Motion Planning Strategy for Finding an Object with a Mobile Manipulator in Three-Dimensional Environments

Judith Espinoza[†], Alejandro Sarmiento[‡], Rafael Murrieta-Cid[†,*], Seth Hutchinson[‡]

† *Centro de Investigación en Matemáticas CIMAT, México*
{*jespinoza,murrieta*}*@cimat.mx*

‡ *Beckman Institute*
University of Illinois, USA
alexsarmiento@google.com, seth@uiuc.edu

Abstract

In this paper, we investigate the problem of minimizing the average time required to find an object in a known 3-D environment. We consider a 7 degrees of freedom mobile manipulator with an "eye-in-hand" sensor.

In particular, we address the problem of searching an object whose unknown location is characterized by a known probability density function (pdf). We present a discrete formulation, in which we use a visibility-based decomposition of the environment. We introduce a sample-based convex cover to estimate the size and shape of visibility regions in 3-D. The resulting convex regions are exploited to generate trajectories that make a compromise between moving the manipulator base and moving the robotic arm. We also propose a practical method to approximate the visibility region in 3D of a sensor limited in both range and field of view. The quality and success of the generated paths depend significantly on the sensing robot capabilities. In this paper, we generate searching plans for a mobile manipulator equipped with a sensor limited in both field of view and range. We have implemented the algorithm and present simulation results.

Keywords: **Search, 3-D Visibility, Mobile Manipulator Robot, Path Planning**

1 Introduction

In this work we investigate the problem of finding a static object. Our goal is to make a robot find the object as quickly as possible on the average. In our scheme, it is important to gain as much new information and in the shortest time as possible. This could be very useful in applications in which the time assigned to the task is limited or not completely known. We present a discrete formulation, in which we use a visibility-based decomposition of the environment to convert the problem into a combinatorial one.

In this paper, we consider two different robot sensor capabilities. First, we assume a robot equipped with an omnidirectional sensor, second we assume a robot equipped with a sensor, whose visibility region is shaped as a frustum.

We believe that our method has a wide range of applications, from finding a specific piece of art in a museum to search and detection of injured people inside a building.

Our search problem is related to art gallery problems, coverage and exploration. The traditional art gallery problem is to find a minimal placement of guards such that their respective visibility regions completely cover a polygon [1]. A variation of the art gallery problem, in which the task is to find

*Corresponding author

the minimal length path that covers the polygon is known as the shortest watchman tour problem [2]. This is not exactly our problem, since in [8, 12] we have shown that a trajectory that minimizes the distance traveled might not minimize the expected value of the time to find an object along it.

In coverage problems (e.g., [3, 4]), the goal is usually to sweep a known environment with the robot or with the viewing region of a sensor. In this problem, it is often desirable to minimize sensing overlap so as not to cover the same region more than once. Our problem is related to the coverage problem in the sense that any complete strategy to find an object must sense the whole environment.

In exploration problems (e.g. [5, 6]), unlike the problem presented in this paper, the environment is not known a priori. These problems, usually involve the generation of a motion strategy to efficiently move a robot to sense and discover its environment. In these problems for the robot to move to an unexplored area, a navigation problem must be solved. For instance, in [7] the authors propose algorithms for sensor-based collision-free motion planning for articulated robot arms with an Eye-in-Hand sensor.

In this work, we *propose a solution to the "where to look" part of the object finding problem* based on three main procedures: 1) We approximate visibility in 3D using a sampling method over the workspace. 2) We generate candidate sensing configurations over the robot configuration space also using sampling. We select a set of sensing configurations having two properties: a low cardinality of sensing configurations and a small distance separating them. Both properties are convenient for diminishing the average time to find the searched object. 3) We determine an order for visiting sensing configurations using a heuristic. Our heuristic for determining a visit order consists of a utility function defined as the ratio of the probability mass of seeing the object over a cost for the robot for reaching a sensing configuration.

The work we report here represents a combination and an extension of our previous work [8, 9, 10, 34]. In [8], we presented an approach to the problem of searching an object sensing at discrete locations in a 2D environment. In [9], we investigated the problem of finding an object in a 3D environment *for the case of a point robot*. In [9], we introduced a probabilistic sampling method to decompose the workspace into convex regions. In [10], we presented motion strategies for searching an object, with a mobile manipulator equipped with an *omnidirectional* "eye-in-hand" sensor. In this paper we extend our work to a mobile manipulator equipped with a sensor *limited in both field of view and range*. Some related work by the authors, where the robot senses continuously as it moves in a 2D workspace is described in [11, 12].

