Intelligent Service Robotics (2019) 12:103–123
https://doi.org/10.1007/s11370-018-0264-y

ORIGINAL RESEARCH PAPER

Optimal motion planning and stopping test for 3-D object reconstruction

Heikel Yervilla-Herrera[1] · J. Irving Vasquez-Gomez[2] · Rafael Murrieta-Cid[1] · Israel Becerra[1] · L. Enrique Sucar[3]

Received: 25 March 2018 / Accepted: 21 September 2018 / Published online: 11 October 2018

Abstract

In this work, two aspects of motion planning for object reconstruction are investigated. First, the effect of using a sampling-based optimal motion planning technique to move a mobile manipulator robot with 8 degrees of freedom, during the reconstruction process, in terms of several performance criteria is studied. Based on those criteria, the results of the reconstruction task using rapidly exploring random tree (RRT) approaches are compared, more specifically RRT* smart versus RRT* versus standard RRT. Second, the problem of defining a convenient stopping probabilistic test to terminate the reconstruction process is addressed. Based on our results, it is concluded that the use of a RRT* improves the measured performance criteria compared with a standard RRT. The simulation experiments show that the proposed stopping test is adequate. It stops the reconstruction process when all the portions of object that are possible to be seen have been covered with the field of view of the sensor.

Keywords Optimal motion planning · Object reconstruction · Termination test

1 Introduction

This work is placed in the context of view planning, which is the task of determining what is the best position or configuration for a sensor in order to inspect an object or scene. In robotics, view planning has become a core problem since the robots need to interact with the uncertainty of a dynamic environment's structure. For example, an unmanned aerial vehicle is exploring a disaster scene or a service robot needs to update the 3D map representation when new furniture is added to the house.

In previous work [23], the authors have presented a method for next best view/state planning for 3D object reconstruction. The proposed method 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. A utility function that integrates several relevant aspects of the problem and an efficient strategy to evaluate the candidate views have been proposed. The proposed approach is able to deal with motion and observation uncertainty by considering an expected utility. The behavior of the proposed modeling has matched with the experimental results in a real robot.

This work was partially funded by CONACYT project 220796.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s11370-018-0264-y) contains supplementary material, which is available to authorized users.

✉ Rafael Murrieta-Cid
murrieta@cimat.mx

Heikel Yervilla-Herrera
yervilla.herrera@cimat.mx

J. Irving Vasquez-Gomez
jivasquezg@conacyt.mx

Israel Becerra
israelb@cimat.mx

L. Enrique Sucar
esucar@inaoe.mx

[1] Centro de Investigación en Matemáticas (CIMAT), Guanajuato, Mexico

[2] Consejo Nacional de Ciencia y Tecnología (CONACYT) - Instituto Politécnico Nacional (IPN), Mexico City, Mexico

[3] Instituto Nacional de Astrofísica Óptica y Electrónica (INAOE), San Andrés Cholula, Puebla, Mexico

In this work, two new aspects of motion planning for object reconstruction have been addressed, which have not been investigated before. First, the effects of using optimal motion planning techniques versus using a standard one on an object reconstruction task are investigated. It is important to have in mind that this paper is not about a comparison between the algorithms by themselves; it is a comparison of the effects produced by the paths generated by those algorithms produce in an object reconstruction task. Typically, the optimal motion planning must deliver shorter trajectories compared with the ones obtained with a technique that does not minimize the length of the trajectory. But, are there other consequences in the reconstruction process? For instance, what is the resulting positioning error? Or does optimal trajectories (in terms of the length) affect the percentage of reconstruction and the quality of the obtained model? Thus, the effect of using a sampling-based optimal motion planning technique [6,12,13] to move a mobile manipulator robot with 8 degrees of freedom, during the reconstruction process, in terms of several performance criteria is investigated. These performance criteria are organized in two types: one is related to the resulting trajectories to reach sub-goals (sensing locations): (i) the cost of the trajectories under the effect of noise, (ii) the positioning error to reach a sub-goal and (iii) the rate of collisions. Other criteria are related to the quality of the reconstructed model: (iv) the percentage of reconstruction, (v) the distance from each 3D point in the reconstructed model to the closest point in a ground truth model and (vi) the average density of 3D points per voxel in the reconstructed model. Finally, the processing time to reconstruct the object is measured. Based on those criteria, we compare the results of the reconstruction task using rapidly exploring random trees (RRT) approaches, more specifically RRT* smart versus RRT* versus standard RRT. Second, the problem of defining a convenient stopping probabilistic test to terminate the reconstruction process is investigated. It is worth mentioning that the used framework, including motion model, observation model, next best view selection and stopping test, is flexible enough such that the used sampling-based algorithms can be interchanged by another equivalent sampling method such as the RRG [6] .

The main contributions of this work are two: (1) It is proposed to use optimal motion planning algorithms to find collision-free trajectories to reconstruct the object. In our approach, we do optimize a utility function composed of several factors (see Sect. 4). One of the elements of the utility function is the length of the path. In this paper, the effect of using an optimal sampling-based motion planning to compute the path to reach the next best view, in the task of object reconstruction is analyzed. Surprisingly, we have found that optimizing the path length has unexpected consequences in the reconstruction task: (1) it reduces the positioning error to reach a sub-goal under the effect of noise, (2) it reduces the rate of collision with the obstacles, (3) it increases the percentage of reconstruction, and it improves the quality of the reconstruction. These results are explained as follows: since the planner delivers a shorter path (w.r.t. a path generated with a standard RRT), the noise in the control during the execution of the path has a smaller opportunity of alternating the path, resulting in a smaller positioning error and a smaller rate of collision. These two resulting properties affect the quality of the reconstruction, given that the sub-goals (next best views) are more accurately reached, having as a final consequence a larger percentage of reconstruction and a more accurate resulting 3-D model. Based on our experimental results, it is concluded that the use of optimal motion planners (in the present work the RRT* and RRT* smart) improves all measured performance criteria compared with a standard RRT. 2) A probabilistic stopping test to terminate the reconstruction process is proposed. The experimental results about this probabilistic test show that the reconstruction is always finished when all the object surface that is possible to be sensed has been perceived. We stress the fact that the works in [22] and [23] do not present these new contributions.

In the next section, related previous work about sampling-based motion planning and object reconstruction with robots is presented.

2 Previous work

The present paper finds itself within the 3D object reconstruction context. There is much work available within that field; however, the authors of [1,16] present relevant surveys concerning the object reconstruction problem. Just to cite some works among many contributions, the work in [21] proposes a method that plans a next best view (NBV) for object reconstruction in the workspace and then inverse kinematics is calculated to obtain a configuration that matches the desired sensor location. In [20], a mobile manipulator robot is used to reconstruct an object. The problem of finding collision-free paths is simplified by decoupling the robot motions. 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).

The work presented in [9] has as a main objective 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) [7] and probabilistic road maps (PRMs) [10] to find collision-free paths. In [8], an approach is proposed 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 combined with a best view selection algorithm based on mean shift.

In [14], 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. Following a similar vein, in [3] it is investigated which formulation of information gain is best for the volumetric 3D reconstruction of an object by a robot equipped with a dense depth sensor. The authors propose formulations that incorporate factors such as visibility likelihood and the likelihood of seeing new parts of the object. Additionally, that work presents a comparative survey of volumetric information formulation performance for active 3D object reconstruction.

In [2], a method for exploring an unknown environment with a unmanned aerial vehicle (UAV) is proposed. The method selects the best movement by determining the frontier voxel that minimizes the cost; such cost incorporates the dynamics of the UAV in order to maintain the robot at optimal speed. In [17], the authors propose a method for inspecting a partially known environment. First, a target goal is computed, and then, it refines a path until a local area is inspected. The inspection is completed when the percentage of unknown volume with respect to the entire unknown volume is lower than a threshold. More recently, in [18], the same authors realized that the completion of a volumetric map does not necessarily describe the completion of a 3D model; hence, they evaluate the model completeness according to the quality of the reconstructed surfaces and extract low-confidence surfaces. The surface information is used to guide the computation of the exploration path. In [19], a motion planning strategy for an active vision-based mapping is proposed. The method actuates in two steps: first it follows the contour of an unknown target, and then, it moves to the missing portions. An imposed constraint is to revisit already scanned zones in order to decrease the localization errors.

To our knowledge, the effect of using a sampling-based optimal motion planning technique to move the robot, during the reconstruction process, in terms of the performance criteria proposed in this work has not been investigated before, nor the difficult problem of defining a convenient stopping test to terminate the reconstruction process.

In the section below, we begin to present the proposed framework for object reconstruction task.

3 Observation and motion models

In this section, we briefly present the observation and motion model used in this work. The motion model considers noise under the controls, and the observation model represents the reconstructed object with a probabilistic octree. For more detailed description, please see [23].

3.1 Observation model

The observation model corresponds to an octree representation of the object and the classes assigned to the voxels in the octree according to their probabilities. We assume that the object shape is unknown, but the position and size of the object are known; with this information an object bounding box, $\mathcal{W}_{\text{box}}$ is established containing the object to be reconstructed, the environment except W_{box} is known. We also assume that before the first robot motion, 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.

To represent the content of the object bounding box W_{box}, a probabilistic occupancy map based on the octomap structure [4] is used, which is an octree with probabilistic occupancy estimation. In this representation, each voxel has associated a probability of being occupied. We transform a sensed observation (a set of 3D points) to classes. Depending on the probability of been occupied, we classify each voxel with one of the three possible classes: (i) occupied, which represents surface points measured by the range sensor (this class has a probability larger than 0.55), (ii) free, which represents free space (this class has a probability less than 0.45) and (iii) unknown, whose space has not been seen by the sensor. This class has the interval $[0.45, 0.55]$. One main advantage of defining these classes is that they allow us knowing 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.

We assume a Kinect sensor providing 3D points as measurements; the sensor has limited range and field of view. A scan recovers information of the workspace inside the sensor's frustum at the view $V(X)$, where X is the system state. A view is calculated by direct kinematics in order to get the pose of the sensor given by a robot state. We use a Denavit–Hartenberg model of our mobile manipulator robot to perform the direct kinematics computation. Once a scan has been made, the sensor readings are integrated into the octree. The occupancy probability of a voxel is updated according to the octomap sensor fusion model proposed in [4]. The matching between the reconstructed point cloud and the new scanned surface is used to re-localize the robot in the reference frame defined by the object bounding box, after each scan.

Visibility computation allows us to determine which type of voxels are visible for a given sensor pose. Visibility is computed using the hierarchical ray tracing presented in [23], which reduces the processing time to calculate it compared with a uniform ray tracing.

3.2 Motion model

The robotic base is controlled by linear and angular velocities. It is assumed that these velocities are not perfect [23]. 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:

$$\begin{aligned} \hat{v} &= v + \epsilon_{\sigma_v^2} \\ \hat{\omega} &= \omega + \epsilon_{\sigma_w^2} \end{aligned} \tag{1}$$

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.

$$\begin{aligned} \hat{v} &= v + \texttt{sample}\,(\sigma_v^2) \\ \hat{\omega} &= \omega + \texttt{sample}\,(\sigma_\omega^2) \end{aligned} \tag{2}$$

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:

$$\hat{\omega}_i = \omega_i + \texttt{sample}(\sigma_{\omega_i}^2) \tag{3}$$

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.

To obtain a robot trajectory, we use the Euler integration method over the robot state variables (x, y, θ_b, θ_1, θ_2, θ_3, θ_4, θ_5) considering noise over the robot controls, modeled as mentioned above. In this work, we distinguish a state from a configuration as following: a state considers the effect of the control noise that determines the reached position and orientation of the robot, while a configuration does not have position and orientation errors.

