, the authors present a planner which integrates two next best view (NBV) algorithms, one for modeling the object and the other for exploration of the environment. For simplifying the problem of finding collision free paths, path planning is decoupled, the mobile base and the manipulator do not move simultaneously (i.e., when the mobile base is moving, the manipulator will remain still and vice-versa). In contrast to [28], in this work we assume that the workspace is known by the robot, but we allow the robotic base and the arm to move simultaneously and the controls for the whole robotic system are directly generated by the planning algorithms. We also take into account noise over the robot's controls. Furthermore, we introduce the expected utility which is useful for reducing the collision rate at execution time.

The aim of the work presented in [13] is to obtain a high quality surface model allowing for robotic applications such as grasping and manipulation. It integrates 3D modeling methods with autonomous view planning and collision-free path planning. That work uses Rapidly-Exploring Random Trees (RRTs) and Probabilistic Road Maps (PRMs) to find collision free paths. However, the integration of those planning methods in the resulting robotic system is not described in detail.

The work presented in [10] proposes an approach to determine the Next Best View for an efficient reconstruction of highly accurate 3D models. The method is based on the classification of the acquired surfaces into *Well visible* and *Barely visible* combined with a best view selection algorithm based on mean shift. That work has been tested in a robotic cell including a robotic arm. In [10] the authors propose as future work to deal with issues mainly related to path planning. A key difference between the work presented in this paper and the approach presented in [13] and [10] is that those works do not deal with noise over the robot controls. The expected utility proposed here finds collision free trajectories under control errors at execution time.

In the recent work presented in [23], the authors present an interesting information gain-based variant of the next best view problem for a cluttered environment. The authors propose a belief model that allows them to obtain an accurate prediction of the potential information gain of new viewing locations. However, the authors do not deal with the problem of planning collision free paths under motion uncertainty. In this work, we deal with that problem.

2.2 Motion Planning under Uncertainty

There are several approaches for motion planning under uncertainty. In [31] a classification of these types of methods is provided. This classification is based on the type of uncertainty: i) motion uncertainty, ii) uncertainty on the observation and state and iii) uncertainty on the environment itself. In this work, we consider uncertainty in the controls and observations but we assume that the environment is not uncertain. We deal with uncertainty on both, controls and states, using a probabilistic motion model and a model to estimate probabilistically a collision free state and trajectory to reach it. We deal with uncertainty on the observations using a probabilistic octree.

Motion planners that deal with uncertainty typically select the less uncertain paths or trajectories based on an evaluation of candidates. Some selection criteria are minimal uncertainty or the smallest collision probability. In [21], an extension to the *Rapidly-Exploring Random Tree* (RRT) [16] is proposed, which modifies the extension step in the RRTs performing simulations based on a stochastic kinematic model. A set of simulations generates a set of particles representing the robot state. The particles are grouped based on a given distance. Each group of particles represents a possible state. In [9] several extensions to deal with uncertainty for RRTs are proposed. In particular, the authors deal with rough terrains. The authors propose a probabilistic model for the terrain and, based on this model, a robot trajectory is selected.

The *Probabilistic Road Map* (PRM) technique [8] has also been adapted to deal with uncertainty. In [6] the shortest path that satisfies a threshold of the probability of collision is selected.

In a recent work [32], van den Berg et al. propose an interesting local optimization approach to find locally optimal stochastic controls. The authors deal with the case of an specific non-linear systems with a Partially Observable Markov Decision Process (POMDP). To make the problem tractable, the belief function is represented in a finite dimensional space by considering Gaussian distributions and implementing the Bayes filter as an extended Kalman filter. The key idea is to recursively quadratize the optimal value function based on the belief.

In contrast to previous work, in this paper we focus on the effect that the uncertainty has over the task of next best view/state planning for object reconstruction. In particular we deal with the problem of generating collision free states and trajectories to reach them. We use a probabilistic motion model as the one proposed in [26]. In the proposed method the samples are used to determine the expected utility to reconstruct an object, selecting the controls to reach the view/state with the maximum expected utility.

3 Definitions and Notation

The workspace, $\mathcal{W}$, is a 3D Euclidean space, $\mathcal{W} = \mathbb{R}^3$ [15]. Let $\mathcal{W}_{obj}$ be the object which is a closed set of points in the workspace. We assume that the object shape is unknown but the position and maximum size of the object is known, with this information an object bounding box, $\mathcal{W}_{box} \subset \mathcal{W}$, is established containing the object to be reconstructed, $\mathcal{W}_{obj} \subset \mathcal{W}_{box}$. The unknown region is the space that hides the object surfaces until a scan is made in that region. Let $\mathcal{W}_{unk}$ denote the unknown region. At the beginning of the reconstruction $\mathcal{W}_{unk} = \mathcal{W}_{box}$.

We assume that the environment except W_{box} is known. Let $\mathcal{O}_{env}$ denote the other obstacles in the environment. Let $\mathcal{O}$ denote the obstacle region so that $\mathcal{O} = \mathcal{W}_{obj} \cup \mathcal{W}_{unk} \cup \mathcal{O}_{env}$. The free space, $\mathcal{W}_{free}$, is the complement of the obstacle region $\mathcal{W}_{free} = \mathcal{W} \setminus \mathcal{O}$.

A range sensor is able to acquire a $2\frac{1}{2}$ image from the scene, i.e., a set of 3D points with respect to the sensor's reference frame. Let $\mathcal{V}$ denote the view space which represents all possible combinations of the sensor position and orientation, $\mathcal{V} \subseteq \mathbb{R}^3 \times SO(3)$. Each element of $\mathcal{V}$ is called a view and it is represented as v. The director ray of the sensor is a vector that points in the orientation of the sensor.

The robot, $\mathcal{A} \subset \mathbb{R}^3$, is the device in charge of placing the sensor at a given view. A configuration of the robot is denoted by q. Our particular robot has eight degrees of freedom (DOF). The position of the robotic base is defined by x, y, the orientation of the robotic base is denoted by θ_b, and $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ denote the orientations of the links of the robotic arm. The set of all possible configurations is the configuration space, $\mathcal{C}$. Let $\mathcal{A}(q) \subset W$ denote the physical space occupied by the robot and the sensor at configuration q. The free configuration space is a subset of $\mathcal{C}$ in which the robot is not in collision with the environment, $C_{free} = \{q|\mathcal{A}(q) \cap \mathcal{O} = \emptyset\}$. $\mathcal{C}_{obs} = \mathcal{C} \setminus \mathcal{C}_{free}$. Let $\mathcal{X}$ denote the state space in which a state, $x \in \mathcal{X}$, is defined as $x = (q, \dot{q})$ [16]. Thus, in this work, the robot's state is a configuration plus the velocities applied to the robot's degrees of freedom used to reach such configuration. Besides, those velocities are not perfect; we assume noise over them, this noise appears in real robotic systems (see Section 6).

We assume that the robot is controlled by a set of velocities applied to each DOF, called control, $u \in \mathcal{U}$. The variation with respect to time of the robot's state is given by the state transition equation:

$$\dot{x} = f(x, u) \tag{1}$$

The robot moves without correcting its pose continuously, but the robot is re-localized after each scan using the matching between the reconstructed point cloud and the new scanned surface. Before the first robot motion, we assume that the positions and orientations of the mobile base and the arm are accurately known with respect to a reference frame defined by the object bounding box.

4 Approach Overview

Here, we present a brief overview of the whole strategy for 3D object reconstruction. Fig. 2 shows the flow diagram of the NBVS planning for object reconstruction. Below we describe the strategy.

The whole process involves two main parts: (i) 3D reconstruction and (ii) next best view/state planning. Our contributions in this work are mainly in the second part.

The 3D reconstruction consists of integrating a series of scans of the object of interest taken from the different sensing positions. The initial view/state is set arbitrarily, and the next ones are determined by the NBVS planning algorithm. After each scan, the sensor readings are integrated into an octree that represents the object's bounding box. From the octree the occupied voxels that represent object surface are integrated with the current surface model via a registration process made possible by the overlap with the previous views. At the same time the robot is re-localized based on this registration step. Our planner assumes that the environment (including the obstacles) is known, except the object bounding box. However, as the robot discovers the object to be reconstructed, the proposed planner considers the new sensed information to avoid collisions with the object to be reconstructed and to find the next best view/state.

The NBVS phase starts by generating a set of view/state points in the configuration space which are ranked according to a utility function. The utility function combines four factors: (i) position, (ii) registration, (iii) surface, and (iv) distance. To perform this evaluation efficiently, it is done through several filters, so that the

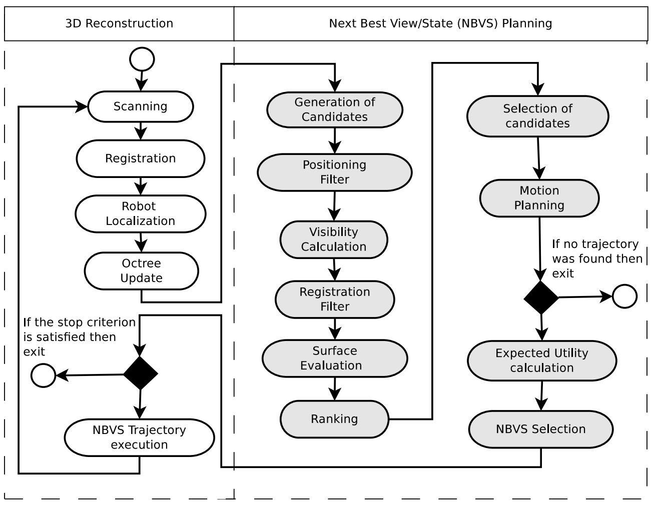


Fig. 2 3D object reconstruction with next best view/state planning. The diagram shows the whole process of object reconstruction. The processes related with the NBVS computation are filled in gray.