Our main contributions are: (1) We propose a probabilistic sampling method to decompose the workspace into convex regions in 3D. We also propose a practical and fast method to approximate the visibility region in 3D of a sensor limited in both range and field of view. (2) We propose a probabilistic control that based on how well the workspace has been covered decides when to stop the sampling generation. (3) We investigate the use of different metrics to quantify the path's cost. (4) We propose a method for selecting a set of sensing configurations having convenient properties for diminishing the average time to find the searched object. (5) We determine an order for visiting sensing configurations using a utility function.

Current technology makes feasible the assumption of equipping a mobile manipulator with an "eye-in-hand" sensor. For instance, using the very small omnidirectional camera described in [13].

The remainder of the paper is organized as follows. In Section 2, we give a formulation of our problem. Then, in Section 3, we present a general overview of our approach. In Section 4, we briefly describe an algorithm to efficiently find an order to visit sensing configurations. In Section 5, we present a sample-based approach to divide a 3D workspace into convex regions and show how to use them to generate trajectories. In Section 6, we study the use of different metrics to quantify the cost of the paths. In Section 7, we present a motion strategy for searching for an object with a mobile manipulator robot equipped with a sensor limited in both field of view and range. In Section 8, we

present simulation results considering different scenarios. Finally, in Section 9, we present some final remarks and future research directions.

2 Problem Definition and Notation

This work addresses the problem of minimizing the expected value of the time to find an object in a known 3-D workspace with a mobile manipulator with 7 degrees of freedom (DOFs).

Roughly speaking, we define the problem of searching for an object as follows: Given one mobile manipulator robot with sensing capabilities, a 3D environment and an object somewhere in the world, develop a motion strategy for the robot to find the object.

The workspace W is modeled as a set of polyhedrons. The obstacles generate *both motion and visibility constraints*. Nevertheless, we need to find collision-free paths to move the robot. Our main interest is finding a static object as quickly as possible on average. This adds a new aspect to the motion planning problem.

In our formulation, we assume that the environment is known, but that we do not have information about the location of the object being searched. Since there is no reason to believe that one object location is more likely than another, we assign equal values to every location. This is equivalent to defining an uniform probability density function (pdf) modeling the object location. Indeed, this assumption is already known in the literature and it is called the principle of insufficient reason [14]. We believe this reasoning is general given that we do not need to assume a relation between a particular type of object and its possible location (for instance, balloons are floating but shoes lie on the ground), which could reduce the scope of the applications.

Let $C(r)$ denote a convex region in a 3-D environment, where r indexes the region label. Note that all points inside $C(r)$ can be connected by a clear line of sight from any point $p(x, y, z)$ inside $C(r)$. The interior of the workspace $int(W)$ is the free space inside the 3D environment. There is a set $\{C(r)\}$ computed so that the union of all $C(r)$ covers the whole environment, that is
$\bigcup_r C(r) = int(W)$.

The robot senses the environment at discrete configurations q_i (also known as *guards*, from the art gallery problem [1]). Let us call $V(q_i)$ the visibility region associated to the limited sensor. Since the pdf modeling the object location is uniform, the probability of the object being in any subset $V(q_i) \subseteq int(W)$ is proportional to the volume of $V(q_i)$. For an omnidirectional sensor we approximate its visibility region assuming $V(q_i) = C(r)$ if the "eye-in-hand" sensor is inside $C(r)$.

We describe the route followed by the robot as a series of sensing configurations $q_{i,k}$ that starts with the robot's initial configuration and includes every other configuration once. While q_i refers to a configuration, $q_{i,k}$ refers to the *order* in which configurations are visited. That is, the robot always starts at $q_{i,0}$, and the k^{th} configuration that the robot visits is referred to as $q_{i,k}$.

For any route R, we define the time to find the object T as the time it takes to go through the configurations – in order – until the object is first seen.