Thus, the reached robot state X_t is given by:

$$\begin{aligned} x_t &= x_{t-1} + \hat{v}_x \Delta t \\ y_t &= y_{t-1} + \hat{v}_y \Delta t \\ (\theta_b)_t &= (\theta_b)_{t-1} + \hat{\omega}_b \Delta t \\ (\theta_i)_t &= (\theta_i)_{t-1} + \hat{\omega}_i \Delta t \end{aligned} \tag{4}$$

The robotic base is able to move omnidirectionally, but it has an orientation θ_b.

We are assuming independent errors, so the probability of reaching a next consecutive state X_t from state X_{t-1}, applying control u_{t-1}, is given by:

$$p(X_t|u_{t-1}, X_{t-1}) = p(\epsilon_{\sigma_{v_x}^2}) p(\epsilon_{\sigma_{v_y}^2}) p(\epsilon_{\sigma_{\omega_b}^2}) \prod_{i=1}^{n} p(\epsilon_{\sigma_{\omega_i}^2}) \tag{5}$$

In our modeling, assuming independent errors is equivalent to have an omnidirectional (holonomic) robot in which moving forward/backward to the right/ to the left or rotating is controlled by independent motors.

Below, we present the next best view selection, which is done through several filters, which first discard candidate configurations and then select the one that optimizes a utility function.

4 Next best view selection

In this work, first candidate robot configurations are generated by uniform sampling; then, some of these configurations are selected as sub-goals. A *sub-goal* is a configuration that has been chosen to be visited by the robot to perform a sensing operation at that place to reconstruct the object.

To select the next best view, which is the next sub-goal to be visited in the reconstruction process, we proceed as follows. The set of robot's configurations generated by uniform sampling is ranked according to a utility function. The used utility function is a product of factors [22] that one wants to maximize:

$$g(X_i) = pos(X_i) \cdot reg(X_i) \cdot sur(X_i) \cdot dist(X_i) \tag{6}$$

where each factor evaluates a constraint.

The four factors are: (i) position, $pos(X_i)$, which is related to collision detection, (ii) registration, $reg(X_i)$, which measures the overlapping between views, (iii) surface, $sur(X_i)$, which is related to new discovered surface, and (iv) distance, $dis(X_i)$, which penalizes the path length. To perform this evaluation efficiently, it is done through several filters, so that the candidate 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 position factor $pos(X_i)$ is evaluated; the candidates in collision are eliminated, $pos(X_i) = 0$. Then, the visibility of each state is calculated using an efficient scheme based on hierarchical ray tracing [23]. For performing the registration process, only the candidates that guarantee a minimum overlap with previous views are maintained, the $reg(X_i)$ factor verifies that a minimum overlap is satisfied, and this factor is given by:

$$reg(x) = \begin{cases} 1 & \text{if } \dfrac{oc_o(X_i)}{oc_o(X_i) + un_o(X_i)} > h \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where $oc_o(X_i)$ indicates the amount of occupied voxels that are sensed and lie inside W_{box}, $un_o(X_i)$ is the amount of

unknown voxels in W_{box}, and h is a threshold. In our experiments, we have fixed h to 50%.

Next the amount of unknown new observed surface, that is the amount of unknown voxels that are observed, is considered by $sur(X_i)$ factor. This factor is normalized by the total number of remaining unknown voxels. Thus, the $sur(X_i)$ factor is given by the following equation.

$$sur(X_i) = \frac{un_o(X_i)}{un_{\text{total}}} \tag{8}$$

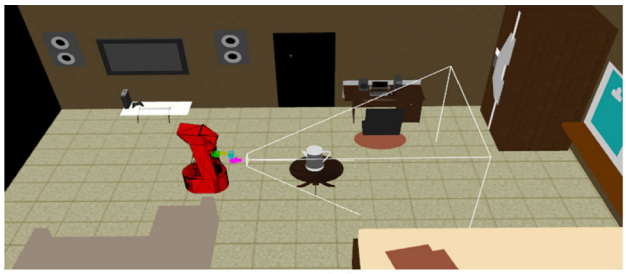
where $un_o(X_i)$ is the amount of unknown voxels that are observed inside W_{box} and un_{total} is the total amount of unknown voxels inside W_{box}.

The candidates that passed the collision and minimum overlap filters are ranked based on their expected utility. The expected utility is the most likely utility that a state will have under noise in the robot's controls. To determine the expected utility, we generate k trajectories per candidate view X_q, by simulating k times the execution of the robot's controls as a stochastic process. For each simulation, the algorithm starts from the current state and applies controls with noise according to a motion model described in Sect. 3.2 and having as sub-goal X_q (that is never reached exactly due to the noise); the actual reached state is called X_i. Further details in the generation of the robot trajectory and how the RRT and RRT* are used for such purposes are described in Sect. 5.1. The utility of a state X_i also has associated a "distance" factor $dist(X_i)$, which is inversely proportional to the traveled length (considering both translation and orientation) from the current robot state to state X_i, given the path followed by the robot under the effect of noise.

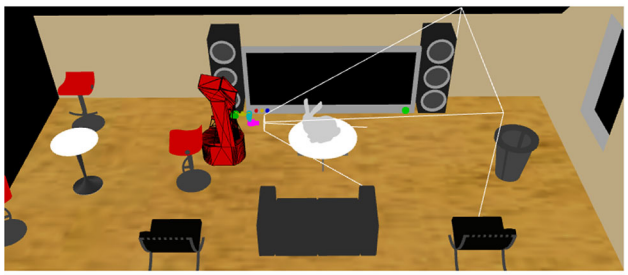
To compute the probability of occurrence of each state X_i, we assume independence of the errors over time; therefore, the probability of each state X_i is calculated as the product of the probabilities of the occurrence of each X_t state, i.e., $P(X_i) = \prod_{t=1}^{T} p(X_t|u_{t-1}, X_{t-1})$. T is the number of times that a control under noise is executed to reach X_i. Each generated state, X_t, is tested for collision if one of them is in collision, then the candidate state, X_i, is unfeasible, i.e., $P(X_i) = 0$. Then, we normalize the probabilities of the k reached states by using the following term $\eta = \frac{1}{\sum_{i=1}^{k} P(X_i)}$. Finally, we compute the expected utility associated with the sub-goal reached in average X_{sg} using the following equation: $E(g(X_{sg})) = \sum_{i=1}^{k} g(X_i) \cdot \eta P(X_i)$.

The *best* candidate X_{sg} according to the expected utility is selected. The 3D reconstruction cycle is repeated until a stopping condition is satisfied or no path was found for any X_q.

In the next section, we study in detail the implications of using shortest trajectories in the task of object reconstruction.



(a)



(b)

Fig. 1 Environments 1 and 2 used for the simulation experiments. Environment 2 has more obstacles than the other. **a** Environment 1. **b** Environment 2

5 Planning robot's trajectories using RRT*: implications of shortest trajectories

In this section, we present several simulation experiments. In the first set of experiments, we compare the processing time using a single RRT* versus using a standard RRT per sub-goal. In the second set of experiments, we compare the RRT* versus the standard RRT in terms of criteria related to the resulting trajectories (the cost under imperfect controls, the positioning error and the rate of collisions). Finally, in the third set of experiments, we compare both methods in terms related to the reconstructed model (the percentage of the reconstruction, the distance from each 3D point in the reconstructed model to the closest point in a ground truth model and the average density of 3D points per voxel in the reconstructed model). We have also analyzed the effect of increasing the noise over the reconstruction process.

We perform the experiments in two different environments (see Fig. 1), one with more obstacles than the other. We have used three different objects to be reconstructed: a teapot, a bunny and a dragon, see Fig. 2. In all the simulation experiments, the robot is a mobile manipulator with 8 degrees of freedom; the robotic base is omnidirectional, see Fig. 6.

5.1 A single tree for several sub-goals versus a tree per sub-goal

In the work in [23], to reach a candidate view X_q a different RRT is used to reach each one of those views; when a node in

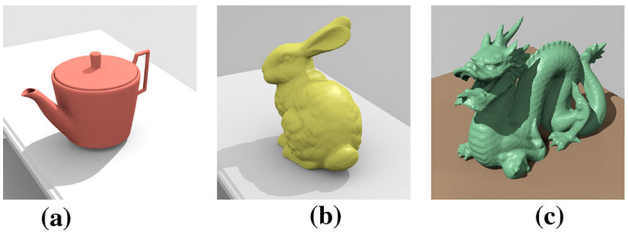


Fig. 2 Object of different complexities used in the simulation experiments: **a** teapot, **b** bunny and **c** dragon

the current RRT is closer to the candidate view than a given threshold, then the candidate view is declared as reached. In this work, we are using a RRT* instead of a RRT. In our current implementation, we proceed differently, in order to have a more efficient implementation that in general consumes less processing time, we generate a single RRT* to reach all candidate views and we stop the RRT* generation process based on the number of nodes in the tree; then, we fix a neighborhood around a goal based on a given metric. All the nodes in the neighborhood have reached the goal. Since as the number of nodes increases in RRT*, the resulting trajectories asymptotically approach optimality; it makes sense to use the same RRT* to reach all candidate views. Figure 3a shows a single RRT* to reach all the 3 configurations; the other sub-figures show a different RRT to reach each configuration. Using a single optimal tree to reach several goals considerably reduces the processing time. Below we present some results.

We have done 10 simulation experiments of a total reconstruction for each of the 3 objects and 2 environments with the RRT* (60 experiments in total) and other 10 simulation

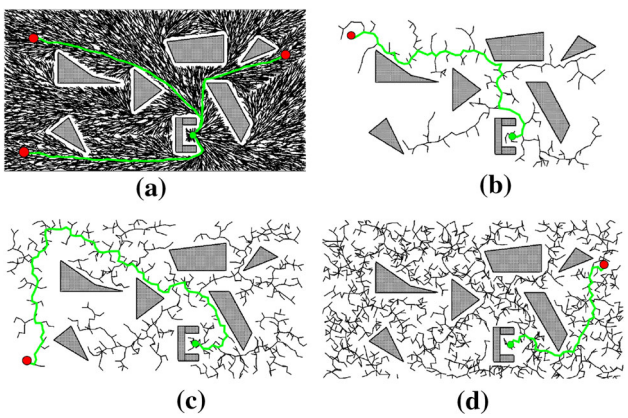


Fig. 3 The top left figure **a** shows a single RRT* to reach 3 configurations shown in red. In the other figures, a different RRT is used to reach each one of the 3 red configurations. **a** A single tree obtained with a RRT* to reach 3 configurations, **b** a tree obtained with a RRT to reach a single configuration, **c** a second tree obtained with a RRT to reach a single configuration, and **d** another tree obtained with a RRT to reach a single configuration (color figure online)

Table 1 Statistical mean of planning time

	RRT	RRT*
Environment 1		
Bunny		
Planning time	96.7687	36.1023
Teapot		
Planning time	131.0451	36.7867
Dragon		
Planning time	114.5917	37.6350
Environment 2		
Bunny		
Planning time	161.6326	39.6830
Teapot		
Planning time	153.1874	41.6643
Dragon		
Planning time	158.6397	42.3413

experiments for each object and environment with the RRT. Table 1 shows the statistical mean of the cumulative processing planning time needed to generate the trajectories per object and per environment, using a single RRT* and a RRT. All the times are given in seconds. One can observe that the cumulative time is always smaller when a single RRT* is used instead of a RRT per sub-goal. The time to generate paths with the RRT* is smaller than the time used to generate paths with the standard RRT, because a RRT is used to reach each one of the candidates to be the next best view, while we use the same RRT* to reach several candidates to be the next best view, taking advantage of the property of the RRT* that gives the shortest path from the initial configuration to any node.

One can also observe that the processing time is a bit larger in environment 2 compared with environment 1; this is expected, since environment 2 has more obstacles than environment 1. Regarding the objects, the time is larger for the dragon object; this is normal since it is harder to sense all the surface of this object. We have found that when the RRT* is used most of the time is used to compute the expected utility and not the trajectory to reach a sub-goal state; indeed, only 39.33 % of the time is used to generate the trajectories. In contrast when the RRT is used, 52.28 % of the time is used to generate the trajectories.