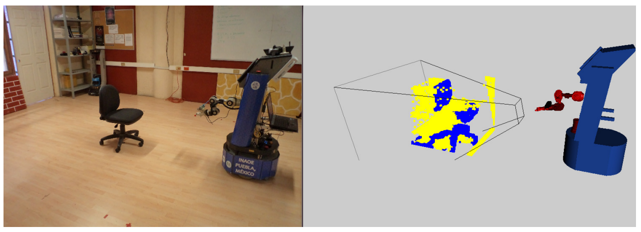


Fig. 3 Partial Model of the scene. To the left is the reconstruction scene. To the right is the partial model. The robot is represented by a triangular mesh. Unknown voxels are painted in yellow and occupied voxels are painted in blue (best seen in color).

candidate view/state that does not pass a filter is eliminated from the candidate set. The factors that consume less processing time are evaluated first.

First the positioning factor is evaluated, the candidates in collision are eliminated. Then the visibility of each view/state is calculated using an efficient scheme based on Hierarchical Ray Tracing (see section 5.4). To perform the registration process, only the candidates that guarantee a minimum overlap with previous views are maintained. The next step is to evaluate that amount of unknown voxels that are observed, this factor is normalized by the total number of remaining unknown voxels. For a reduced set of candidates a trajectory to each candidate state is planned using the Rapidly Exploring Random trees method. The candidate states are evaluated according to their distance to the current robot state considering the path followed by the robot. Then, the remaining candidates are ranked based on their expected utility. The expected utility is the expected value of the utility that a view/state will have under positioning error conditions. To determine the expected utility we generate several samples on the control space, based on the error distribution, then we transform those samples to the utility space and finally we compute the expected utility by associating a probability to each sample.

Finally the *best* candidate according to the expected utility is selected as NBVS. The 3D reconstruction–NBVS cycle is repeated until the surface factor is lower than a threshold (see SubSection 7.6), or no path was found for any of the candidate views/states.

5 Observation Model

In this section we describe the representation of the object bounding box, the integration of the sensor readings and the visibility calculation for a view/state.

5.1 Probabilistic Octree

To represent the object bounding box, W_{box}, we use a probabilistic occupancy map based on the octomap structure [5], which is an octree with probabilistic occupancy estimation. See Fig. 3. In this representation each voxel has associated a probability of being occupied. 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.

Similarly to the work presented in [13], in this work, we transform a sensed observation (a set of 3D points) to classes. For doing so we use as elementary unit the voxels. Depending on the probability of been occupied, we classify each voxel with one of three possible classes: i) occupied, which represents surface points measured by the range sensor, ii) free, which represents free space and iii) unknown, whose space has not been seen by the sensor. Each class has a defined probability interval. In our implementation the unknown voxel class has the interval $[0.45, 0.55]$. Class free has a probability less than 0.45 and class occupied has a probability larger than 0.55. One main adventage of defining these classes is that they allows us to know the amount of overlapped surface (voxels classified as occupied) between the new sensed surface and the partial model of the object. The amount of overlap is central to achieve a successful registration between the new data and the model of the object (see Section 7.1.)

5.2 Scanning

A scan recovers information of the workspace inside the sensor's frustum. Let $\mathcal{F}(v_x)$ denote the subset of the workspace that lies inside the sensor's frustum at the view v_x. Due to the fact that the sensor is attached to the end-effector of the robot, we simply specify the frustum with $\mathcal{F}(x)$. v_x is calculated by direct kinematics in order to get the pose of the sensor given a robot state [15]. We use a Denavit-Hartenberg model of our mobile manipulator robot to perform the direct kinematics computation.

The set of points that belongs to the object surface is denoted by S. $S = \{p | p = [x, y, z]^T\}$. Let S^i_{known} denote the subset of the object surface that becomes known in the i-th iteration [29]. The accumulated surface after k scans is denoted by $S_{known} = \bigcup_{i=1:k} S^i_{known}$. With respect to a sensing state, x, the surface of the object discovered is denoted by S^x_{known}.

5.3 Octree Update

Once a scan has been made, the sensor readings are integrated into the octree. Given an iteration i, for each point, p, of the measured surface points, S^i_{known}, a ray is traced between the sensor position and p. The occupancy probability, $p(n|z_{1:i})$, of a traveled voxel, n, is updated according to the octomap sensor fusion model [5].

5.4 Visibility Calculation

The visibility calculation is the simulation of a scan inside the octree. This simulation allows us to determine which type of voxels are visible for a given sensor pose.

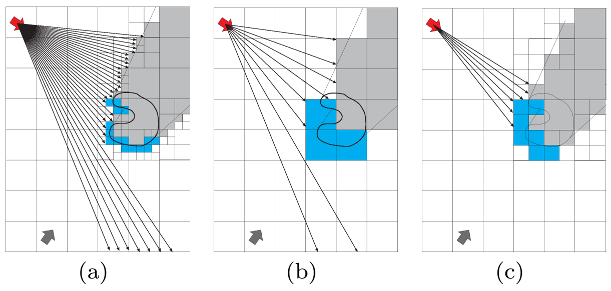


Fig. 4 Examples of uniform ray tracing and hierarchical ray tracing. (a) Uniform ray tracing. All sensor rays are traced in the octree. (b) Rays traced in a coarse octree. (c) Ray tracing in a finer octree only for touched voxels.

Therefore, it will help us to determine the goodness of a state (given the associated sensor pose). Usually, this task is achieved with a uniform ray tracing, it traces a number of rays inside the map simulating a range sensor (Fig. 4(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 [34].

In [34], we introduce 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. 4). 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 voxel size increases by a factor 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 an occupied voxels is touched. In this work, we increase the resolution when occupied and unknown voxels are touched. The advantage of refining the resolution for an unknown voxel is to discard empty part of the space (voxels classified as free). In this way the evaluation of the utility function will be more precise. Section 8.1.2 details several experiments where there is 60% of processing time reduction with only a loss of 1% of coverage.

6 Probabilistic Motion Model

A mobile manipulator has several sources of position error. In the mobile base, the error comes up usually from the slipping between the wheels and the ground; without a re-localization process, the error is accumulated and the pose uncertainty increases. The error sources of the arm are kinematics calibration, dynamic errors and position between links [19]; often a calibration step is performed and only residual errors are kept, namely, the uncertainty of the links positions with respect to the base does not grow. Given that the arm and sensor are attached to the base, the sensor pose is affected in position, and orientation and the error increments as the robot executes larger trajectories.

6.1 Control with Noise

The robotic base is controlled by linear and angular velocities. These velocities are not perfect. To model the imperfection of the velocities we use random variables with zero mean and variance σ^2. Thus, the linear and angular velocities are given by:

$$\hat{v} = v + \epsilon_{\sigma_v^2}$$

$$\hat{w} = w + \epsilon_{\sigma_w^2}$$

To obtain instances of this error we generate a random sample of the error with zero mean and variance σ^2. We take samples from a normal distribution. The imperfection of the robot motion depends on the values of σ_v^2 and σ_ω^2. Large values correspond to large errors.

$$\hat{v} = v + \texttt{sample}(\sigma_v^2)$$

$$\hat{\omega} = \omega + \texttt{sample}(\sigma_\omega^2)$$

Function `sample` generates a random sample of zero mean and variance σ^2.

To model errors in the motion of the robotic arm a similar approach is used, considering that the angular velocity of each link of the robotic arm is not perfect. Thus

$$\hat{\omega}_i = \omega_i + \texttt{sample}(\sigma_{\omega_i}^2)$$

Typically, the motion of the robot arm is more accurate than the robotic base. Thus, $\sigma_{\omega_i}^2$ will be smaller than σ_ω^2.

6.2 Numerical Integration

To simulate the uncertainty over the robot motion we use the Euler integration method over the robot state variables $(x, y, \theta_b, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$.

$$x_t = x_{t-1} + \hat{v}\cos(\theta_b)_{t-1}\Delta t$$

$$y_t = y_{t-1} + \hat{v}\sin(\theta_b)_{t-1}\Delta t$$

$$\theta_b = (\theta_b)_{t-1} + \hat{\omega_b}\Delta t$$

$$\theta_i = (\theta_i)_{t-1} + \hat{\omega_i}\Delta t$$

6.3 Probability of Reaching the Next State

The probability of reaching a next state after having applied a single control is given by

$$p(x_t|u_t, x_{t-1}) = p(\epsilon_{\sigma_v})p(\epsilon_{\omega_b})\prod_{i=1}^{n} p(\epsilon_{\omega_i}) \quad (2)$$

In general to reach a new state more than a single control is needed. Assuming independence of the controls over time, the probability of $p(x_t|u_{1:t}, x_{0:t-1})$ is given by:

$$p(x_t|u_{1:t}, x_{0:t-1}) = \prod_{i=1}^{t} p(x_i|u_i, x_{i-1}) \quad (3)$$

7 View/State Planning for Object Reconstruction

Object reconstruction is achieved by repeating the steps of scanning, registration, model update, next best view/state planning and execution of the calculated set of controls to reach the planned view/state.

To plan the next best view/state (NBVS), we sample the robot's state space and rank those samples with a utility function, so that the sample with the highest evaluation is selected as the NBVS. Our approach contrasts with related work where the candidates are generated in the workspace and inverse kinematics is required to 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). In addition, to generate samples in the state space avoids inverse kinematics calculation.

Below we describe the utility function, the expected utility calculation, an analysis of the number of samples required to guarantee collision free trajectories, and the efficient evaluation strategy to calculate the NBVS.