Our goal is to find the route that minimizes the expected value of the time it takes to find the object

$$E\left[T|R\right] = \sum_j t_j P\left(T = t_j\right) \tag{1}$$

where

$$P\left(T = t_j\right) = \frac{Volume\left(V(q_{i,j}) \setminus \bigcup_{k<j} V(q_{i,k})\right)}{Volume(int(W))}. \tag{2}$$

Here, t_j is the time it takes to the robot to go from its initial configuration – through all sensing configurations along the route – until it reaches the j^{th} *visited* configuration $q_{i,j}$, i refers to the label

(identifier) of the configuration. Since the robot only senses at specific configurations, $P(T = t_j)$ is the probability of seeing the object for the first time from configuration $q_{i,j}$. The probability of seeing the object for the first time from configuration $q_{i,j}$ is proportional to the volume visible from $q_{i,j}$ minus the volume already seen from configurations $q_{i,k}$, $\forall k < j$ as stated in equation 2.

3 Approach Overview

This work builds on our previous research on object finding with mobile robots. In [8] we proposed an approach to solve this problem in a 2-D polygonal environment. The basic idea was to define a set of sensing locations from which the whole environment is covered. We then defined a graph over these locations. We shown that even the problem of minimizing the expected value of the time to find an object in a 2-D polygonal workspace with a point robot is NP-hard [8, 12]. Therefore, we propose the heuristic of a utility function defined as the ratio of a gain over a cost. We use this utility function to drive a greedy algorithm in a reduced search space that is able to explore several steps ahead without incurring a too high computational cost. The present work addresses the same problem but in a 3D workspace with a mobile manipulator robot.

Due to the computational complexity of the tasks we have to solve and integrate in a motion strategy, we propose approximated solutions based on sampling and heuristics, which aim to diminishing the expected value of the time for finding an object.

In general, the expected value of some criterion depends on two main factors: 1) the value of the random variable, quantified in a given metric, and 2) the probability mass associated to the random variable. In our search problem, the expected value of the time to find an object depends on two main factors: 1) the *cost* of moving the robot between two configurations, quantified in a given metric, and 2) the probability mass of seeing the object, which is equivalent to the *gain*.

For a 7 degrees of freedom mobile manipulator, we use known algorithms to compute the shortest paths (with respect to a chosen metric) to move a robot between configurations, –cost.

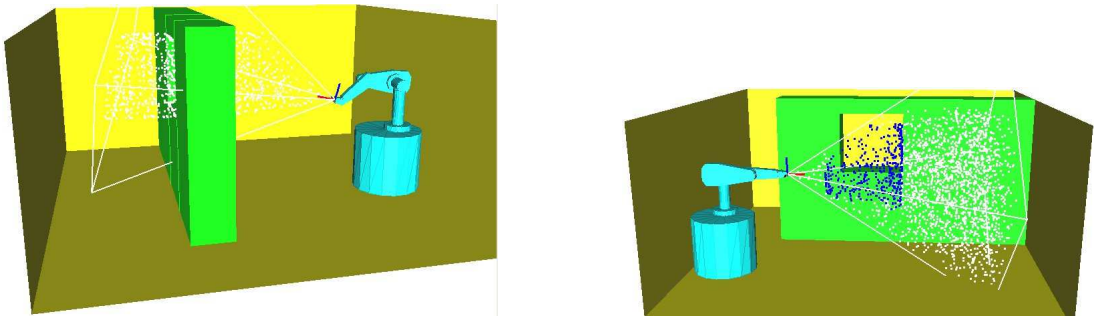
We investigate the use of different metrics to quantify the cost of the paths. We generate, metric-dependent trajectories, that is, trajectories that find a compromise between moving the base or moving the robotic arm (see Section 6). Having several means to measure the cost of reaching a sensing configuration q_i is interesting, since, depending on the user requirements, it can be applied to minimize energy, time, etc. Furthermore, in our problem, it can be used to coordinate both the translation of the robot base and the rotations of the robot base and arm to save time for reaching a given sensing configuration.

To compute the probability mass of seeing the object in a 3D environment, –gain, we need to calculate the size and shape of visibility regions and their intersections, but the computational complexity of such a task in a 3D environment populated with obstacles is very high [15]. For this reason, we decided to use a sampling scheme to calculate a lower bound on the visibility region of any point. The proposed algorithms are not deterministic, so we provide probabilistic bounds on their performance.