Figure 4 shows the processing time per iteration, for one experiment with the bunny object in environment 1; each iteration corresponds to determine the next best view (sub-goal) and the path to reach it. Figure 5 shows the cumulative processing time as the reconstruction process keeps progressing.

In the next section, we compare the RRT and RRT* in terms of criteria related to the resulting trajectories. First,

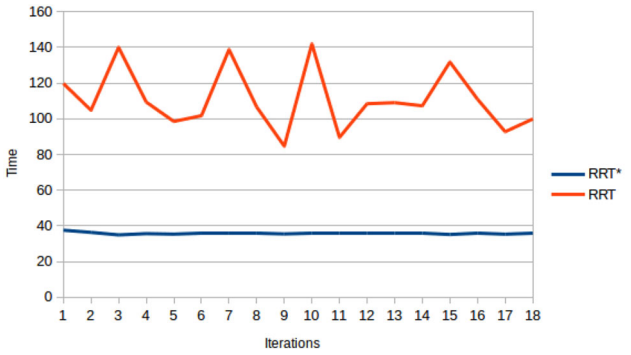


Fig. 4 Processing time per iteration, for one experiment with the bunny object in environment 1. Each iteration corresponds to determine the next best view (sub-goal) and the path to reach it

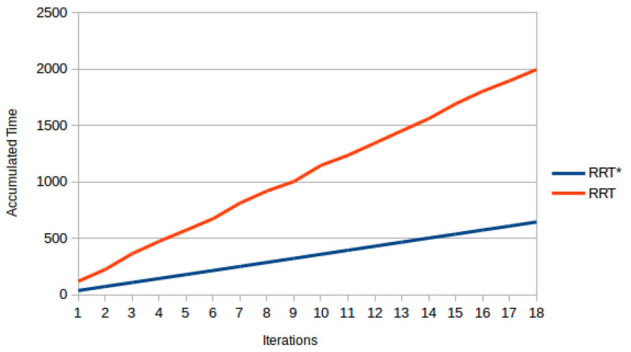


Fig. 5 Cumulative planning time for one experiment with the bunny object in environment 1

we compare the cost obtained with a RRT versus a RRT*. Second, we compare the robot location and error, and then, we analyze the rate of collision; all these experiments are done under the influence of noise.

5.2 Comparing the RRT* versus RRT: cost, positioning error and rate of collisions

In this section, we study the effect of noise in the reconstruction process. In particular, we analyze the pertinence of using a RRT* versus a standard one. We assume a Markov decision process (MDP), in which one does not know deterministically the next state due to the noise over the controls, but once that the state is reached, the state is known.

We compare the trajectories in terms of the metric given in Eq. 9, which measures the cost between two given states X_p and X_{p+1}, using a RRT* versus using a standard RRT. The robot has 8 degrees of freedom, (x, y) denote the robot's position, $\theta_{j=1}$ denotes the orientation of the robotic base and θ_j with $j \in [2, \ldots, 6]$ denotes the orientations of the arm's links.

$$d(X_{p+1}, X_p) = \sqrt{(x_{p+1} - x_p)^2 + (y_{p+1} - y_i)^2 + \sum_{j=1}^{n} (\theta_{p+1,j} - \theta_{p,j})^2} \tag{9}$$

The next best view/state is computed using the method proposed in [23], which is summarized in Sect. 4.

For the RRT*, as the number of nodes increases, the resulting paths get closer to the shortest ones. In the experiments, we build a RRT* until a given number of nodes in the tree are reached. We fixed this number to 10,000 nodes. For a given sub-goal X_q that does not consider noise, we fix a neighborhood based on the metric defined in Eq. 9. All the nodes X_c in the RRT* closer to the goal that a given threshold γ are considered as candidates, that is all the nodes with $d(X_q, X_c) < \gamma$. Among those candidates, we choose the one having the optimal utility defined by the utility function in Eq. 6. We call this trajectory the reference one.

In the case of the standard RRT, for each sub-goal X_q we build a different RRT; we consider that the sub-goal X_q is reached if there is a node in RRT that is closer to X_q than the same γ used in the RRT* experiments. In the work presented in [6], it has been shown that the standard RRT by itself will never generate a path that converges to the optimal one; hence, increasing the number of sampling will not improve the resulting path; it will just consume more time. The resulting trajectory is the reference trajectory for the standard RRT.

Figure 6 shows a reference trajectory (without noise) generated with the RRT*, Fig. 6 shows that path followed by the robotic base and Fig. 6 shows the path followed by the end effector of the arm. Figure 7 shows a reference trajectory generated with the standard RRT.

Then, we perturb both the reference trajectories of the RRT* and standard RRT by adding noise to the controls according to the model described in Sect. 3.2. The noise over the robot's control is white Gaussian with zero mean and standard deviation of 0.0096. We consider noise only in the robotic base (position and orientation) This standard deviation is the same for the 3 degrees of freedom.

Every perturbed trajectory reaches a given state called X_i. To model an average sub-goal, denoted X_{sg}, we compute the average of the position and orientation considering all the X_i, that is: $\hat{x} = \sum_{i=1}^{n} \frac{x_i}{n}$, $\hat{y} = \sum_{i=1}^{n} \frac{y_i}{n}$ and $\hat{\theta} = \sum_{i=1}^{n} \frac{\theta_i}{n}$. Recall that we assume a MDP, once that the state is reached, it is known, so state X_{sg} is the departing state for computing the next view.

We compare the RRT* versus the RRT in terms of the cumulative cost under imperfect controls, which is the average of the sum of the cost given in Eq. 9 between all the pair of states related to all the perturbed trajectories. Again, we have

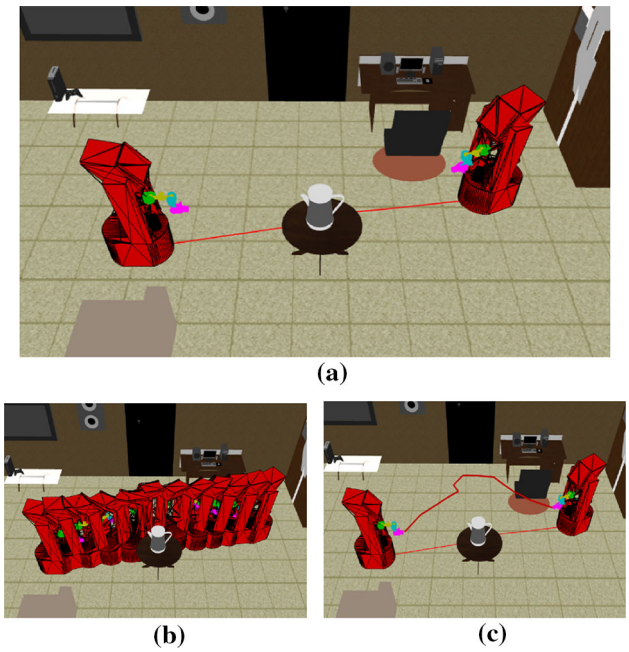


Fig. 6 Robot's trajectory without noise using a RRT*. **a** The path of the center of the robot base, **b** the robot trace, and **c** the path of the sensor

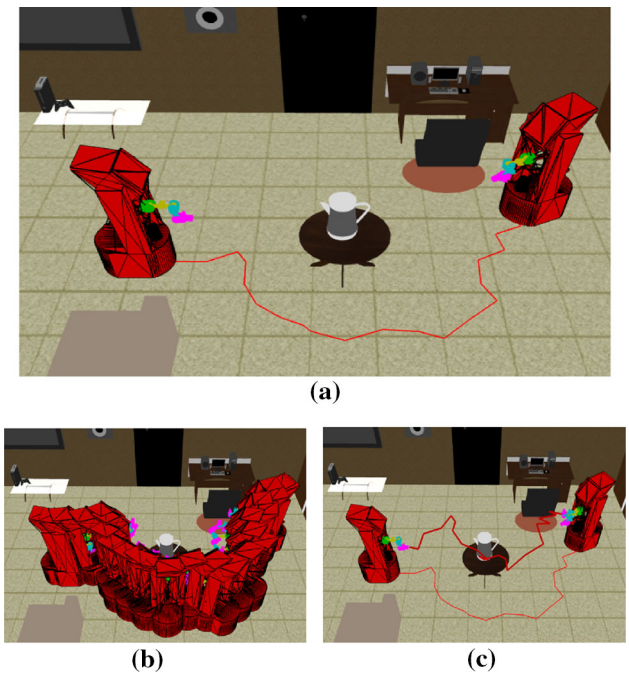


Fig. 7 Robot's trajectory without noise using a RRT. **a** The path of the center of the robot base, **b** the robot trace, and **c** the path of the sensor. The trajectories are clearly longer than the ones obtained using the RRT*

done 10 simulation experiments of a total reconstruction for each object and environment. Table 2 shows the statistical mean and standard deviation of the cumulative cost under imperfect controls per object and per environment, using a

Table 2 Cumulative cost

	RRT	RRT*
Environment 1		
Bunny		
Mean cumulative cost	105.4807	69.0065
Standard deviation cumulative cost	38.4804	20.9711
Teapot		
Mean cumulative cost	97.7761	72.0794
Standard deviation cumulative cost	37.6580	24.3741
Dragon		
Mean cumulative cost	133.4752	70.6189
Standard deviation cumulative cost	35.8326	19.8063
Environment 2		
Bunny		
Mean cumulative cost	92.5726	69.1047
Standard deviation cumulative cost	36.0682	19.9069
Teapot		
Mean cumulative cost	93.0627	60.7837
Standard deviation cumulative cost	48.7516	23.2541
Dragon		
Mean cumulative cost	83.7586	62.1794
Standard deviation cumulative cost	35.5682	24.6971

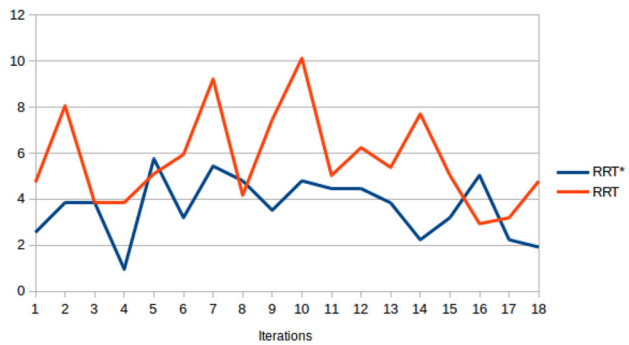


Fig. 8 Statistical mean by iteration of the trajectory cost defined by Eq. 9, of an experiment in environment 2 for the teapot object. The mean by iteration of the trajectory cost is very often smaller when the RRT* is used

RRT* and a RRT. Both the mean and standard deviation of the cumulative cost are smaller when the RRT* is used for the 3 objects in both environments. Since the RRT* produces shorter paths with respect to the ones generated with the RRT, it is understandable that the cumulative cost is smaller even in the presence of noise.

Figure 8 shows the statistical mean by iteration of the trajectory cost defined in Eq. 9, and Fig. 9 shows the cumulative mean cost, for a simulation experiment where the object to be reconstructed is the teapot in environment 2. Each iteration corresponds to reach a next best view, that is, to reach the

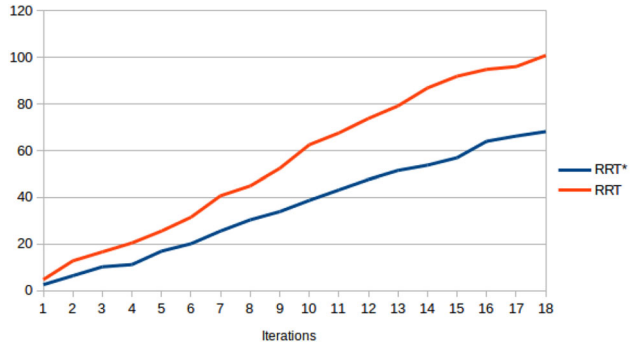


Fig. 9 Cumulative mean cost of an experiment in environment 2 for the teapot object

next sub-goal where a sensing operation is performed during the reconstruction process.