7.1 Utility function

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

$$g(x) = pos(x) \cdot reg(x) \cdot sur(x) \cdot dist(x) \quad (4)$$

where each factor evaluates a constraint, below we detail each constraint. The utility function is a multiplication because if any of the factors is zero then it is not a valid state and it is worthless to calculate the others.

7.1.1 Positioning

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

7.1.2 Registration

To register the new scan, previous works have proposed to assure a minimum amount of overlap [34] or consider all causes of failure [17]. 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 [18]). In this work, we propose a simple factor that is fast for evaluation, $reg(x)$. It is 1 if a minimum percent of overlap with previous surfaces exist, and 0 otherwise. See equation (5).

$$reg(x) = \begin{cases} 1 & \text{if } \dfrac{oc_o(x)}{oc_o(x) + un_o(x)} > h \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

where $oc_o(x)$ indicates the amount of occupied voxels that are touched by the sensor and lie inside W_{box}, $un_o(x)$ is the amount of unknown voxels in W_{box}, and h is a threshold. This factor allows us to evaluate a large amount of views efficiently and has been tested in the experiments with a real robot with good results.

7.1.3 New surface

$sur(x)$ evaluates a view/state depending on how much surface from the unknown volume is seen, i.e. the amount of visible unknown voxels. Such function returns values between 0 and 1. If no new information is obtained then the function returns zero and the maximal amount of information obtained corresponds to see the whole volume of the object, in that case the function returns one. See equation (6).

$$sur(x) = \frac{un_o(x)}{un_{total}} \quad (6)$$

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

7.1.4 Distance Factor

Candidate states are also evaluated according to their distance to the current robot state. The function is shown in eq. (7):

$$dist(x_n) = \frac{1}{1 + \rho(x_0, x_n)} \tag{7}$$

where ρ is the summation of the weighted Euclidean distance between the nodes of the path $P = \{x_0, x_1...x_n\}$ between the current robot state x_0 and the candidate state x_n, as defined in equation (8).

$$\rho(x_0, x_n) = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{m} w_j (x_i(a_j) - x_{i-1}(a_j))^2} \tag{8}$$

where $x(a_j)$ is the j-th degree of freedom of the state x, w_j is a weight assigned that degree of freedom (we have determined appropriated weights experimentally, see Section 8), and m is the number of degrees of freedom.

Unlike our previous approach, where a distance in the workspace was defined [36], 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.

Note that this measure is already normalized between zero and one, in the sense that if the robot does not move, the measure is one, and as the length of the robot motion increases, tending to infinity, the measure tends to zero.

7.2 Expected Utility

To deal with spatial uncertainty, we propose to evaluate a candidate view/state with the expected utility instead of a deterministic utility. The expected utility denotes the most likely utility of a view/state, x, when a given trajectory between the robot current state, x_c, and x is executed. In other words, the average utility when the robot moves to the indicated state many times. Formally, let the utility of a candidate view/state, x, be a random variable G_x, with probability distribution $p(g)$. The expected utility of x is defined by equation (9).

$$E(G_x) = \int_{-\infty}^{\infty} g \cdot p(g) dg \tag{9}$$

In the view planning problem, g depends on the shape, surface and size of the object plus the restrictions described in section 7.1, therefore its calculation is very expensive. In addition, the distribution over the utility is not trivial to determine, given that the error is modeled by a distribution over the control space, and such distribution has to pass through several non linear transformations, from the control space, $\mathcal{U}$, to the state space, $\mathcal{X}$, and then to the utility G. Given the previous reasons, we approximate equation (9) with equation (10).

$$E(G_x) \approx \sum_{i=1}^{k} g_i \cdot p(g_i) \tag{10}$$

where k is a number of samples, g_i the utility of the i-sample and $p(g_i)$ the probability of reaching that sample.

Next, we describe the expected utility calculation, summarized in algorithm 1. The idea is to compute the expected utility using several states around x whose distribution depends on the executed trajectory. The set of samples is denoted by $S = \{s_1, s_2...s_k\}$, $S \subset \mathcal{X}$. The algorithm generates the samples and for each sample, its utility and probability of being reached is computed. Finally, the probabilities are normalized and the expected utility is computed.

In detail, the algorithm generates k samples by simulating k times the execution of the trajectory as a stochastic process. For each simulation, the algorithm starts from the current state (line 2) and applies each control according to a motion model (line 4). The sample motion model, described by Thrun in [26], returns the next state given a current state and an applied control; however the applied control is perturbed with an stochastic error. Each generated state, x_t, is tested for collision (line 6), if one of them is in collision then the candidate state, x, is denoted as unfeasible (line 7). Marking the candidate as unfeasible when at least one generated state of the simulations is in collision is a conservative strategy that could be replaced by a weighted strategy. The last state of each simulation is taken as a sample (line 10). The utility of each sample is calculated using equation (4) (line 9). To compute the probability of occurrence of each sample, we assume independence of the errors over time, therefore the probability of each sample is calculated as the product of the probabilities of the occurrence of each intermediate state (line 10). Finally, we normalize the probabilities of the k samples and compute the expected utility (lines 11 and 12).

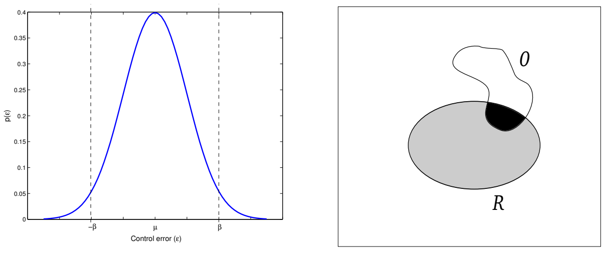
7.3 Analysis

In order to generate a sample state each control is perturbed with a stochastic error, see section 6. Sampling the error, ϵ, using all the domain of the distribution, $(-\infty, \infty)$, could generate a sample far away from the mean. This situation becomes important given that any sample that is in collision makes the candidate to be discarded (line 7 of the algorithm 1). To deal with

Algorithm 1: Expected utility. The algorithm calculates the expected utility, $E(x)$, of a candidate view/state x. It requires the candidate view/state, x, the current configuration, x_c, the sequence of controls $\tau = \{u_1, ...u_m\}$ that connects x_c with x and the number of samples, k.

```
   input  : $x$, $x_c$, $\tau$, $k$
   output: $E(x)$
1  for $i \leftarrow 1$ to $k$ do
2      $x_0 \leftarrow x_c$;
3      for $t \leftarrow 1$ to $m$ do
4          $x_t \leftarrow$ SampleMotionModel($u_t, x_{t-1}$) ;
5          $p_t \leftarrow p(x_t|x_{t-1}, u_t)$ ;
6          if $x_t \in \mathcal{X}_{obs}$ then
7              return Unfeasible;
8      $s_i \leftarrow x_m$;
9      $g_i \leftarrow g(s_i)$; // Utility of the sample
10     $P_i \leftarrow \prod_{t=1}^{m} p_t$; // Probability of the sample
11 $\eta \leftarrow \frac{1}{\sum_{i=1}^{k} P_i}$;
12 return $E(x) \leftarrow \sum_{i=1}^{k} g_i \cdot \eta P_i$;
```

this issue, we restrict the samples to a closed interval, $\epsilon \in [-\beta, \beta]$, see Fig. 5a. Considering the errors for a given control, we could form a hyper-volume R, where a sample could fall. See Fig. 5 b.



(a) Error distribution (b) Sampling region in $\mathcal{X}$

Fig. 5 Figure a) shows the limits applied to the error distribution during the sampling. Figure b) shows the hyper volume R which is the region in $\mathcal{X}$ where the samples could fall. R region could be intersected by O as shown in the figure. The size of R depends on the distribution of the error and the maximum values of the error, $[-\beta, \beta]$.

The hyper volume R does depend on both, the state transition equation and the sampling interval $[-\beta, \beta]$. However, we do not need to know the exact shape of R, since we estimate its volume by counting samples of final states generated through simulations. That is, we estimate the volume R, by simulating the robot trajectory using the state transition equation $\dot{x} = f(x, u)$, numerical integration is used to obtain the robot's trajectories. The final robot's states generating R are the states that use the imperfect controls having a bound over probabilistic density function given by the interval $[-\beta, \beta]$.

It is important to detect collisions inside R before sending the robot, this issue is related to how many samples are drawn. Below, we analyze how likely it is to detect a collision as the number of samples, k, increases. Let us assume that the obstacle region $\mathcal{O}$ intersects R. The probability that a sample does not hit the obstacle, event A, is equal to the probability of falling into R and not in O, namely $P(A) = P(\overline{O}|R)$, using the conditional probability definition:

$$P(A) = \frac{P(\overline{O} \cap R)}{P(R)} \tag{11}$$

given that O partitions R, $P(R) = P(O \cap R) + P(\overline{O} \cap R)$, we rewrite eq. (11) as:

$$P(A) = \frac{P(R) - P(O \cap R)}{P(R)} \tag{12}$$

by the distributive property,

$$P(A) = 1 - \frac{P(O \cap R)}{P(R)} \tag{13}$$

given that we know the sampling interval, $[-\beta, \beta]$, we make the probability $P(R) = P(-\beta \leq \epsilon \leq \beta)$, such probability depends on the error distribution. In our experiments, we assume a normal distribution and we sample ϵ inside the interval $[-3\sigma, 3\sigma]$. So, according to the "empirical rule" the probability of $P(-3\sigma \leq \epsilon \leq 3\sigma) = 0.997$. Once that we have identified how to calculate $P(R)$, we simplify the notation defining η as:

$$\eta = \frac{1}{P(R)} \tag{14}$$

On the other hand, $P(O \cap R)$ should be computed as the probability that a sample ϵ belongs to $[-\beta, \beta]$ and at the same time that ϵ belongs to an interval $\mathcal{E}_{obs}$ such that every $\epsilon \in \mathcal{E}_{obs}$ leads to a collision with the obstacle region. However, to determine analytically $\mathcal{E}_{obs}$ is not trivial. In consequence an approximation can be obtained through simulations.