We also propose a sample-based convex cover algorithm that allows us to divide the environment into overlapping convex regions (see Section 5.2) and also to estimate the volume of each piece.

Using our convex cover decomposition, we propose motion strategies to our search problem, for the mobile manipulator with an "eye-in-hand" sensor like the one shown in Fig. 1.

In the case of a robot equipped with an omnidirectional sensor, our searching protocol is as follows: the robot always starts at a particular region in $\{C(r)\}$ (associated to the starting configuration) and visits the other regions as time progresses. The robot follows the shortest paths for some given metric between them. Note that the convex cover gives flexibility to where to place the sensor; any configuration q_i which places the sensor inside $C(r)$ is a valid candidate for an omnidirectional sensor.



Mobile manipulator robot and its visibility region (a) Set $\{s(t)\}$ (b)

Figure 1: Visibility Region and Set $\{s(t)\}$

For the case of a sensor limited in both field of view and range, first the whole environment is divided into a set of convex regions $\{C(r)\}$, then each convex region is covered with the sensor frustum denoted by $\mathcal{F}$. We establish a route to cover the whole environment by decomposing the problem into two parts: First an order to visit convex regions $C(r)$ is established. Second, sensing configurations in a configuration space $\mathcal{C}$ of 7 dimensions are generated to collectively cover each convex region, the sensing configuration are linked in a graph and perform a graph search to generate the search trajectory. The order to visit sensing configurations is established based on our utility function.

Finally, the robot moves for placing its sensor at every q_i to cover each convex region $C(r)$. More details are given in Section 7.

4 Choosing an Ordering to Visit Sensing Configurations

Below, we briefly describe our algorithm to find an order to visit sensing configurations.

Utility function and efficient algorithm: We propose a greedy algorithm, called *utility greedy*, that maximizes a utility function. This function measures how convenient it is to visit a determined configuration from another, and is defined as follows:

$$U\left(q_k, q_j\right) = \frac{P\left(q_j\right)}{Time\left(q_k, q_j\right)}. \tag{3}$$

This means that if a robot is currently in q_k, the utility of going to configuration q_j is directly proportional to the probability of finding the object there and inversely proportional to the time it must invest in traveling. A robot using this function to determine its next destination will tend to prefer configurations that are close and/or configurations where the probability of seeing the object is high. Notice that $P\left(q_j\right)$ is equal to $P\left(T = t_j\right)$ defined in equation 2.

The utility function in (3) is sufficient to define a 1-step greedy algorithm. At each step, simply evaluate the utility function for all available configurations and choose the one with the highest value. This algorithm has a running time of $O\left(n^2\right)$, for n configurations. However, it might be convenient to explore several steps ahead instead of just one to try to "escape local minima" and improve the quality of the solution found.

The final algorithm to find an order to visit sensing configurations is as follows (for more details see [8, 12]):

(1) For the last configuration along the current solution (initially just the robot starting configuration) explore the possible routes (create a tree breadth-first) until the number of nodes is of order $O(n)$.

(2) For each node that needs to be expanded, compute the set of configurations that are not strictly dominated by others and only choose those as children. This can be done with a convex hull algorithm with complexity $O(n \log n)$.

(3) When the number of nodes in the exploration tree has reached order $O(n)$, choose the best leaf according to the heuristic in (3), discard the current tree and start over with the best node as root.

5 Sample-based Sensing Strategy in 3-D

There have been many approaches that use samples to capture connectivity of high dimensional spaces, for example, [17], [18] just to name a pair. Notice that in our work we also want to connect sensing configurations, but we have an additional goal. *We are interested in representing the free space inside the workspace for searching an object*, and not only for abstracting the configuration space for path planning purposes.

Our sampling based sensing strategy is inspired by the work reported in [19] originally proposed to construct a probabilistic road-map in the configuration space. Notice that our work is different in several aspects. First, as it was mentioned above, we are interested in representing the workspace for searching an object, and not only for abstracting the configuration space. Second, the technique reported in [19] throws away "useless" samples, but we keep all of them since we need to estimate the size of the volume of the convex regions. Finally, we can determine a clear-cut threshold of when it is no longer useful to continue sampling.