Now, we compare both methods in terms of the positioning error given in Eq. 10 and the rate of collisions.

Recall that the motion model that we use corresponds to the one in Eq. 4 of Sect. 3.2. We are sampling a noise distribution to generate imperfect controls, when those controls are executed a state (position and orientation) is reached with a certain probability. We assume a Markov decision processes (MDPs); once that the state is reached, it is known.

$$\begin{aligned} Error_{pos} &= \sqrt{(x_i - x_{sg})^2 + (y_i - y_{sg})^2} \\ Error_{ori} &= \sqrt{(\theta_i - \theta_{sg})^2} \end{aligned} \tag{10}$$

The positioning error has two components, one is the location $Error_{pos}$ and other is the orientation $Error_{ori}$, $(x_{sg}, y_{sg}, \theta_{sg})$ is the sub-goal configuration without considering noise and (x_i, y_i, θ_i) is the reached state with the effect of noise. To compute the positioning error, only the degrees of freedom of the robotics base (x, y, θ) are taken into account, given that the degrees of freedom of the arm are typically less noisy.

We proceed in the same way that in the previous set of experiments. That is, first we compute reference trajectories without noise and then we perturbate the trajectories with noise. The noise is also the same standard deviation of 0.0096 only in the robot base. We do one complete reconstruction experiment for each object and each environment (six whole reconstructions). In each of these reconstruction experiments, we generate 100 trajectories with noise for the RRT and also 100 for the RRT* to reach each next best view. Over these 100 trajectories, we compute the mean of the error given in Eq. 10. Then to assess each whole reconstruction trajectory, the statistical mean per reconstruction is computed by using the mean of each sub-goal. Table 3 shows the mean and variance of the location and orientation positioning errors, per object and per environment. The small numbers in the location errors are due to two reasons: (1) the size of the environment 1 is in the order of 6 units long × 6 units width, and the size of environment 2 of the order of 6 units long × 4 units width, (2) the noise over the robot's control has a standard deviation of 0.0096. The orientation error is given in radians. One can observe in Table 3 that both the location and orientation errors and their variances are smaller when the RRT* is used.

Table 3 Statistical mean and variance of location and orientation errors

	RRT	RRT*
Environment 1		
Bunny		
Mean location error	0.1463	0.1001
Mean orientation error	0.0925	0.0626
Variance location error	0.0064	0.0028
Variance orientation error	0.0053	0.0024
Teapot		
Mean location error	0.1381	0.1069
Mean orientation error	0.0863	0.0674
Variance location error	0.0053	0.0033
Variance orientation error	0.0044	0.0027
Dragon		
Mean location error	0.1528	0.1011
Mean orientation error	0.0966	0.0639
Variance location error	0.0066	0.0030
Variance orientation error	0.0053	0.0027
Environment 2		
Bunny		
Mean location error	0.1405	0.1054
Mean orientation error	0.0883	0.0626
Variance location error	0.0055	0.0033
Variance orientation error	0.0045	0.0027
Teapot		
Mean location error	0.1353	0.1034
Mean orientation error	0.0859	0.0653
Variance location error	0.0052	0.0029
Variance orientation error	0.0041	0.0026
Dragon		
Mean location error	0.1391	0.0999
Mean orientation error	0.0851	0.0677
Variance location error	0.0052	0.0030
Variance orientation error	0.0044	0.0027

Since the RRT* delivers shorter paths (w.r.t. paths generated with a standard RRT), the noise in the control during the execution of the trajectories has a smaller opportunity of alternating the original paths resulting in a smaller error.

Figure 10a shows the mean of the location error by iteration for one experiment with the dragon object in environment 2. Each iteration corresponds to reach a next best

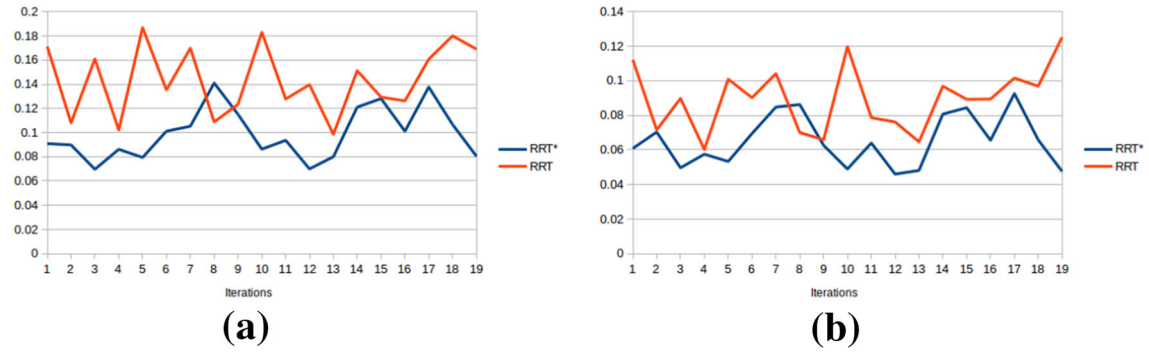


Fig. 10 Positioning error, dragon object, environment 2. The mean over the position error is smaller in almost all the iterations when a RRT* is used. **a** Mean of location error, environment 2, dragon object, and **b** mean of orientation error, environment 2, dragon object

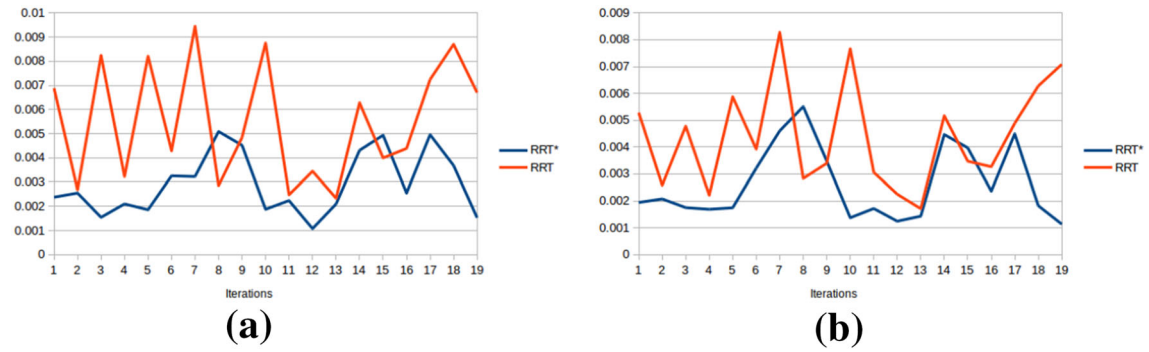


Fig. 11 Variance of positioning error, dragon object, environment 2. The variance over the position error is smaller almost in all the iterations when a RRT* is used. **a** Variance of location error, environment 2, dragon object, and **b** variance of orientation error, environment 2, dragon object

view, that is, to reach the next sub-goal where a sensing operation is performed during the reconstruction process. Figure 10b shows the mean of the orientation error by iteration for the same experiment.

Figure 11 shows the variances of the positioning error by iteration for the same experiment with dragon object in the environment 2. Both the mean and the variance over the position error are smaller almost in all the iterations when a RRT* is used. Since the RRT* generates shorter trajectories, the noise over the controls produces a smaller positioning error, at each sub-goal, compared with the positioning error related to using a standard RRT. The mean over the position error is smaller in all the experiments when a RRT* is used, see Table 3.

Figure 12 shows trajectories under noise using a RRT and a RRT*.

Figure 13 shows the desired sub-goal and the one reached under noise using a RRT and a RRT*. We have also analyzed the effect of increasing the noise over the rate of collision in the reconstruction process. An addition set of experiments was performed with a noise having zero mean and 0.025 standard deviation in the 3 degrees of freedom (x, y, θ) of the robot base. We generate 10 experiments per object and per environment with the RRT and other 10 experiments with the RRT*.

The rate of collisions is defined as the ratio between the number of experiments in which a global reconstruction trajectory yields a collision of the robot with an obstacle (denoted tc) divided by the total number of simulation experiments (denoted by tnt), that is: $\frac{tc}{tnt}$. A global reconstruction trajectory is the trajectory that the robot follows to visit sub-goals in a given order until the reconstruction process is terminated.

Table 4 shows the rate of collision per object and environment. Since the level of noise is very large, it is 0.025 (2.6 times larger than in the other experiments in the paper), the rate of collision is also large. However, it is smaller when the RRT* is used (see Table 4). Furthermore, the percentage of object reconstruction is in the worst results at least 65 % (as defined in the next section, please see below), when the RRT is used, since the collision happens at the end of the reconstruction process, when it is harder to see new object surface. If a collision occurs, then the reconstruction process is considered as finished.

In the next section, we analyze in detail the percentage of object reconstruction and the quality of the reconstructed model for a level of noise of 0.0096.

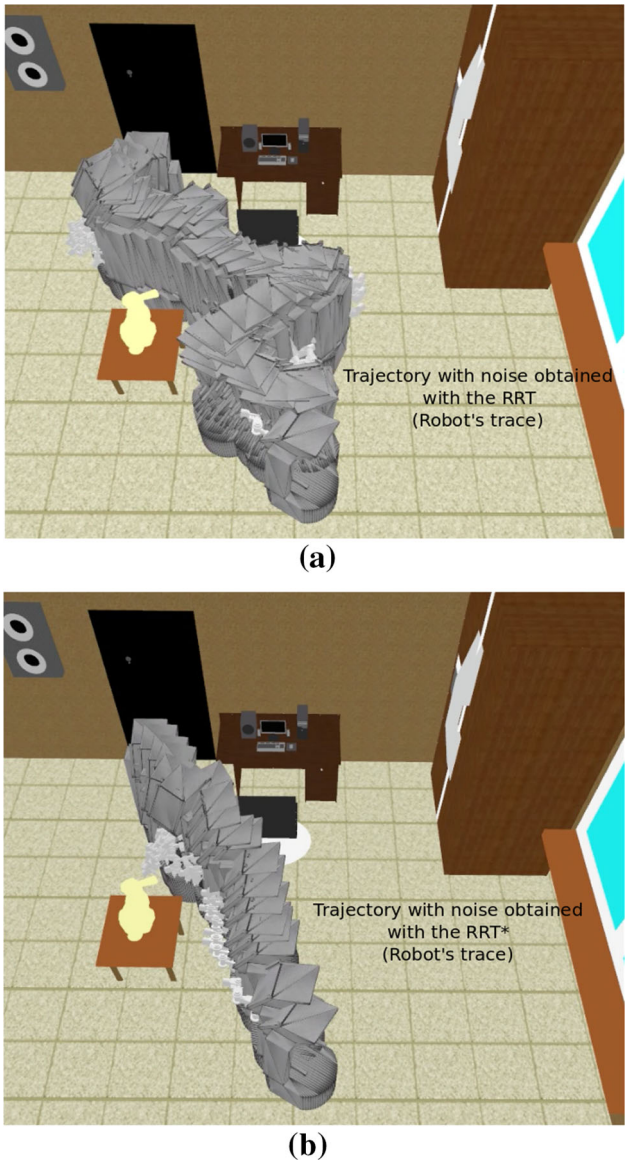


Fig. 12 **a** The resulting trajectory under noise using a standard RRT, and **b** The resulting trajectory under noise using a RRT*. The path obtained with the RRT* is shorter.