For analysis purposes, let us assume $\gamma = P(O \cap R)$. Substituting η and γ, we rewrite equation (13) as:

$$P(A) = 1 - \eta\gamma \tag{15}$$

Now, assuming that each sample is drawn independently, the probability of drawing k samples and that not a single one hits the obstacle is:

$$P(A_k) = (1 - \eta\gamma)^k \tag{16}$$

Bounding (16) by an α, lead us to $(1 - \eta\gamma)^k \leq \alpha$, then, solving for k:

$$k \log(1 - \eta\gamma) \leq \log \alpha \tag{17}$$

$$k \geq \frac{\log \alpha}{\log(1 - \eta\gamma)} \tag{18}$$

Therefore, assuming certain probability of intersection, γ, and for a larger enough k, we can expect with certain $(1 - \alpha)$ probability that the real movement is collision free. For example, making $\alpha = 0.01$, $\eta = \frac{1}{0.997}$ and $\gamma = 0.05$ the number of samples should be more than 90 to be 0.99 sure that the movement is collision free.

In practice, we perform extensive off-line simulations to estimate the probability $\gamma = P(O \cap R)$. This is feasible, since the environment is known, including the object's bounding box.

Thus, we estimate the probability of collision inside R, that is, region O intersected with R, by counting the number of final states that are in collision divided by the total number of final states defining R.

7.4 Collision Free Trajectories with RRTs

The Rapidly Exploring Random Tree method (RRT) is a data structure and algorithm that is designed for efficiently searching non convex high-dimensional configuration or state spaces [16]. The RRT can be considered as a Monte Carlo way of biasing search into largest Voronoi regions. RRTs and related methods have been used in a variety of applications, for instance motion planning for humanoid robots [14], or to plan navigation routes for a Mars rover that take into account dynamics [30]. In [22], a sensor-based RRT, called SRT, is used for exploration of 2D environments. In [7], the authors have extended the RRT and other sampling-based motion planning algorithms to find optimal paths.

In this work, we adapt the RRT Ext-Ext [16] to plan robotic trajectories between the current robot state and the candidate view/states. The RRT Ext-Ext algorithm grows two balanced trees, one from the current state and one from the candidate state.

One advantage of using an RRT is that the resulting view/state will be at the same time collision free and will satisfy the sensing constraints (e.g. the sensor will be pointing to the object to be reconstructed, new surface sensed from that state will be present and the amount of overlapping surfaces that facilitates the registration process will be enough). Furthermore, such method directly gives us a set of controls to reach such state. In contrast, if the goal is given as a sensor pose, which will be reached by the robot using inverse kinematics, then depending on the robotic system geometry, it might happen that such sensor pose is not reachable.

7.5 Stop Criteria

The stop criterion decides when next best view planning task and the object's reconstruction are terminated. Previous works have proposed several criteria based on the partial model of the object [29,12]. However, to consider only the object is not enough, i.e., the robot state can be in collision with the environment or an obstacle may be blocking the way.

In this work, we have used two stopping criteria. In Subsection 8.1.4 we have used the number of scans as stopping criterion, in order to make a fair comparison with the Information Gain method. In the rest of the paper the reconstruction process is stopped if the surface factor is lower than a threshold, or no path was found for any of the candidate views/states. However, these criteria might be improved, for instance using a criterion similar to the one proposed in [3], in which a bound over the number of samples is established based on the probability that a sample does not provide new coverage. Nevertheless, we believe that the problem addressed in this work is more complex than the one presented in [3]. In this work the samples that would establish the stopping criterion lie in the configuration space rather than in the 3-D space. Note that in sampling-based motion planning (only considering the problem of finding collision free paths) if a solution exists then the probability of finding it tends to one as the number of samples tends to infinity. But, to our knowledge, if a solution does not exist then an ideal stopping criterion is unknown. In the problem addressed in this work, useful sampling configurations must be collision free and reachable by a trajectory under control errors and they must also provide new covered objects surface. Consequently, finding the ideal stopping criteria is not trivial and it deserves a careful analysis. We left this for future work.

7.6 Efficient Evaluation Strategy

In order to evaluate the candidate states efficiently, we perform the evaluation through several filters according to the utility function. If a candidate does not pass a filter then it is deleted from the candidate view/state set. The factors that consume less processing time are evaluated first, such that a time consuming factor is only evaluate when the configuration satisfies the others requirements.

1. **Generation of candidates**.A set of samples is generated by sampling the state space with a uniform distribution. If the director ray of the sensor intersects the object bounding box then the sample

is considered as a candidate. This step has constant complexity. (In our experiments to generate ten thousands candidates requires less than a second.)

2. **Positioning filter**. This filter checks the positioning factor of the utility function, $pos(x)$. Here, we evaluate whether or not the candidate x is collision free.
3. **Visibility calculation**. This process calculates the visibility of a state. In other words, it determines the amount of unknown and occupied voxels that are visible. These quantities will be used by the following filters.
4. **Registration filter**. This filter verifies if the candidates satisfy the registration filter, $reg(x)$, of the utility function.
5. **Surface evaluation**. This process evaluates the surface factor, $sur(x)$, of the utility function.
6. **Ranking**. The candidates are ranked depending on the evaluation provided by the surface factor.
7. **Selection of candidates.** In this step, the number of candidates is reduced to a small set, typically 25 samples. We made this restriction given that the following step is computationally expensive.
8. **Motion Planning** In this step, for each of the remaining candidates a trajectory from the current robot state is planned. We use a Rapidly-Exploring Random Tree (RRT) Ext-Ext [16] for each of the candidates.
9. **Expected utility calculation** So far we have several candidates that are collision free, guarantee an overlap, see unknown surface and have a collision free trajectory from the current robot state. Based on these factors a deterministic utility can be computed. In order to compute an expected utility, for each state candidate several trajectories considering control errors are generated. This is done using stochastic simulations, so that each state candidate has several uncertain trajectories in order to calculate the expected utility.

8 Experiments

In this section, we present a comprehensive set of experiments. Three groups of experiments are presented: simulation with perfect positioning (Section 8.1), simulations with uncertainty in positioning (Section 8.2) and real robot experiments (Section 8.4). The first group of experiments evaluates the processing time, the reconstruction of complex objects and includes a comparison with the information gain approach. The second group, simulations with uncertainty, shows the advantage of the expected utility approach in terms of a reduction of the collision rate. The third group of experiments show the behavior of the method in a real environment with obstacles.

The variables that we measure in the experiments are: i) the percentage of coverage, it is computed as the ratio of correspondent points over the total number of points in the ground truth model; a correspondent point is a ground truth point closer than a threshold (3 mm) to a built model point, and ii) the collision rate, calculated as the number of times that the robot collides divided by the total number of times that the robot executed a planned trajectory.

8.1 Simulations without motion uncertainty

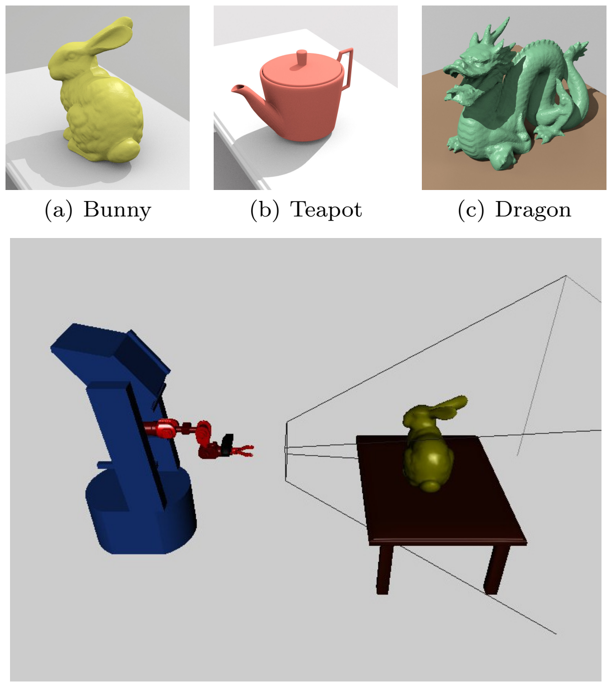
The following experiments evaluate the performance of the proposed NBVS planning method assuming perfect positioning. The first experiment measures the processing time required to calculate the visibility of a view. The second experiment measures the reconstruction coverage for different objects. The third experiment compares the proposed utility function against the Information Gain approach.

8.1.1 Scene configuration

The object to be modeled is set over a table in the reconstruction scene. The sensor is mounted on a mobile manipulator robot with eight degrees of freedom. See Fig. 3. Three different objects were reconstructed: the Stanford bunny, a teapot and a dragon. Range sensing was simulated using the Blensor Simulator (A Free Open Source Simulation Package for Light Detection, Ranging and Kinect sensors). Robot motion was simulated with our own simulator. The scenes can be downloaded from: *https://jivasquez.wordpress.com* The simulated sensor is a time of flight camera of 176 × 144 points. The parameters used in the simulation experiments are depicted in Table 1.