Other approaches, more related to this work, combine approximated visibility computation and probabilistic sampling based path planning algorithms to avoid occlusions [20, 21, 22, 23]. However, they do not address the same problem presented in this paper; that is, [20, 21, 22, 23] do not aim at diminishing the expected value of the time to find and object.

5.1 Hidden Guard Set

In order to capture the size and shape of the workspace W we generate a set of independent, uniformly distributed samples S in the interior of W. Among these samples, we choose a *hidden guard set* G. A set is called a hidden guard set if it covers the environment and individual members of the set are not visible to each other. Since we are approximating W with sample points, for the set G to cover the environment, every sample in S must be visible to at least one guard in G. That is,

$$\bigcup_{g_i \in G} Vis(g_i) = S, \tag{4}$$

where $Vis(g_i)$ is the set of points in S whose line segment to g_i does not intersect the the workspace obstacle region $\left(\overline{W}\right)$. Also, since guards must not be mutually visible, $g_i \notin Vis(g_j) \quad \forall j \neq i$.

The algorithm to generate samples and the hidden guard set is as follows:

(1) Initialize the sample set S to empty.

(2) Initialize the hidden guard set G to empty.

(3) Set the counter of the number of consecutive visible samples n to zero.

(4) While n is less than a constant m,

(4.1) Generate a uniformly distributed random sample in the interior of W.

(4.2) If the sample is visible from at least one of the current guards in G, add it to the sample set S and increment n.

(4.3) Else, add it as a new guard to G and reset n to zero.

(5) Return.

It is evident that this algorithm will generate a set of hidden guards that see all the other samples and – with a large enough m – the vast majority of the environment.

To determine how well the workspace has been covered, consider an environment of unit volume like the one shown in Fig. 2. Suppose that the portion of the environment that is visible from the guard set is A and has measure $\mu(A) = 1 - \epsilon$, while B is another portion with measure $\mu(B) = \epsilon$ that is not yet visible and does not contain a single sample point.

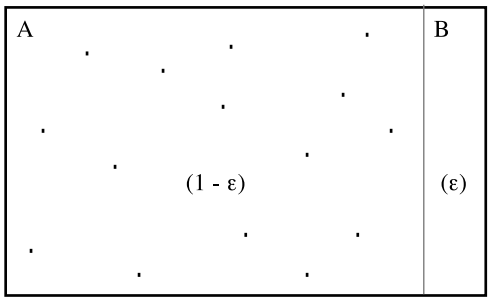


Figure 2: An environment of unit volume and the sizes of covered (A) and uncovered (B) regions

If samples are drawn independently, the probability of m consecutive points not falling into the uncovered region B is $P(A_m) = (1 - \epsilon)^m$. After m consecutive samples, it is still possible that the actual size of B is greater than ϵ, but with a large m, we can bound this probability with a small value α. For this, we determine m as follows,

$$\begin{aligned} (1-\epsilon)^m &\leq \alpha \\ m\log(1-\epsilon) &\leq \log(\alpha) \end{aligned}$$

$$m \geq \frac{\log(\alpha)}{\log(1-\epsilon)} \tag{5}$$

Therefore, choosing a large enough m we can expect with certainty $(1 - \alpha)$ that the size of the unseen region B is at most ϵ.

The algorithm above returns a set of guards that with high certainty cover the vast majority of the environment, and we can estimate the size of the visibility region of each guard (and their intersections) with a Monte Carlo approach [24].

5.2 Sample-based Convex Cover

The algorithm we propose for the sample-based convex cover is based on the idea that there is a dual between a *maximum hidden guard set* and a *minimum convex cover*. It has been proved that a minimum convex cover is an NP-hard problem [25], therefore our aim is just an efficient algorithm that

tries to generate as few convex regions as possible. We have found that in practice, our algorithm does find a minimal cardinality set in many instances.