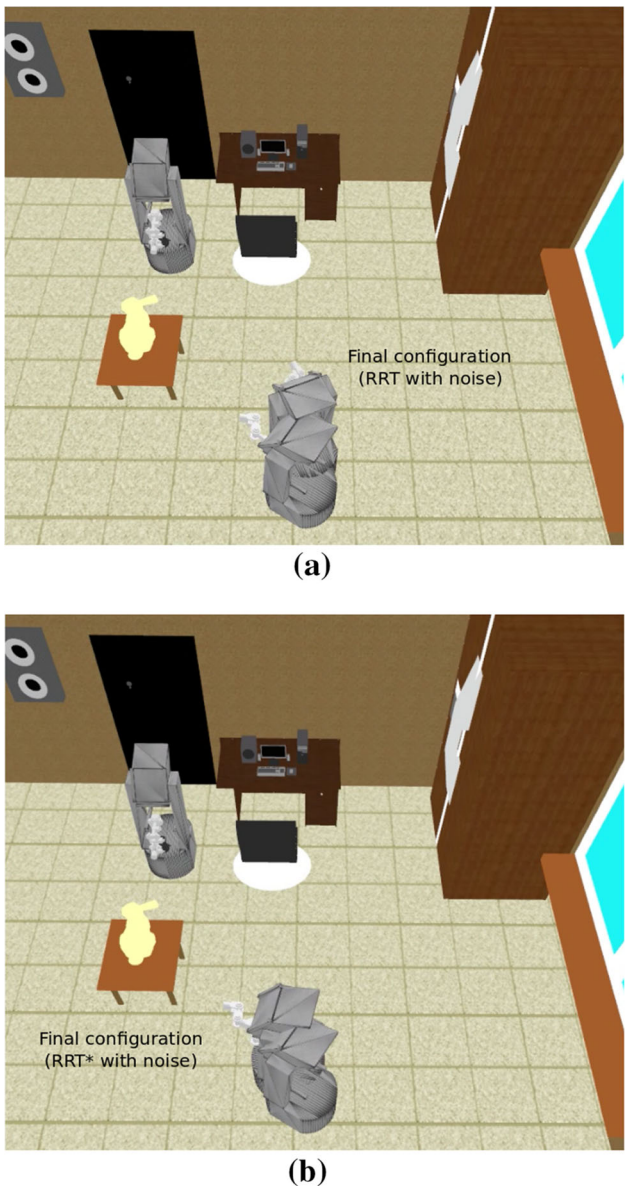


Fig. 13 **a** The desired sub-goal and the one reached under noise using a standard RRT, and **b** the desired sub-goal and the one reached under noise using a RRT*. Since the RRT* generates shorter trajectories then the noise over the controls produces a smaller positioning error, compared with the one related to a standard RRT

5.3 Percentage of reconstruction and quality of the reconstructed model

In our simulation experiments, we have a cloud of 3D points corresponding to the ground truth model of the object to be reconstructed. To assess the quality of the reconstructed model, we proposed the following quantities.

The percentage of object reconstruction is computed as the ratio of correspondent points (denoted cp) multiplied by 100 over the total number of points in the ground truth model (denoted NpM), that is $\frac{cp \times 100}{NpM}$. A correspondent point is a ground truth point closer than a threshold (0.005 units) to a built model point.

M_{dmin} in Eq. 11 is the statistical mean of the minimum distance from every point in the ground truth model to the closest point in the reconstructed object model, $dmin_i$ is the minimum distance from point i in the ground truth model to the closest point in the model obtained after having finished the reconstruction task, and NpM is the total number of 3D points in the ground truth model. The smaller M_{dmin} implies that the reconstructed model is more similar to the real object, since every point in the ground truth model has a point close in the reconstructed model.

Table 4 Rate of collision per object and environment

	RRT	RRT*
Environment 1		
Bunny		
Rate of collision	0.8	0.4
Teapot		
Rate of collision	0.8	0.5
Dragon		
Rate of collision	0.7	0.4
Environment 2		
Bunny		
Rate of collision	0.8	0.5
Teapot		
Rate of collision	0.8	0.5
Dragon		
Rate of collision	0.7	0.5

Table 5 Statistics of performance criteria in environment 1

Performance criterion	RRT	RRT*
Bunny		
Percentage of reconstruction	88.6269	98.0718
Mean: number of sensing locations	15.3	15.9
Mean: minimum distance	0.0153	0.0140
Mean: density of points	3.337867×10^6	3.910928×10^6
Teapot		
Percentage of reconstruction	84.8015	94.7671
Mean: number of sensing locations	13.5	16.7
Mean: minimum distance	0.0101	0.0095
Mean: density of points	2.554481×10^6	3.620996×10^6
Dragon		
Percentage of reconstruction	89.8169	95.2738
Mean: number of sensing locations	19.3	18.7
Mean: minimum distance	0.0033	0.0026
Mean: density of points	2.949319×10^6	3.794405×10^6

$$M_{dmin} = \sum_{i=1}^{NpM} \frac{dmin_i}{NpM} \tag{11}$$

Other criterion that we are using to assess the quality of the reconstructed model is the density; den_i is the density of points of the voxel i, $NpVox_i$ is the number of 3D points inside that voxel and $VolVox_i$ is the volume of the voxel.

$$den_i = \frac{NpVox_i}{VolVox_i} \tag{12}$$

M_{den} in Eq. 13 is the statistical mean of the density of points per voxel of a reconstructed model after having finished the reconstruction task, and $NvoxO$ is the total number of occupied voxel in the reconstructed model. We prefer reconstructed models having a larger M_{den}.

$$M_{den} = \sum_{i=1}^{NvoxO} \frac{den_i}{NvoxO} \tag{13}$$

Table 6 Statistics of performance criteria in environment 2

Performance criterion	RRT	RRT*
Bunny		
Percentage of reconstruction	86.0907	95.7208
Mean: number of sensing locations	15.5	17.1
Mean: minimum distance	0.0158	0.0140
Mean: density of points	3.822429×10^6	5.164318×10^6
Teapot		
Percentage of reconstruction	83.8062	93.0099
Mean: number of sensing locations	17.4	14.2
Mean: minimum distance	0.0099	0.0087
Mean: density of points	3.490615×10^6	3.819518×10^6
Dragon		
Percentage of reconstruction	79.2797	92.9745
Mean: number of sensing locations	16	17.8
Mean: minimum distance	0.0054	0.0027
Mean: density of points	3.384160×10^6	4.253567×10^6

We have done other 10 simulation experiments per each one of the three objects in the environment 1 (see Fig. 1) and other 10 in environment 2 (see Fig. 1), in order to assess the performance criteria defined above. In this set of experiments, the noise over the robot's control is white Gaussian with zero mean and standard deviation of 0.0096, for both the RRT and the RRT*. This standard deviation is the same for the three degrees of freedom (position and orientation) of the robotic base. The degrees of freedom of the arm do not have noise.

Table 5 shows the results of these experiments in environment 1, depending on whether a RRT or a RRT* is used. Table 6 shows the same results in environment 2. We can observe that for the 3 objects the percentage of reconstruction and the average density per voxel are larger and the average minimum distance is smaller when a RRT* is used. The typical large average density appears because the volume of each voxel is very small ($8 \times 10^{-6} u^3$). Often the number of sensing location (sub-goals) is smaller when a RRT is used, this happens because the RRT yields more collisions of the robot and an experiment is stopped when the robot collides. These results allow us to conclude that both the percentage of reconstruction and the quality of the models are better when a RRT* is used. This is explained because the RRT* yields a smaller number of collisions and the robot reaches

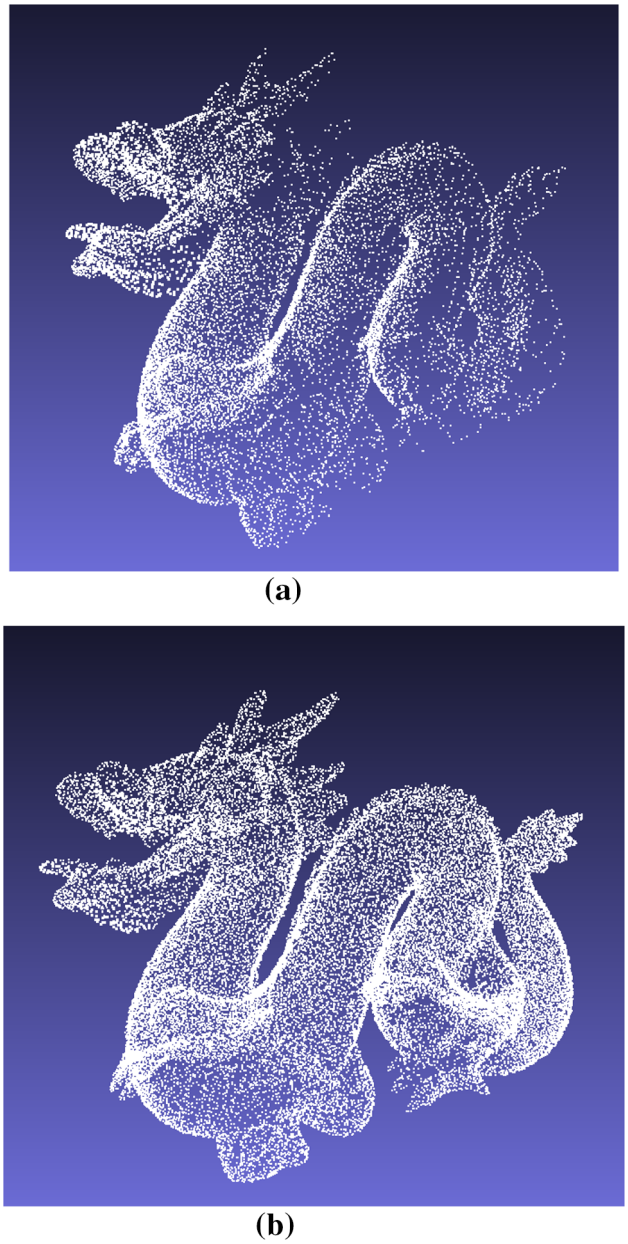


Fig. 14 Resulting models in one of the simulation experiments. The density of 3D points is larger when a RRT* is used; this is the typical result in our experiments. **a** Model obtained using a RRT and **b** model obtained using a RRT*

the sub-goals more precisely having as a consequence a larger percentage of reconstruction and a better model.

Figure 14 shows the resulting reconstructed model of the dragon object, which is the most complex of the three objects we are using in our experiments. It is clear that the density of 3D points is larger when a RRT* is used, which is the typical result in our experiments.

In the multimedia material of the paper, we have added a video illustrating the object reconstruction task and comparing the trajectories with and without noise obtained with the RRT versus the ones obtained with the RRT*.

Below, we present experiments in a more complex environment, and we also present a comparison with the paths generated by a RRT* smart algorithm in the context of object reconstruction task.

6 A more complex environment and comparison with RRT* smart

In this section, we present simulation experiments comparing the RRT, the RRT* and RRT* smart in a complex environment. The two main differences between the RRT* and the RRT* smart are the concepts of path optimization and intelligent sampling. Initially the RRT* smart searches the state space as the RRT* does, and once a path is found, the RRT* smart optimizes it by connecting the directly visible nodes in such path. This optimized path yields biasing points that are later used in the intelligent sampling stage.

To generate the trees, we proceed as in the previous sections, that is, for the RRT we stop its construction when there is node in tree that is closer to the candidate sensing configuration X_q than a given threshold. For the RRT* and the RRT* smart, we stop their construction based on the number of nodes in the tree. The RRT* number of nodes was set to 10000 nodes and for the RRT* smart was set to 2500 nodes. The difference in number of nodes is due to the fact that the RRT* smart converges faster to the optimal path than the RRT*; hence, considering a smaller number of nodes in the RRT* smart, it is possible to obtain similar cost performance as using a larger number of nodes in RRT*.

In a first series of experiments, we have included a comparison without noise between the three planning algorithms. In a second set of experiments, we have included noise in the controls. Finally, in the third experiment, we have presented an application to a slightly dynamic environments wherein the workspace is changing. It is important to note that the comparisons are about the effects that the paths generated by the three algorithms produce in an object reconstruction task and not about the problem of finding a path from a configuration A to another configuration B. The simulation experiments are done in the environment shown in Fig. 15. In this environment, there are several obstacles that generate motion and visibility constraints.