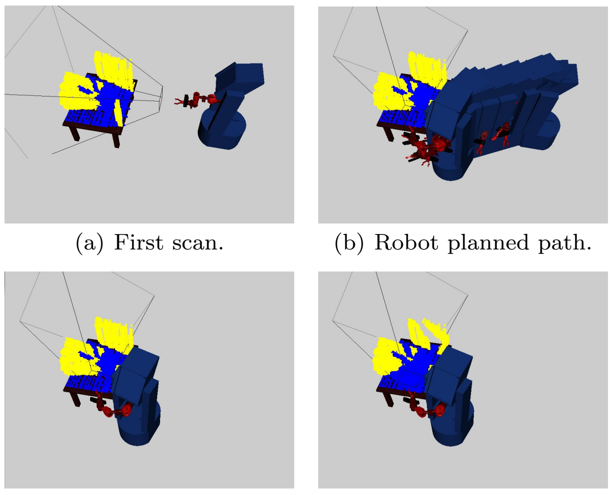
Parameter	**Value**
Number of state samples per iteration	10 000
Octree resolution	2 cm
Maximal number of nodes in the RRTs	10 000
Overlap threshold (h)	50 %
Distance weights (w_i)	[10.0, 10.0, 0.5, 0.05, 0.05, 0.05, 0.05, 0.05]
Machine	Intel core i5 microprocessor with 2Gb of RAM
Stop criteria	12 scans

Table 1 Parameters for the simulation experiments.


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

(d) Reconstruction Scene.

Fig. 6 Synthetic objects and reconstruction scene.


(a) First scan. (b) Robot planned path.

(c) Planned NBVS. (d) Second scan.

Fig. 7 Stages of the Bunny reconstruction. Unknown voxels are shown in yellow, occupied voxels are displayed in blue (best seen in color).

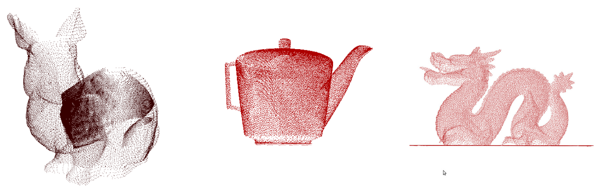


Fig. 8 Final representations (point clouds) from the reconstructed 3D synthetic objects.

8.1.2 Processing time of the Visibility Calculation

One of the most expensive aspects of the NBVS planning is to calculate the visibility of a view. The proposed Hierarchical Ray Tracing (HRT) reduces the visibility computation time, depending on the resolution parameter (section 5.4). We have tested different resolution parameters for the reconstruction of the Bunny object. The results are summarized in Table 2. 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 a time reduction, given that the overhead of the ray tracing structure increases the processing time.

In conclusion, HRT allows us to evaluate a large set of views in a short time, making it possible that even a naive set of random views could be useful to determine the NBVS. A drawback of this method is that there is no finer ray tracing for obstacles outside the bounding box. Therefore, the best performance of the HRT is obtained when the object has a clear space around it.

Table 2 View evaluation times using 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

8.1.3 Reconstruction of complex objects

In this experiment, the method is tested with different complex objects (the Bunny, the Dragon and the Teapot). We present quantitative results to evaluate the performance of the proposed approach. We use a resolution parameter $a = 2$.

Fig. 7 depicts several stages in the reconstruction of the Bunny. Fig. 8 shows the final representation of the reconstructed objects. Table 3 presents average results for the reconstruction of the objects in terms of the number of scans needed to reconstruct the 3D 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 view/state in order to reconstruct different complex objects with a large coverage in an acceptable processing time.

Table 3 Reconstruction results for each object.

Object	Scans	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 %

8.1.4 Proposed utility vs information gain

We estimate the goodness of a view measuring four factors: i)positioning, ii) overlap, iii) unknown surface (amount of unknown voxels) and iv) path distance. The factor of unknown surface is important given that it estimates how much surface could be discovered in the next scan. Another way to estimate how much surface could be discovered is computing the Information Gain (IG) of a scan [12]. In this experiment we compare the use of the surface factor in the proposed utility function versus the use of IG. The comparison was done by reconstructing the Bunny object 5 times using the surface factor in our proposed utility (that we will call deterministic utility, DU), and 5 times using information gain instead of the surface factor (we will call IG to this function). In order to make a fair comparison against the information gain method, the number of scans is used as stopping criterion in the experiments, both methods are stopped after 12 scans.

Figs. 9 and 10 show the average surface coverage and average unknown volume, respectively. In this experiment IG at initial iterations gets a higher coverage than DU, however at the final iterations both approaches converge to the same coverage.

In the IG approach all voxels inside the view frustum and belonging to the volume W_{unk} are considered to select a new sensing location, while in our approach only the voxels lying on the surface of W_{unk} and inside the frustum are considered. In our experiments we have observed that at the beginning of the reconstruction, larger percentage of the objects volume will appear, compared with the percentage of the object surface. The IG approach counts both the voxels truly belonging to the object plus the volume of the free space behind the object. This explains why the IG approach reports a larger percentage of coverage at the beginning of the reconstruction process. However, after several scans both utilities converge to the same coverage.

Processing time for evaluating both functions is quite similar. In our implementation, to evaluate DU for all candidates takes an average time of 428 s., in contrast, IG factor takes 484 s., that is 13% more than DU. It is worth to say that the evaluation processing time of DU can be significant reduced using the hierarchical ray tracing [34], as demonstrated in section 8.1.2.

As it was already mentioned, in IG approach all voxels inside the view frustum are used to compute the information gain, while using the HRT only voxels classified as occupied inside the view frustum and belonging to the surface of the object are considered to compute the progress in the reconstruction process. An advantage of this is that voxels classified as free do not need to be further refined. Thus, the benefit of applying the HRT is that the detection of voxels belonging to the object surface will be done at a finer resolution but in a shorter time since free voxels do not need to be refined. It might be possible to adapt the IG approach to use the HRT method, but this will require further research.

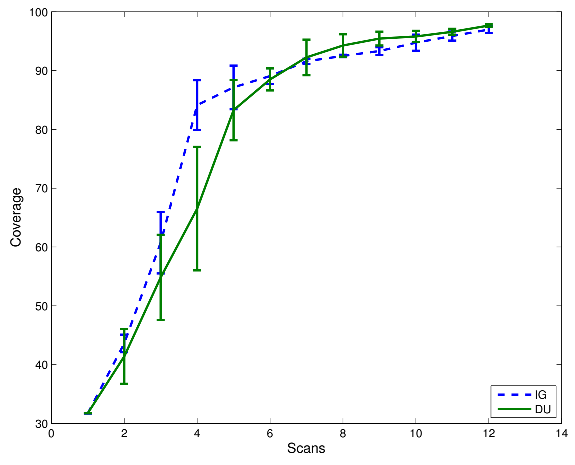


Fig. 9 Comparison of the average surface coverage using information gain (IG) and the proposed utility (DU). Vertical lines show the standard deviation. Both methods converge to the same coverage.

8.2 Simulations under motion uncertainty

This experiment analyzes the expected utility method. Two experiments are presented. The first experiment analyzes the effect of the number of samples (k) over the collision rate. The second experiment compares the

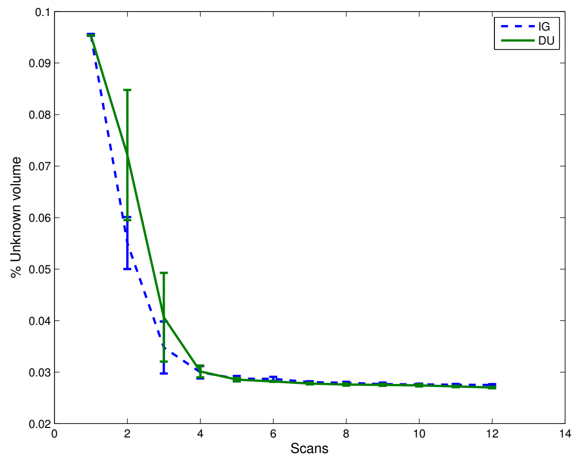


Fig. 10 Comparison of the unknown volume in the octree using information gain (IG) and the proposed utility (DU). Both methods leave the same unknown volume in the octree.

expected utility versus the deterministic utility under different conditions of motion uncertainty.

8.2.1 Configuration

Imperfect controls are modeled using two parameters, σ_v and σ_w, which correspond, respectively, to a standard deviation of an error on the linear and angular velocities. Larger values correspond to more imprecise motions. The values of σ_v and σ_w were computed by statistical modeling using a real robot. 70 repetitions were done to compute the error in terms of angular velocity ω and 70 others to obtain the error in terms of linear velocity. The error on the angular velocity was computed based on the difference between a perfect rotation in place vs the measured executed one. Similarly, the error on the linear velocity was computed based on the difference between a perfect straight line translation vs the measured executed one. Table 4 shows the standard deviation of both the linear and angular velocities. We assume perfect positioning of the arm therefore standard deviations of the arm joints are zero.

Parameter	Value
σ_v	1.65 mm/seg
σ_w	0.28 rad/seg

Table 4 Parameters of the motion model

The compared approaches are detailed below:

- Deterministic utility without motion uncertainty (DU). This is the ideal case, and it corresponds to the best possible performance.
- Deterministic utility with motion uncertainty (DU-MU). We expect to significantly improve this case using expected utility. The model of the imperfect controls is described in Section 6. The parameters σ_v and σ_w correspond to the ones shown in Table 4.
- Expected utility with motion uncertainty (EU-MU). Here, the same error model is used but the next best view/state is determined using expected utility.

8.2.2 Number of samples k

This experiment shows the effect of the number of samples k over the collision rate. Fig. 11 shows that for EU-MU the collision rate approximates zero as k increases. The graph suggests that the collision rate exponentially converges to zero as k increases, a expected behavior that matches with the analysis presented in section 7.3. The graph is only for the Bunny object but the same behavior was observed for the other two objects. An observed phenomenon of this experiment is that even for a single sample the proposed expected utility has a lower collision rate. This phenomenon is due to the fact that motion planner returns a set of controls that take the robot very close to a goal state, but does not reach it. For example, a bidirectional RRT has a gap where the trees are connected. To simulate the robot movement produces a better approximation than assuming that the robot reaches the goal state.