Given our strategy of stochastically choosing a hidden guard set, there will be a neighborhood around each guard that only that particular guard can see (since a small perturbation on a guard is not likely to make it visible to the others). Likewise, provided an adequate number of samples, there will be a set of sample points that only one particular guard can see. We call this set of points, the *kernel* of the guard, and define it as follows:

$$Ker\left(g_i\right) = Vis\left(g_i\right) \setminus \bigcup_{j \neq i} Vis\left(g_j\right). \tag{6}$$

In any minimum convex cover, every convex region $C(r)$ has a set of points only contained in that particular region. Otherwise, region $C(r)$ could simply be removed and the cover would not have been minimal. Although we know there is not an exact equivalence, we use guard kernels as an approximation to these unique subsets. Thus, the main idea behind our convex cover algorithm, is that by "growing" convex regions around the guard kernels, we can generate a low cardinality convex cover.

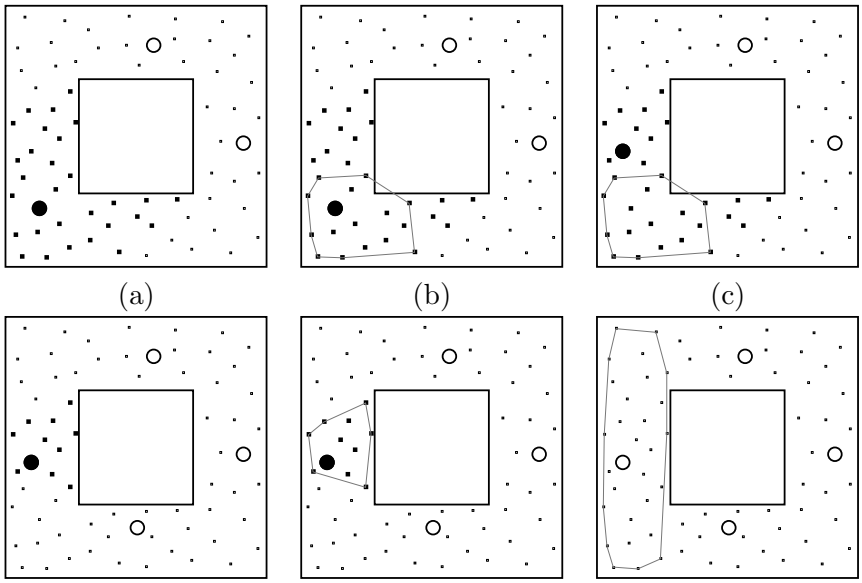


Figure 3: Convex cover algorithm: In the figure a guard as well as the point samples that are only visible by this guard (its kernel) are highlighted (they are shown with darker color)

The following description of our algorithm is depicted in Fig. 3. There, guards are circles and point samples small squares. The current guard, as well as its kernel are highlighted.

(1) Choose the guard g with the largest kernel (Fig. 3 (a)), $g = \text{argmax}_j \left\{ \left| Ker\left(g_j\right) \right| \right\}$, and compute the convex hull $Q_O = Conv\left(Ker\left(g\right)\right)$ of the kernel. For this, we start with the guard and add kernel samples to the current convex hull using the iterative algorithm described in [26]. Each kernel point adds new edges to the current hull, and we check to make sure that they do not collide with facets of the environment.

(2) If at some point, the convex hull collides with the obstacle region (Fig. 3 (b)), that is if $Q_0 \bigcap \overline{W} \neq \emptyset$. Then, we move the guard g to a random location inside the kernel but outside the current hull

(Fig. 3 (c)). This does not violate the hidden property because the kernel was only visible to this guard. Then, we re-process the rest of the samples with the algorithm described in Section 5.1. After this, at least one new guard will be generated and added to G (Fig. 3 (d)) because the original guard g can no longer see all the previous kernel points (since it was moved outside a growing convex region). This process tries to generate a maximum cardinality hidden guard set in situations where a single guard covers both sides of a convex corner. In this case, the guard is moved to one side of the corner and a new one generated on the other side. We then go to (1) and start over.

(3) If the whole kernel can be contained in a single convex region that does not intersect the environment boundary (Fig. 3 (e)), $Q_0 \bigcap \overline{W} = \emptyset$, we keep "growing' this region adding sample points in S as long as doing so does not generate a collision with the obstacle region (Fig. 3 (f)). That is, we generate the convex hull $Q_{i+1} = Conv\,(Q_i \bigcup \{s\})$ for some $s \in S$, but only if $Q_{i+1} \bigcap \overline{W} = \emptyset$. When the convex region has reached its maximum size, we remove the original guard g from G.