6.1 Simulation experiments without noise

We present, first, a simulation experiment without noise in the robot controls. The main performance metrics reported are the planning time, the cumulative cost and parameters related to the quality of the reconstructed object. The shown

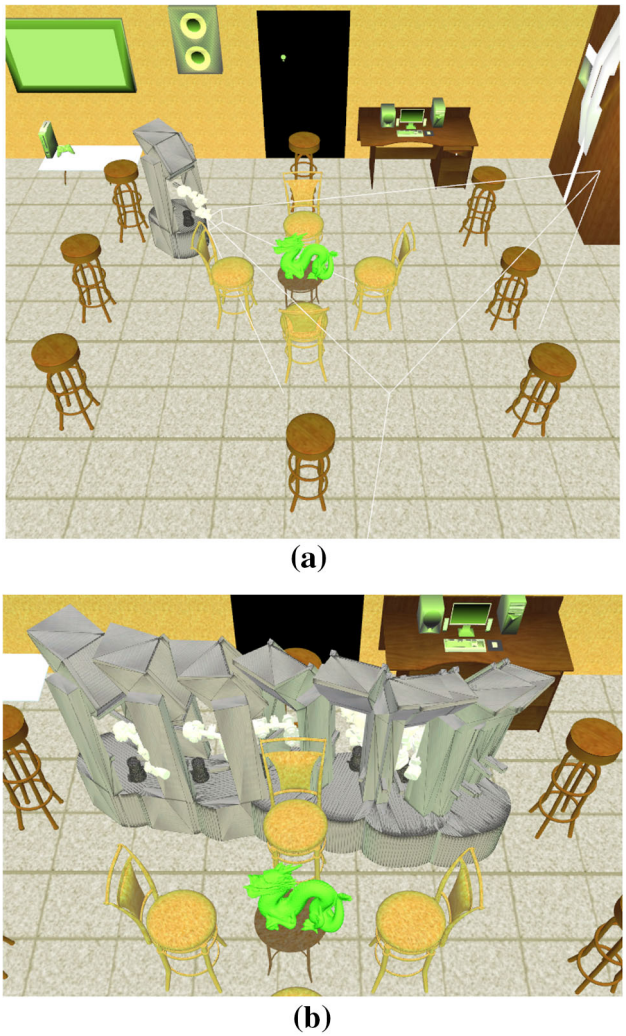


Fig. 15 A more complex environment with obstacles that generate motion and visibility constrains and a path in a narrow passage. **a** A more complex environment, and **b** a narrow passage

statistics correspond to the mean values of 5 experiments per approach.

In the experiments presented in the previous sections, the robot can get close enough to the object to avoid any visibility obstructions due to the presence of obstacles. The more complex environment shown in Fig. 15a not only induces more motion constrains, but also adds visibility obstructions that the robot can no longer avoid by just getting closer to the object, so only certain viewing directions allow the robot to see the object. Figure 15b shows the trajectory followed by the robot while traversing through a narrow passage formed by a chair and a stool.

Table 7 shows statistics concerning the planning time in seconds for each planning algorithm. The planning times show a similar tendency as in previous experiments. The time to generate paths with the RRT* is smaller than both the times used to generate paths with the standard RRT and the RRT* smart. This is because a single RRT* is used to reach several candidates to be next best view, while in the cases of the RRT and the RRT* smart, a tree is used to reach each one of the candidate views. The RRT construction was stopped when there is node in tree that is closer to X_q than a given threshold; however, if no path to X_q is found, then the tree construction is also stopped at the 10,000 nodes. The RRT does not always find such paths, so in several trials it contains 10,000 nodes, which compared to the 2500 nodes in the RRT* smart, yields the smaller time for the RRT* smart versus the RRT.

Table 7 Planning time

	RRT	RRT*	RRT* smart
Environment 3, without noise			
Dragon			
Planning time	163.1932	41.7801	83.9357

Table 8 Cumulative cost

	RRT	RRT*	RRT* smart
Environment 3, without noise			
Dragon			
Mean cumulative cost	135.8250	89.4250	79.7412
Std. cumulative cost	23.4115	6.1374	11.1536

As expected, regarding the cumulative cost, the RRT* smart has the smaller cost, followed by the RRT*, and the RRT with largest cost. See Table 8. Note that the RRT* smart includes extra procedures to shorten the path length, compared to the RRT*, which further reduce the cumulative cost.

It is interesting to note that without noise, the quality of the reconstructions is equivalent for the planning algorithms, which is assessed by the percentage of reconstruction, number of sensing locations, minimum distance and density of points. See Table 9.

6.2 Comparison among standard RRT, RRT* and RRT* smart under noise

In this section, we present experiments under the effect of noise in the robot controls. One objective of this section is to plan paths without considering the noise and later to analyze which type of paths is more robust to the addition of noise, the paths generated with a RRT, the ones generated with a RRT* or the paths obtained with a RRT* smart. Other objective is to analyze the effect over the reconstructed object according to the planner used to visit the next best views under the effect of the noise. The shown statistics correspond to the mean values of 10 experiments per approach.

Table 9 Statistics of performance criteria in environment 3 without noise

Performance criterion	RRT	RRT*	RRT* smart
Dragon			
Percentage of reconstruction	95.4945	96.3713	95.6037
Mean: number of sensing locations	23.3	22.5	23.8
Mean: minimum distance	0.0022	0.0021	0.0023
Mean: density of points	3.978640×10^6	4.015042×10^6	3.755400×10^6

Table 10 Statistical mean of planning time

	RRT	RRT*	RRT* smart
Environment 3			
Dragon			
Planning time	159.8413	40.6030	84.9180

Table 11 Cumulative cost

	RRT	RRT*	RRT* smart
Environment 3			
Dragon			
Mean c. cost	125.2542	80.1661	72.9591
Std. dev. c. cost	25.4420	25.2109	49.0747

Regarding planning time and cumulative cost (see Tables 10 and 11), we observe similar tendencies as in the experiment without noise. The RRT* has the smallest planning time, followed by the RRT* smart, and the largest time is again for the RRT. Concerning the cumulative cost, we see again that the RRT* smart has the smallest cost, followed by the RRT* and by the RRT. However, notice in Table 11 that the cumulative cost in the case with noise tends to be smaller than in the experiments without noise (see Table 8). This happens because if there is noise, collisions might occur due to inaccuracy of the controls. If a collision happens, the reconstruction procedure stops, reducing the mean cumulative cost, but also reducing the percentage and quality of the reconstruction.

Table 12 presents the mean of location and orientation error and its variance for each of the planning algorithms. The RRT has the largest location and orientation errors, followed by the RRT*, while the RRT* smart presents the smallest errors. Since the RRT* smart generates the shortest paths, it gives less opportunity to the error to affect them, which translates in smaller location and position errors. Indeed, this property of the paths generated by the RRT* smart also affects the collision rate, which again the smallest one is produced by this algorithm. See Table 13.

The statistics related to the quality of the reconstruction are shown in Table 14. The percentage of reconstruction is a bit larger for the RRT* smart compared to the RRT*,

Table 12 Statistical mean and variance of location and orientation errors

	RRT	RRT*	RRT* smart
Environment 3			
Dragon			
Mean location error	0.1204	0.0859	0.0574
Mean orientation error	0.0744	0.0544	0.02956
Variance location error	0.0044	0.0020	0.0364
Variance orientation error	0.0034	0.0018	0.0027

Table 13 Rate of collision per object and environment

	RRT	RRT*	RRT* smart
Environment 3			
Dragon			
Rate of collision	0.7	0.4	0.3

with the RRT yielding the poorest performance. The RRT* smart also yields the largest number of sensing locations, given that the paths present less collisions in the presence of noise; hence, the robot has a better opportunity to visit more sensing locations. Regarding the mean minimum distance between the closest point in the ground truth model to a given point in the reconstructed model, the RRT* smart has the best performance presenting the smallest mean values. Concerning the density points mean, the RRT* smart generated the largest mean of number of points over the volume of the voxel. Indeed, we have observed that the performance of the reconstruction procedure is heavily affected by integrating optimal planning methods, yielding a largest percentage of reconstruction and better quality. This happens because shorter paths are less affected by noise; the noise has less opportunity to deform the path, reducing the number of collisions and reaching more accurately the desired sensing state. Thus, the RRT* smart that generates the shortest paths results in the largest percentage and best quality of reconstruction. In fact, it is under the effect of noise where it is more suitable to use an optimal planning method.

Table 14 Statistics of performance criteria in environment 3 under noise

Performance criterion	RRT	RRT*	RRT* smart
Dragon			
Percentage of reconstruction	72.6142	88.6122	89.9458
Mean: number of sensing locations	14.9	17.9	18.8
Mean: minimum distance	0.0056	0.0032	0.0030
Mean: density of points	2.084575×10^6	3.352875×10^6	3.565053×10^6

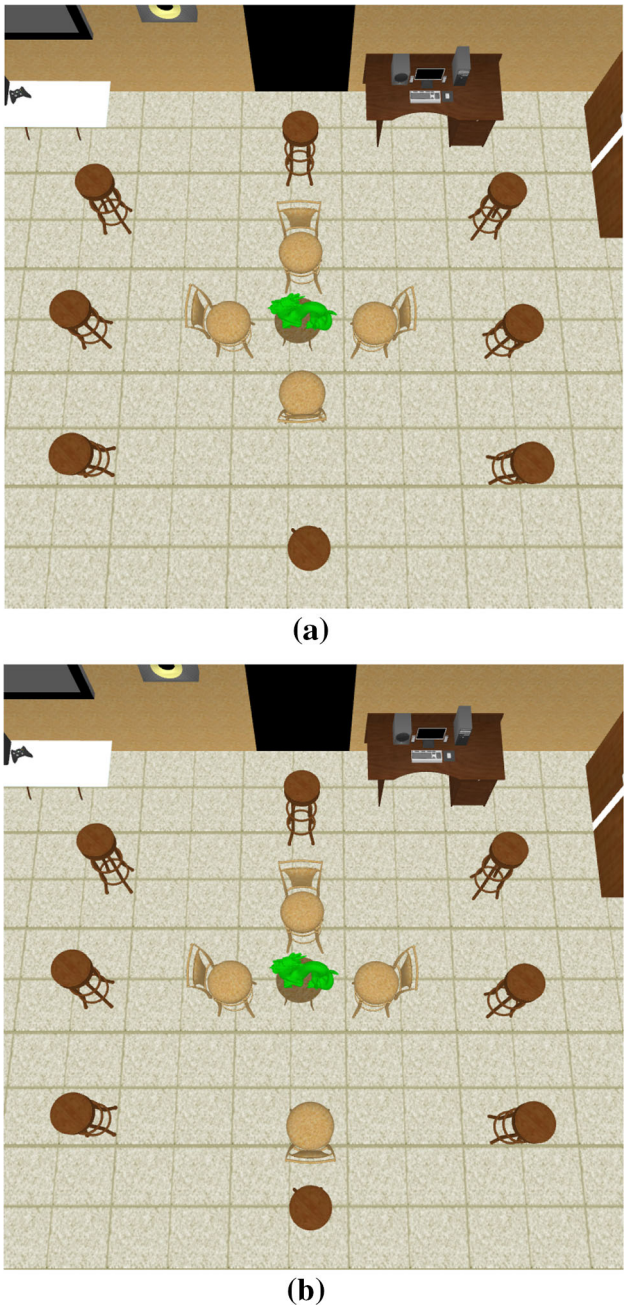


Fig. 16 An environment that changes, a chair in the bottom has changed location. **a** Obstacles, and **b** an obstacle changes location