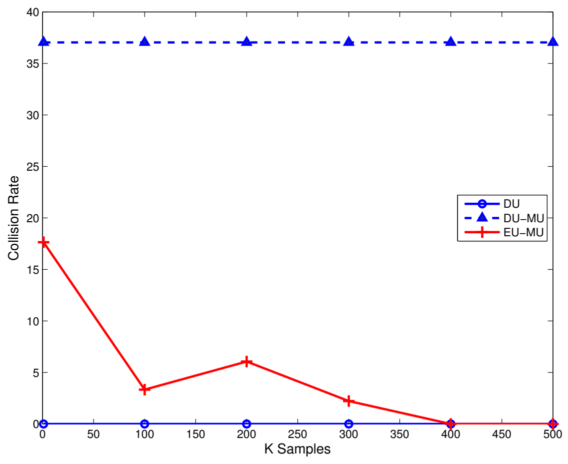


Fig. 11 The figure shows the behavior of the collision rate as the number of samples, k, increases in the reconstruction of the Bunny object. As the number of samples increases the collision rate decreases. When k is larger than 400 the collision rate is zero. DU and DU-MU collision rates are displayed to compare the performance of EU-MU; they are constant given that the number of samples does not interfere with them.

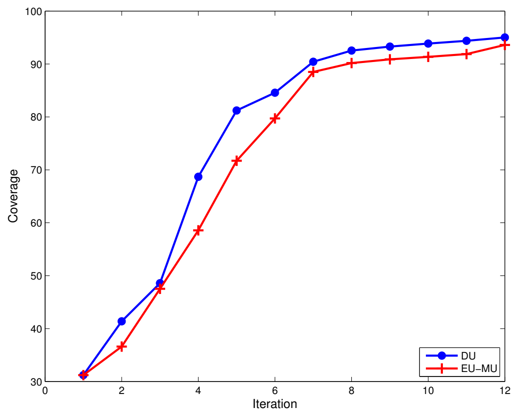
8.2.3 Comparison versus a deterministic utility

In this experiment, we compare the proposed expected utility versus the deterministic utility. In a simulated environment, we reconstruct ten times the three different objects under different conditions of motion uncertainty. We set $k = 500$.

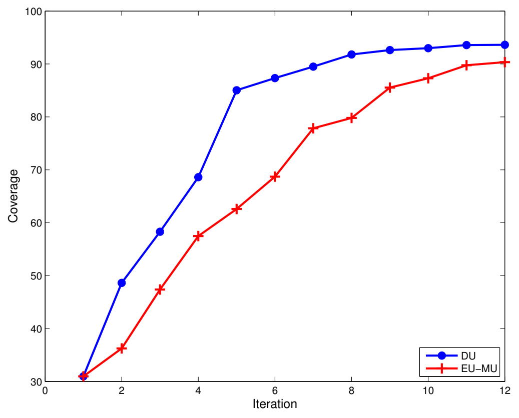
Table 5 compares the reconstruction coverage and collision rate for each object. In general, the results show that expected utility successfully decreases the collision rate and increases the reconstruction coverage. The coverage of expected utility keeps closer to the topline coverage. Fig. 12 shows the mean coverage for the Bunny and Teapot objects.

Object	Approach	Mean Coverage	Max. Coverage	Collision Rate
Bunny	DU	95.0	95.0	0 %
	DU-MU	48.2	87.4	37 %
	EU-MU	78.9	95.0	0 %
Teapot	DU	93.6	94.5	0 %
	DU-MU	41.3	68.2	60 %
	EU-MU	86.1	92.9	0 %
Dragon	DU	87.5	87.8	0 %
	DU-MU	36.1	55.6	45 %
	EU-MU	81.0	87.5	0 %

Table 5 Coverage results for each approach. EU-MU increases the reconstruction coverage and decreases the collision rate.



(a) Bunny



(b) Teapot

Fig. 12 The coverage of the expected utility approach keeps very close to the case where there is no motion uncertainty. At the end, EU-MU is less than 5% below than DU. DU-MU was not displayed given that not a single reconstruction reached 12 scans.

8.2.4 Cluttered Environment

In this experiment we test the method in an environment with many obstacles. This scenario is challenging for a robot that has imperfect controls, given that it has to avoid collision with the obstacles while it reconstructs the object. We compare the proposed expected utility versus a method that uses a deterministic utility. In both cases the robot has the same noise over the controls (see section 8.2). Fig. 13 shows the environment where the robot moves and a path generated by the planner.

Table 6 shows results of the reconstruction of an object in an environment with several obstacles, comparing the deterministic utility vs the proposed expected utility. The results are the average of five repetitions of the experiment. The processing time is the average time for the whole next best view/state determination. The simulations were executed using a Core i3 machine with 2GB of RAM. The parameters of the experiment are the same to the parameters of the real robot experiments (see Table 7).

Performance metrics	Deterministic Utility	Expected Utility
Number of sensing scans	2.0	12.6
Coverage	38.4 %	73.64 %
Remaining unknown volume	0.085 m^3	0.023 m^{3}
Traveled distance per scan	6.15	4.67
Collision rate	50 %	0 %
Processing time of the planning process	250 s	467 s

Table 6 Results of the reconstruction of of the Bunny object in an environment with several obstacles.

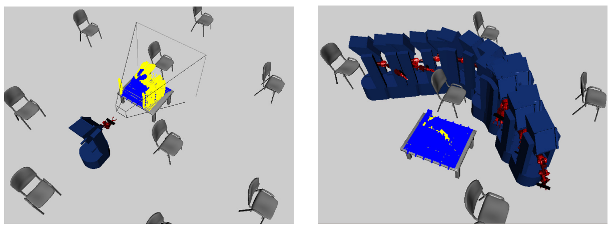


Fig. 13 Object reconstruction in an environment with several obstacles. In this experiment the robot has to avoid collisions with the obstacles while it reconstructs the target object. The left figure shows the environment where the robot moves. The obstacles are a table with the object to be reconstructed and nine chairs placed around the table. The right figure shows a path generated by the planner to reach the next best view/state.

The expected utility method is able to deal with a cluttered environment. The collision rate is zero and the surface coverage is 73%. In contrast, the collision rate for the deterministic utility is very high (50%). This high collision rate is due to the fact that the method is trying to maximize the amount of coveraged surface, but often the views that cover more surface are far from the current configuration and they are more susceptible to collide during the robot's motion. The processing time increases because it is more difficult to find a collision free path with many obstacles.

These simulation experiments show that the expected utility significantly reduces the collision rate and increased the object's coverage.

8.3 Analysis of simulations results

Based on the simulations results, we conclude that: (1) the proposed method can reconstruct complex objects with a high coverage in a reasonable time. (2) The proposed method achieves the same coverage compared to information gain but requires less processing time thanks to the hierarchical ray tracing. However, it might be possible to adapt the IG approach to use the HRT method, this issue is left for future work. (3) The proposed approach deals effectively with the uncertainty in the controls of a mobile manipulator robot with 8 DOF based on the concept of expected utility. The expected utility significantly reduces the collision rate and increased the object's coverage.

8.4 Real Robot Experiments

In this section, we present three experiments with a real robot. The first one has the objective to compare the deterministic utility vs expected utility. The second experiment has the objective of showing that the proposed method is able to correctly work in an environment with obstacles. The third experiment shows that the method is able to reconstruct complex objects.

All our experiments were done using the following hardware. The mobile base is a PatrolBot robot, the arm is Katana 180 6M robot, which is mounted on the mobile base. A Kinect sensor is placed in the arm end effector. Planning, octree update and registration are done in a laptop equipped with a Intel core i5 microprocessor with 2Gb of RAM. Table 7 presents the parameters used in the experiments. The octree resolution (3 cm) is determined by the Kinect resolution. We have selected such size of 3 cm for the voxels at the finest resolution for allowing the voxels to be larger than the error of the Kinect sensor. In this way, different sensor readings of the same point will statistically correspond to the same voxel. We have considered 3 cm as an upper bound of the error considering that the distance between the sensor and the object to be reconstructed would be less than 2 meters.

The remaining parameters were set taking into account our analysis and the simulation results, however they were calibrated according to the scene in order to get successful reconstructions. The process to find the parameters was to iteratively increase the value calculated in simulation, until the robot was able to successfully reconstruct the objects.

Parameter	Value
State samples per iteration	100 000
Maximal number of nodes in the RRTs	10 000
Expected utility samples (k)	500
Minimal overlap threshold (h)	65 %
Distance weights (w_i)	[10.0, 10.0, 0.5, 0.05, 0.05, 0.05, 0.05, 0.05]
Octree resolution	3 cm
Stop criteria	30 unknown voxels

Table 7 Parameters for the real robot experiments.

8.4.1 Experiment # 1: Chair: Utility vs Expected Utility

This experiment performs the reconstruction of an office chair using the mobile manipulator robot. Fig. 3 (in Section 3) shows the experimental setup. As mentioned above, the objective of this experiment is to compare the deterministic utility function presented in Subsection 7.1 vs. the expected utility described in Subsection

7.2. In this experiment, the object is reconstructed three times. In each reconstruction, each time that the robot collides the experiment is terminated.

The results from Experiment # 1 are summarized in Table 8. The performance metrics used in the comparison are the statistical mean of the following elements: number of sensing scans, unknown remaining volume of the object at the end of the reconstruction process, traveled distance in the configuration space computed using Eq. 8, number of nodes in the robot's path, collision rate, processing time per iteration of the planning process, and processing time per iteration of the registration process and octree update.