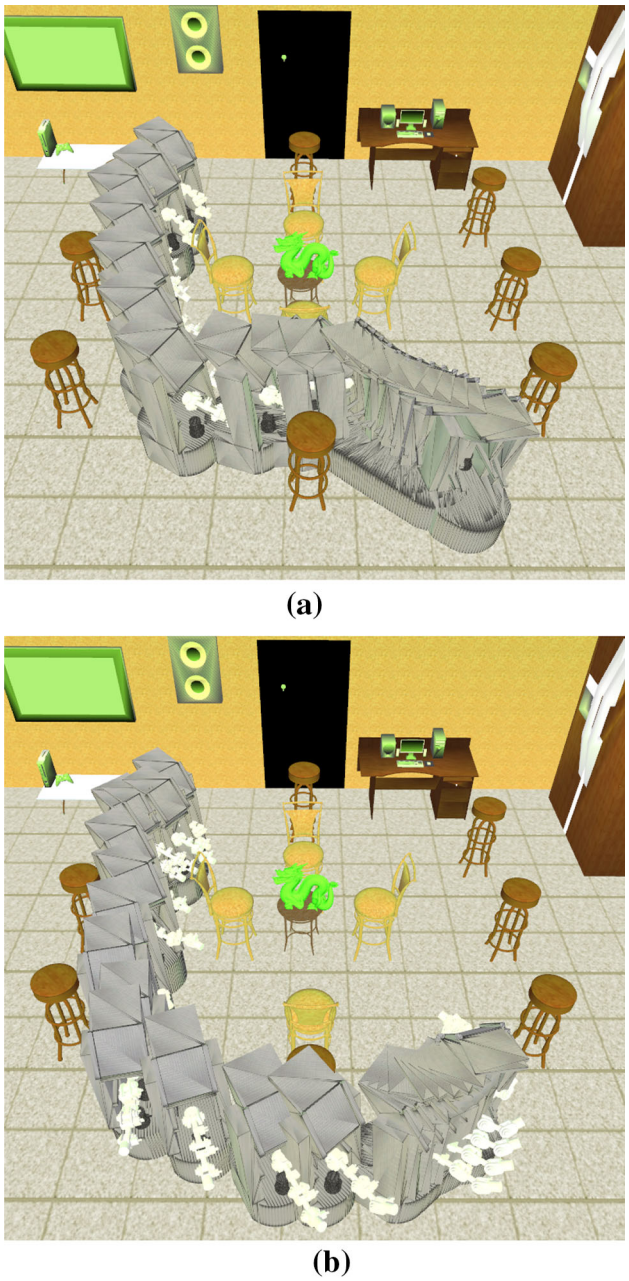


Fig. 17 The change of location of the chair yields that the robot follows another collision-free path in the tree. **a** Original path and **b** new path

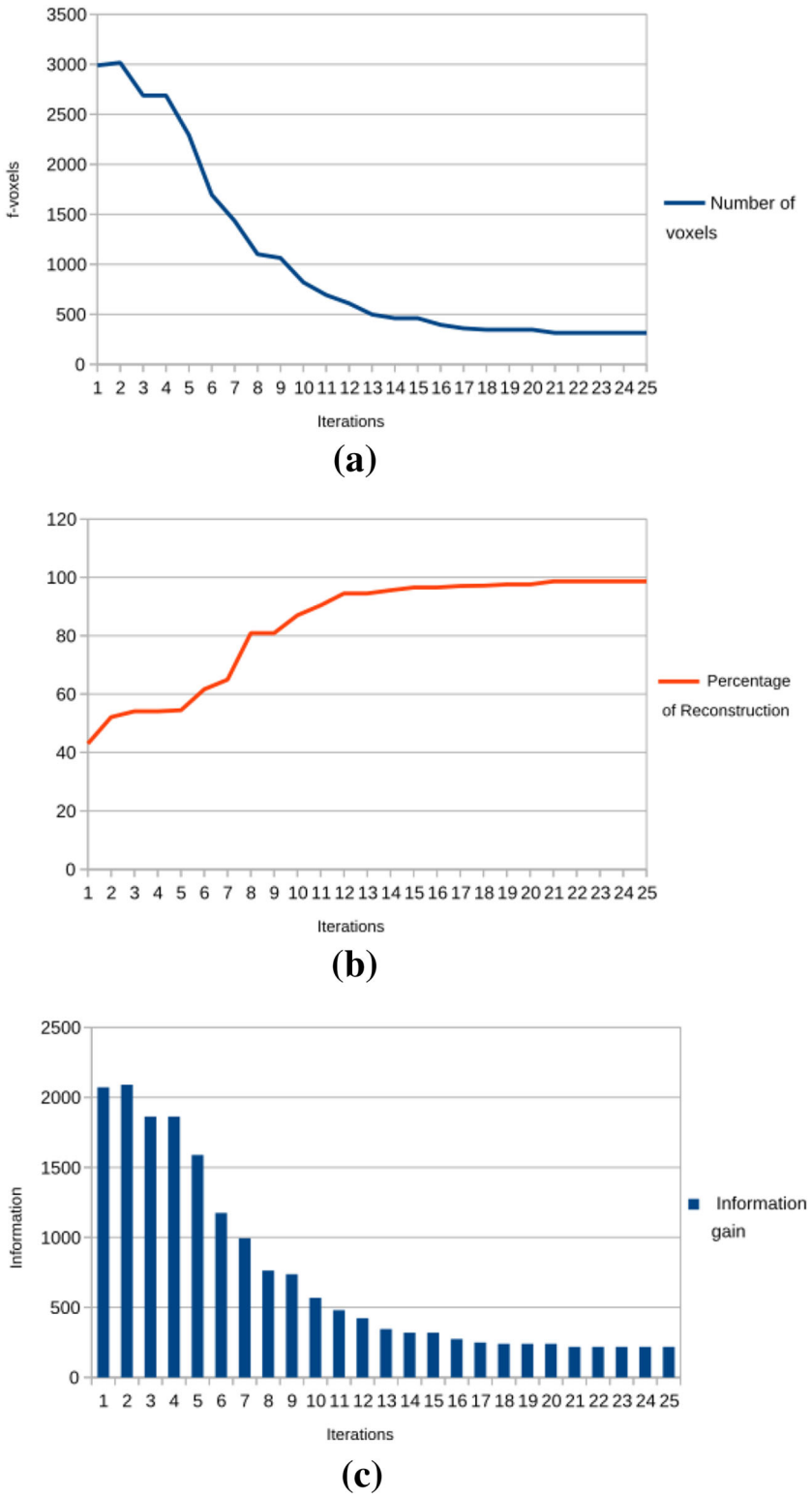


Fig. 18 Experiment with bunny object in environment 1. One can observe that as the number of iterations increases, the percentage of reconstruction also increases and the number of f-voxels and the summation of information gain decrease until at some number of iterations they remain constant. **a** Number of f-voxels, **b** percentage of reconstruction and **c** summation of information gain considering every f-voxel

6.3 Using the same tree for an obstacle that changes location

In this section, we present a simulation in which an obstacle changes location. Since we are using a single RRT* to reach several candidate sub-goals (candidate next best views) and the RRT* has the nice property of producing the optimal trajectory from the root to every node in the tree, it is possible to deal with obstacles that change location by testing every branch of the tree for collision and prune the branches that

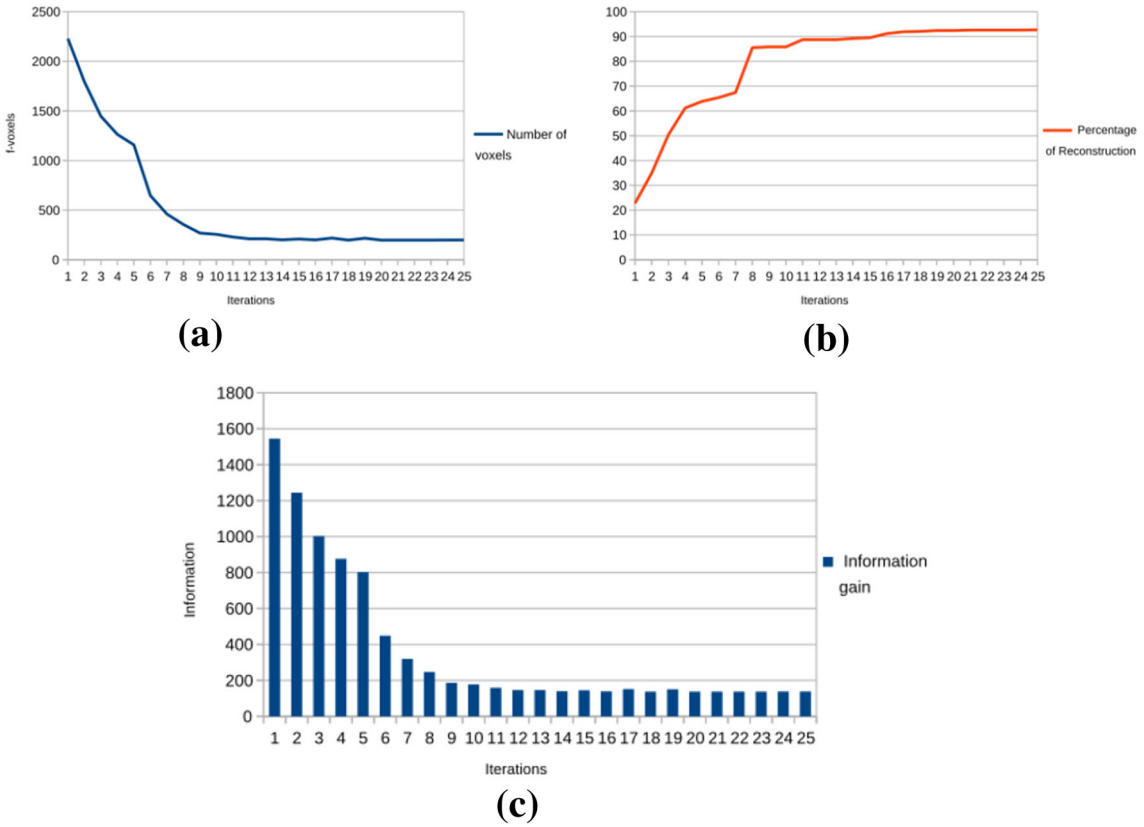


Fig. 19 Experiment with teapot object in environment 2. At some number of iterations, the percentage of reconstruction, the summation of information gain and the number of f-voxels remain constant. **a** Number of f-voxels, **b** percentage of reconstruction and **c** summation of information gain considering every f-voxel

lead the robot to a collision with the obstacle at the new location without the need to compute again the tree.

Figure 16a shows the initial distribution of obstacles, and Fig. 16b shows that an obstacle, a chair, has changed its location. Figure 17a shows the original path, and Fig. 17b shows the new path. For this example, it was possible to find a new collision-free path only pruning the branches in the tree leading to a collision and querying the tree again. In more complex cases, it is possible to prune the branches resulting in a collision and to add new branches to the tree by keep sampling; hence, adding more nodes, that is, it is also possible to repair the tree using re-planning without computing the whole tree from scratch.

Below, a stopping probabilistic test for the object reconstruction task is proposed together with a experimental evaluation of the proposed test.

7 Proposed stopping probabilistic test and experimental evaluation

In this section, we propose a way to determine when it is convenient to terminate the reconstruction task.

An *occplane* is the frontier between the visible and occluded space [11]. In [4], the voxels are labeled in 3 classes: unknown, free and occupied. In our method, at the beginning of the reconstruction process all the voxels are labeled as unknown. There are voxels that belong to the interior of the object to be reconstructed; hence, they are never sensed. Note also that because of the shape of the object or the presence of obstacles around the object, there might be a portion of the surface of the object that cannot be sensed by the sensor field of view.

Let us define an f-voxel. An f-voxel belongs to the class unknown and has an adjacent voxel labeled as free; those voxels delimit an occplane. In our simulation experiments, we have studied the behavior of the f-voxels as the reconstruction process progresses, and we have noticed that the number of f-voxels remains constant when all the object surface that is possible to be sensed is perceived. Note that having a constant number of f-voxels is exactly the same as having a constant number of free and occupied voxels. Also, we know that if the number f-voxels is zero, then the object has been totally reconstructed.