Performance metric	Deterministic Utility	Expected Utility
Number of sensing scans	4	14
Unknown volume	0.07 m^3	0.018 m^3
Traveled distance	28.7	72.52
Number of nodes in the robot's path	70.5	56.8
Collision rate	15.38 %	2.04 %
Processing time of the planning process	24.73 s	175.74 s
Processing time of the registration process and octree update	5.77 s	4.82 s

Table 8 Results of Experiment # 1 comparing the deterministic utility vs. the expected utility

The performance metrics shown in Table 8 allows one to observe that the main advantage of the expected utility over the deterministic one is that the collision rate is significantly reduced. In this experiment the collision rate with deterministic utility was 15.38% while the collision rate was only 2.04% with expected utility. Other advantage of expected utility is that the percentage of remaining unknown volume is also smaller. A disadvantage of the expected utility is that the processing time per iteration of the planning process is larger compared with deterministic utility. This is because it is based on simulations of the robot trajectories and sensing states. In this experiment the processing time was almost seven times larger. However, the processing time remains in the order of few minutes. Furthermore, the processing time can be improved with better hardware and implementing some computations in parallel. Note that the expected utility can be computed in parallel, using a different processor for generating each simulated robot trajectory yielding a final state sample.

The smaller distance traveled and the smaller number of sensing scans related to deterministic utility is explained because each time that robot collides the experiment is finished. Hence, with deterministic utility the robot rarely reaches the threshold in terms of unknown voxels to stop the reconstruction process.

Fig. 14 shows the robot taking a scan of the object during the experiment. Fig. 15 depicts the object and the point cloud model. Fig. 16 presents the voxels-based object model after finishing the reconstruction process. Fig. 17 shows two examples of planned paths and the ones computed with an odometer, and fig. 18 shows an example of the registration process.

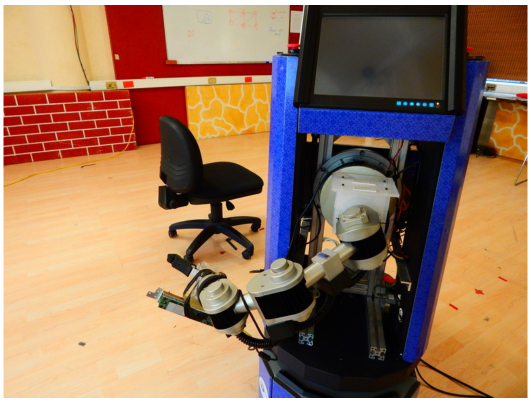


Fig. 14 Example of a view/state.

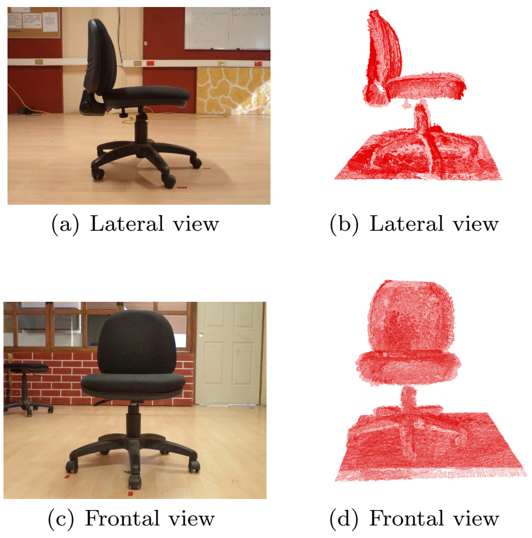
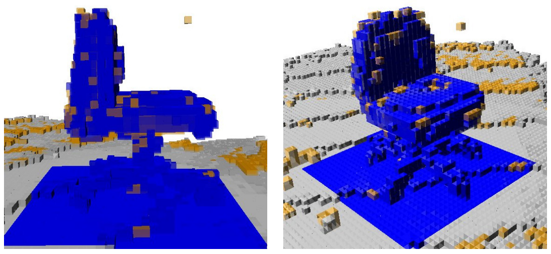


Fig. 15 Real object and point cloud model.

8.4.2 Experiment # 2: Object Reconstruction in an Environment with Obstacles

The objective of this experiment is to show that the method is capable of reconstructing an object in an



(a) Lateral view (b) Other view

Fig. 16 Final voxel representation.

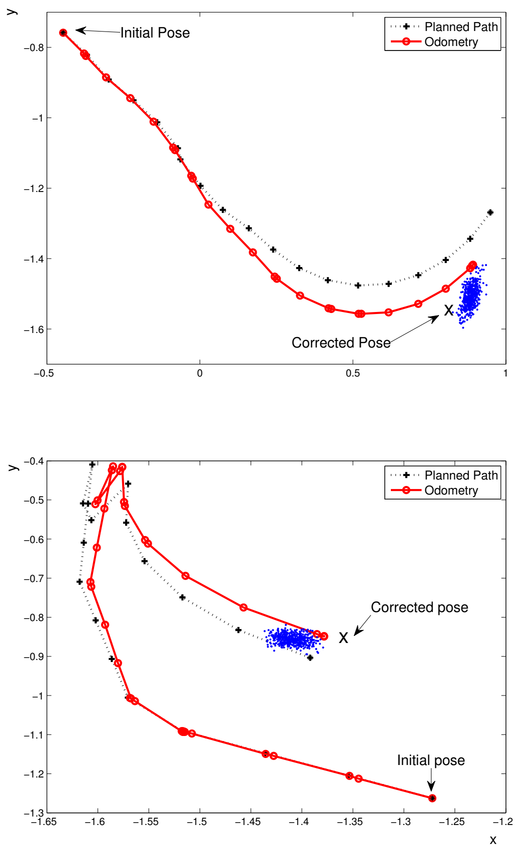


Fig. 17 Examples of robot's paths (best seen in color). The figures compare the planned path (black dotted line) versus the executed path (red continuous line) registered by the odometer. The blue dots represent the samples used to compute the expected utility. The big cross indicates the corrected robot position computed based on the registration of scans.

environment with obstacles. The suitcase to be reconstructed is in the center of the scene. The robot and the obstacles are around the suitcase. See Fig. 19. The output model of the reconstruction is shown in Fig. 20. Fig. 21 depicts the sequence of planned view/states to reconstruct the object. In this experiment the method

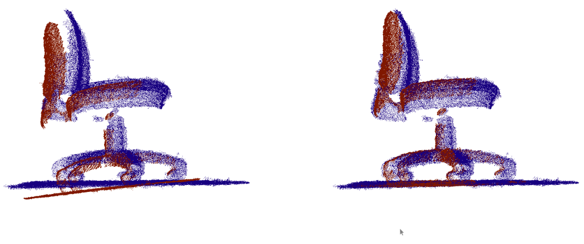
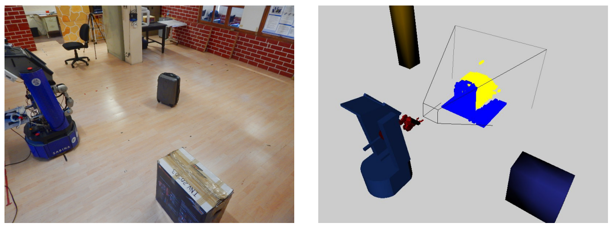


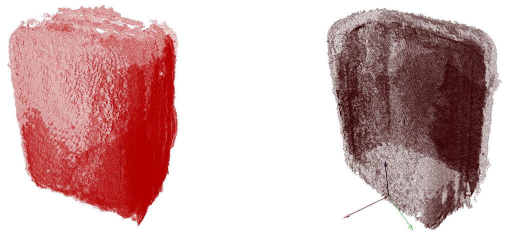
Fig. 18 Example of the point cloud registration. The figure in the left shows the data before registration and the figure in the right the data after registration.



(a) Reconstruction scene. (b) Scene representation.

Fig. 19 Reconstruction scene with obstacles.

was capable to plan each view/state avoiding collisions with the environment and the object.



(a) Front view (b) Rear view

Fig. 20 Accumulated point cloud of the object.

8.4.3 Experiment # 3: Reconstruction of a more complex object

This experiment shows the reconstruction of a more complex object (a NAO robot). The object was placed over a box and both were reconstructed. Fig. 22 shows the object and the reconstructed point cloud. Seventeen scans were required to reach the stop criteria. Fig. 23 shows each view/state where a scan was taken. The mean time for calculating each view/state was 118.6 seconds (using the expected utility approach). The collision rate was zero. This experiment demonstrates that our method can reconstruct a complex object, avoid collisions, and obtain a reasonable 3-D model.

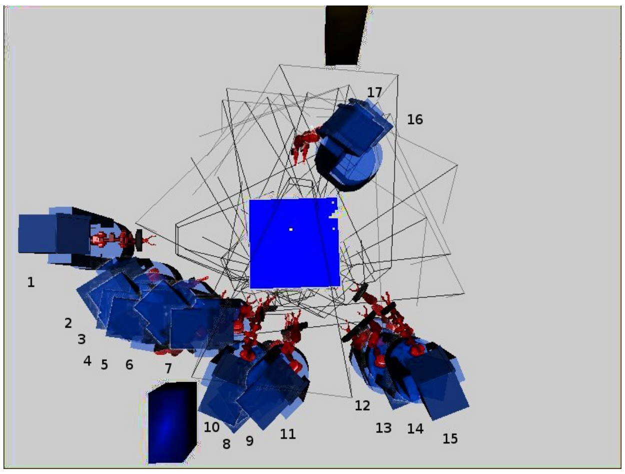


Fig. 21 Sequence of robot view/states from where a scan was done. The sequence shows that robot evades the object and completes the reconstruction.

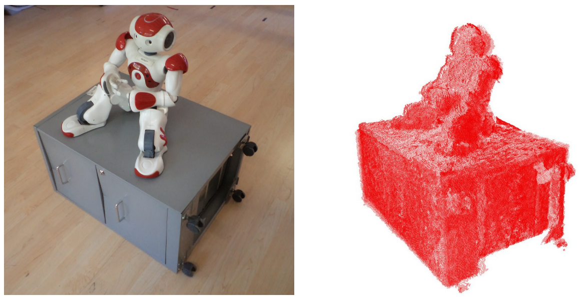


Fig. 22 Reconstruction of a NAO robot. Left figure shows the real object. Right figure shows the reconstructed point cloud.

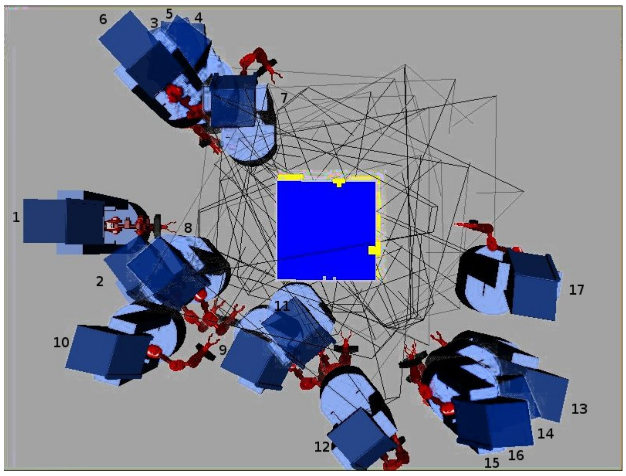


Fig. 23 Sequence of Robot view/states during the reconstruction of the NAO robot. In this sequence, the robot moves itself to the left (upper side of the figure) for scans 3-7. Then, the robot continues the reconstruction of the object to the right (lower side of the figure).

8.5 Analysis of experimental results

The experiments empirically validate the method, evaluate its performance along various metrics, and demonstrate its ability to perform well in different settings. In particular, we have shown that expected utility significantly reduces the collision rate (in our experiments the collision rate was reduced to 2.04 %). The experiments also show that the method works correctly in an environment with obstacles. We have also demonstrated that the method is able to reconstruct complex 3-D objects in a realistic environment with a mobile manipulator robot.

9 Conclusions and Future Work

We have presented a method for next best view/state planning for 3D object reconstruction. This is one of the first methods that determines the view directly in the state space, following a methodology in which a set of candidate view/states is directly generated in the state 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. Our approach is able to deal with motion and observation uncertainty. Another contribution of the approach is the evaluation of a candidate state in terms of its expected utility, unlike previous approaches where perfect positioning is assumed and only a deterministic utility is considered. Our approach plans safe robot states, in terms of collision between the robot and the environment. An analysis of the expected utility algorithm was provided and the behavior matches with the experimental results in the real robot.

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, performing an efficient evaluation of the candidate views. Quantitatively, this method achieves the same coverage but with smaller processing time compared with previous works. We have implemented the whole method both in simulation and in a real mobile manipulator of 8 DOF with an eye-in-hand sensor. The results show that the approach effectively increases the object coverage and also decreases the rate of robot's collisions. To our knowledge, this is one of the first works in which a method to reconstruct a 3D object is implemented in a real mobile manipulator robot.

As future work, we would like to use the probabilities in the octree representation of the space to estimate the goodness of the next sensing scan. We think that a promising modeling technique for planning the task of object reconstruction is a partially observable Markov decision process (POMDP) that allows one to take into account, both, the uncertainty of reaching the state and the uncertainty in the observations.

We would also like to remove the assumption that the size and approximate location of the object to be reconstructed is known. A possible way of reaching this objective is roughly segmenting the object from the background using RGB and range data. However, such approach might generate non trivial issues and would have to be tested experimentally.

Acknowledgement: I. Vasquez-Gomez was supported by a Conacyt PhD. scholarship, he thanks Luis Valentín, Rigoberto López and David Carrillo for their help in the development of the robotic system. R. Murrieta-Cid was supported in part by Conacyt grant No. 220796.

References

1. J. Aleotti, D. L. Rizzini, and S. Caselli. Perception and grasping of object parts from active robot exploration. *Journal of Intelligent and Robotic Systems*, 76(3-4):401–425, 2014.
2. C.I. Connolly. The determination of next best views. In *Proc. IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 432–435, St. Louis, MO, USA, 1985.
3. J. Espinoza, A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson. A motion planning strategy for finding an object with a mobile manipulator in 3-d environments. *Advanced Robotics*, 25(13-14):1627–1650, 2011.
4. 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 *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1078–1083, Kobe, Japan, 2009.
5. A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: an efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.
6. Y. Huang and K. Gupta. Collision-probability constrained prm for a manipulator with base pose uncertainty. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1426–1432, St. Louis, MO, USA, Oct 2009.
7. S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, June 2011.
8. L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, Aug 1996.
9. G. Kewlani, G. Ishigami, and K. Iagnemma. Stochastic mobility-based path planning in uncertain environments. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1183–1189, St. Louis, MO, USA, 2009.
10. S. Khalfaoui, R. Seulin, Y. D. Fougerolle, and D. Fofi. An efficient method for fully automatic 3d digitization of unknown objects. *Computers in Industry*, 64(9):1152–1160, 2013.
11. M. Krainin, B. Curless, and D. Fox. Autonomous generation of complete 3d object models using next best view manipulation planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 5031–5037, Shanghai, China, 2011.
12. S. Kriegel, C. Rink, T. Bodenmuller, A. Narr, M. Suppa, and G. Hirzinger. Next-best-scan planning for autonomous 3d modeling. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2850–2856, Vilamoura, Algarve, Portugal, 2012.
13. S. Kriegel, C. Rink, T. Bodenmller, and M. Suppa. Efficient next-best-scan planning for autonomous 3d surface reconstruction of unknown objects. *Journal of Real-Time Image Processing*, pages 1–21, 2013.
14. J. J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Motion planning for humanoid robots. In *Proc. The Eleventh International Symposium on Robotics Research*, pages 365–374, Siena, Italy, 2003.
15. S. M. LaValle. *Planning Algorithms.* Cambridge University Press, 2006.
16. S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.
17. K. Low and A. Lastra. An adaptive hierarchical next-best-view algorithm for 3d reconstruction of indoor scenes. In *Proc. 14th Pacific Conference on Computer Graphics and Applications*, Taipei, Taiwan, 2006.
18. K. Low and A. Lastra. Predetermination of icp registration errors and its application to view planning. In *Proc. Int. Conf. on 3D Digital Imaging and Modeling*, pages 73–80, Montreal, Quebec, Canada, 2007.
19. C. Lu. *The Error Propagation in Robots.* PhD thesis, University of Ottawa, 1989.
20. E. Marchand and F. Chaumette. An autonomous active vision system for complete and accurate 3d scene reconstruction. *The International Journal of Robotics Research*, 32(3):171–194, 1999.
21. N.A Melchior and R. Simmons. Particle rrt for path planning with uncertainty. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1617–1624, Roma, Italy, 2007.
22. G. Oriolo, M. Vendittelli, L. Freda, and G. Troso. The srt method: Randomized strategies for exploration. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 4688–4694, Seoul, Korea, 2001.
23. C. Potthast and G. Sukhatme. A probabilistic framework for next best view estimation in a cluttered environment. *J. Vis. Comun. Image Represent.*, 25(1):148–164, 2014.
24. W. R. Scott, G. Roth, and J.-F. Rivest. Pose error effects on range sensing. Technical report, NRC Institute for Information Technology; National Research Council Canada, 2002.
25. W. R. Scott, G. Roth, and J.-F. Rivest. View planning for automated three-dimensional object reconstruction and inspection. *ACM Comput. Surv.*, 35:64–96, March 2003.
26. S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics.* The MIT Press, 2005.
27. L. Torabi. *Integrated View and Path Planning for a Fully Autonomous Mobile-Manipulator System for 3D Object Modeling.* PhD thesis, Simon Fraser University, 2012.
28. L. Torabi and K. Gupta. An autonomous 9-dof mobile-manipulator system for in situ 3d object modeling. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and*

Systems, pages 4540–4541, Vilamoura, Algarve, Portugal, 2012.

29. 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*, 31(1):82–100, 2012.
30. C. Urmson and R. Simmons. Approaches for heuristically biasing rrt growth. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1178–1183, Las Vegas, Nevada, USA, 2003.
31. J. van den Berg, P. Abbeel, and K. Goldberg. Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. In *Proc. Robotics: Science and Systems*, Zaragoza, Spain, 2010.
32. J. van den Berg, S. Patil, and R. Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research*, 31(11):1263–1278, 2012.
33. J. I. Vasquez, E. Lopez-Damian, and L. E. Sucar. View Planning for 3D Object Reconstruction. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4015–4020, St. Louis, MO, USA, 2009.
34. J. I. Vasquez, L. E. Sucar, and R. Murrieta-Cid. Hierarchical Ray Tracing for Fast Volumetric Next-Best-View Planning. In *Proc. Tenth Conf. on Computer and Robot Vision*, pages 181–187, Regina, Saskatchewan, Canada, 2013.
35. J. I. Vasquez, L. E. Sucar, and R. Murrieta-Cid. View planning for 3d object reconstruction with a mobile manipulator robot. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4227–4233, Chicago, IL, USA, 2014.
36. J. I. Vasquez-Gomez, L. E. Sucar, R. Murrieta-Cid, and E. Lopez-Damian. Volumetric next-best-view planning for 3d object reconstruction with positioning error. *Int. J. Adv. Robot. Systems*, 11(159):1–13, 2014.