The information gain of a voxel vox is given in Eq. 14 [5]. The summation of the information gain for every f-voxel can

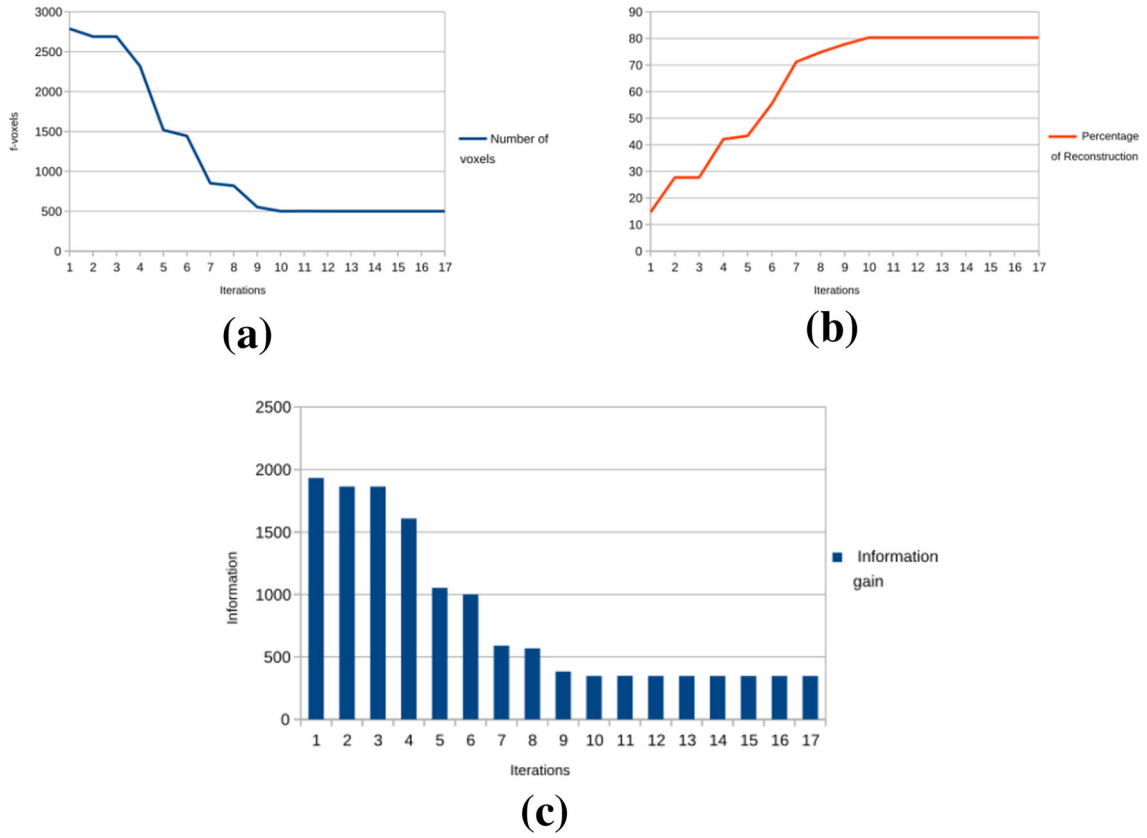


Fig. 20 Experiment with dragon object in environment 2. Again at some number of iterations the percentage of reconstruction, the summation of information gain and the number of f-voxels remain constant. **a** Number of f-voxels, **b** percentage of reconstruction and **c** summation of information gain considering every f-voxel

be used as an alternative way to estimate the progress of the reconstruction task.

$$I(vox) = -P(vox)\ln(P(vox)) - \overline{P}(vox)\ln(\overline{P}(vox)) \quad (14)$$

Figure 18 shows the number of f-voxels, the summation of the information gain for every f-voxel and the percentage of reconstruction for the bunny object, in environment 1, as the number of sensing operations (iterations) increases. Figure 18a shows the number of f-voxels. Figure 18b shows the percentage of reconstruction, and Fig. 18c shows the summation of the information gain for every f-voxel. Figures 19 and 20 show the number of f-voxels, the summation of the information gain for every f-voxel and the percentage of reconstruction for the teapot and dragon objects, respectively, in environment 2.

A probabilistic test to stop the reconstruction process can be based on the model given by Inequality (15) [15].

$$m \geq \frac{\log(\alpha)}{\log(1-\epsilon)} \quad (15)$$

If m sensing operations are done independently, the probability that m consecutive sensing operations does not cover an unseen ϵ portion of the object is $P = (1-\epsilon)^m$. In this analysis, ϵ can be thought as the percentage of the object that has not been seen yet plus the percentage of the object that cannot been seen (voxels that belong to the interior of the object or voxels that the sensor field of view cannot cover). This probability can be bound with a value α; then, $(1-\epsilon)^m \leq \alpha$. Hence in Eq. 15, m corresponds to the number of sensing operations needed to be certain with confidence $1-\alpha$ to sense $1-\epsilon$ of the portion of the box containing the object.

This model can be used in two different scenarios: one is required to know the percentage of voxels which are never visible to the sensor (voxels that belong to the interior of the object or voxels that cannot be covered, for instance, because of the arm geometry). If that information is available, then ϵ is set to that amount and m is computed only once at the beginning of the reconstruction process.

In a second scenario, the percentage of voxels which are never visible is unknown. In that case, if the portion $1-\epsilon$ changes between two consecutive operations, then m is recomputed. The reconstruction finishes with certainty $1-\alpha$

when the portion $1-\epsilon$ remains constant for m sensing operations. Two ways to know whether or not $1-\epsilon$ changes between two consecutive sensing operation is by counting the number of f-voxels at each sensing operation or by computing the summation of the information gain. If the number of f-voxels or the summation of the information is constant or the variation is smaller than a threshold, then $1-\epsilon$ is also constant.

This stopping test shall always terminate the reconstruction task, provided that all the object is perceived or all the portion of the object that can be perceived is covered with the sensor field of view.

We have done several simulation experiments with the teapot, bunny and dragon objects (see Fig. 2), and the stopping probabilistic test delivers good results; the reconstruction is always finished when all the object surface that is possible to be sensed has been perceived.

In the next section, the conclusion of this work is presented and future work is proposed.

8 Conclusion and future work

In this paper, the effect of using sampling-based optimal motion planning techniques for the task of object reconstruction, in terms of the following performance criteria: (i) the processing time to reconstruct the object has been investigated. Criteria related to the resulting trajectories to reach sub-goals (sensing locations): (ii) the cost of the trajectories under the effect of noise, (iii) the positioning error to reach a sub-goal and (iv) the rate of collisions. Criteria related to the quality of the reconstructed model: (v) the percentage of reconstruction, (vi) the distance from each 3D point in the reconstructed model to the closest point in a ground truth model and (vii) the average density of 3D points per voxel in the reconstructed model.

Based on those criteria, the results of the reconstruction task using rapidly exploring random trees (RRT) approaches are compared, more specifically, RRT* smart versus RRT* versus standard RRT. It has been observed that the performance of the reconstruction procedure is heavily affected by integrating optimal planning methods, yielding a largest percentage of reconstruction and better quality. This happens because shorter paths are less affected by noise; the noise has less opportunity to deform the path, reducing the number of collisions and reaching more accurately the desired sensing state. Thus, the RRT* smart that generates the shortest paths results in the largest percentage and best quality of reconstruction. In fact, it is under the effect of noise where it is more suitable to use an optimal planning method.

A probabilistic test to terminate the reconstruction process has also been proposed. The results of our experiments show that the reconstruction is always finished, when all the object surface that is possible to be sensed has been perceived. As future work, it would be interesting to test the method with a real robot and verify the stopping probabilistic test with a larger variety of objects to be reconstructed.

References

1. Chen S, Li Y, Kwok NM (2011) Active vision in robotic systems: a survey of recent developments. Int J Rob Res 30(11):1343–1377
2. Cieslewski T, Kaufmann E, Scaramuzza D. (2017) Rapid exploration with multi-rotors: a frontier selection method for high speed flight. In: Proc. IEEE/RSJ int. conf. on intelligent robots and systems
3. Delmerico J, Isler S, Sabzevari R et al (2018) A comparison of volumetric information gain metrics for active 3D object reconstruction. Auton Robots 42:197. https://doi.org/10.1007/s10514-017-9634-0
4. Hornung A, Wurm KM, Bennewitz M, Stachniss C, Burgard W (2013) Octomap: an efficient probabilistic 3D mapping framework based on octrees. Auton Robots 34(3):189–206
5. Isler S, Sabzevari R, Delmerico J, Scaramuzza D (2016) An information gain formulation for active volumetric 3D reconstruction. In: Proc. IEEE int. conf. on robotics and automation, pp 3477–3484
6. Karaman S, Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. Int J Rob Res 30(7):846–894
7. Kavraki LE, Svestka P, Latombe J-C, Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Rob Autom 12(4):566–580
8. Khalfaoui S, Seulin R, Fougerolle YD, Fofi D (2013) An efficient method for fully automatic 3d digitization of unknown objects. Comput Ind 64(9):1152–1160
9. Kriegel S, Rink C, Bodenmller T, Suppa M (2013) Efficient next-best-scan planning for autonomous 3d surface reconstruction of unknown objects. J Real-Time Image Process 1–21
10. LaValle SM, Kuffner JJ (2001) Randomized kinodynamic planning. Int J Rob Res 20(5):378–400
11. Lozano Albalate MT, Devy M, Sanchiz Marto JM (2002) Perception planning for an exploration task of a 3D environment. In: Proc. int. conf. on pattern recognition, pp 704–707
12. Nasir J, Islam F, Malik U, Ayaz Y, Hasan O, Khan M, Muhammad MS (2013) RRT*-SMART: a rapid convergence implementation of RRT*. Int J Adv Rob Syst 10(7)
13. Noreen I, Khan A, Habib Z (2016) A comparison of RRT, RRT* and RRT*—smart path planning algorithms. Int J Comput Sci Netw Secur 16(10)
14. Potthast C, Sukhatme G (2014) A probabilistic framework for next best view estimation in a cluttered environment. J Vis Commun Image Represent 25(1):148–164
15. Sarmiento A, Murrieta-Cid R, Hutchinson S (2005) A sample-based convex cover for rapidly finding an object in a 3-D environment. In: Proc. IEEE int. conf. on robotics and automation, pp 3497–3502
16. Scott WR, Roth G, Rivest JF (2003) View planning for automated three-dimensional object reconstruction and inspection. ACM Comput Surv (CSUR) 35(1):64–96
17. Song S, Jo S (2017) Online inspection path planning for autonomous 3d modeling using a micro-aerial vehicle. In: IEEE international conference on robotics and automation, pp 6217–6224, 29 May–3 June, Singapore, Singapore
18. Song S, Jo S (2018) Surface-based exploration for autonomous 3D modeling. In: IEEE international conference on robotics and automation, Brisbane, Australia

19. Srinivasan Ramanagopal M, Nguyen APV, Ny J Le (2018) A motion planning strategy for the active vision-based mapping of ground-level structures. IEEE Trans Autom Sci Eng 15(1):356–368
20. Torabi L, Gupta K (2012) An autonomous 9-dof mobile-manipulator system for in situ 3d object modeling. In: Proc. IEEE/RSJ int. conf. on intelligent robots and systems, pp 4540–4541
21. Torabi L, Gupta K (2012) An autonomous six-dof eye-in-hand system for in situ 3d object modeling. Int J Rob Res 31(1):82–100
22. Vasquez-Gomez JI, Sucar LE, Murrieta-Cid R (2014) View planning for 3D object reconstruction with a mobile manipulator robot. In: Proc. IEEE/RSJ int. conf. on intelligent robots and systems, pp 4227–4233
23. Vasquez-Gomez JI, Sucar LE, Murrieta-Cid R (2017) View/state planning for three-dimensional object reconstruction under uncertainty. Auton Robots 41(1):89–109