(4) If all the guards have been processed, terminate, otherwise continue at (1).

The previous iterative process of choosing one guard and covering its kernel with a convex region continues until all the guards have been processed and no samples remain outside at least one convex region. Every time a guard's kernel is covered and the convex region further expanded, all samples inside are "logically removed" from the environment. For this reason, algorithm is guaranteed to terminate.

This algorithm scales well from 2 to 3 dimensions, as long as a suitable convex hull algorithm and a segment-facet intersection test are implemented. Therefore, it is possible to use the same algorithm to generate a convex cover in 3D environments. To illustrate this point, Fig. 4 shows the resulting convex regions when we apply our algorithm to a 3-D environment (here, one of the boxes is on the ground while the other is "floating"). In Fig. 3 (a-h) a mesh is used to show the convex regions obtained with our algorithm. We use the iterative algorithm described in [26] to compute a convex hull in 3D. This convex hull uses the 3D point samples that cover the free space inside the 3D environment (the interior of the 3D workspace $int(W)$) to obtain the convex regions.

6 A Mobile Manipulator: Metric-dependent Trajectories

We have applied our search strategy to a mobile manipulator with an "eye-in-hand" sensor, like the one shown in Fig. 1 (a). This robot is made of an arm with four degrees of freedom (DOFs) mounted on a mobile base with three DOFs (translation and rotation in the plane). Since the first rotation of the arm (around a vertical axis) and the center of rotation for the mobile base are off-center, they are not the same axis. Thus, the entire system has a total of seven internal DOFs – two translations and five rotations. Fig 1 (a) also shows the coordinate frame for the end effector, where our sensor resides.

Notice that to place the robot's sensor in a given configuration in the 3D workspace, 6 DOF should be determined: 3 for the sensor's position, and 3 (angles) for the sensor's orientation. However, we only consider 2 angles related to the sensors orientation: rotation around the $z - axis$, and $y - axis$, a robot sensor's rotation around the $x - axis$ is not considered.

Our robot has 7 DOF, so it is a redundant system for the task of placing the sensor in a configuration given by the (x, y, z) workspace's coordinates and rotations around the $z - axis$ and $y - axis$. Thus, our robot has 2 extra DOF with respect to the 5 DOF required to define the desirable sensor configuration. In [27], the authors propose an approach for moving a mobile manipulator ensuring stability, the redundancy of the system is used to compensate system stability when the robot is unstable. In this

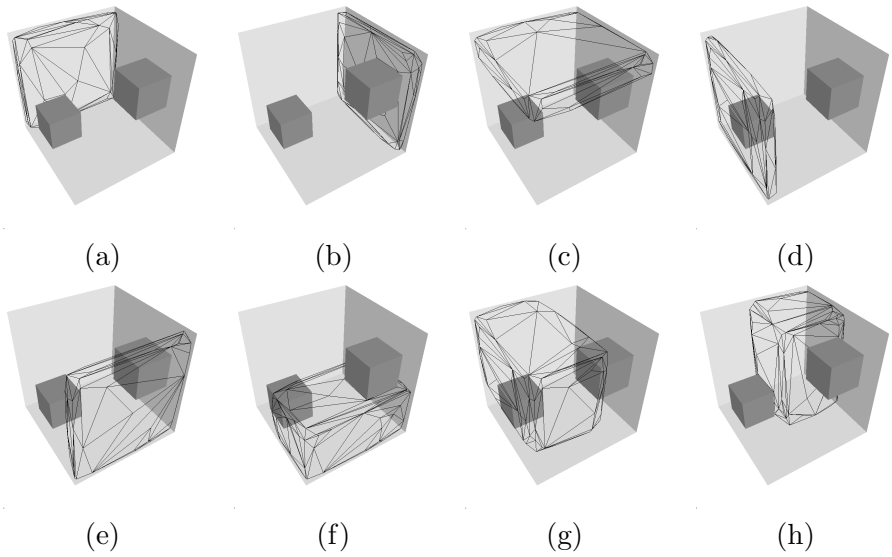


Figure 4: An 3-D environment and the eight covering convex regions generated (a) – (h)

paper, the extra degrees of freedom allow to sense a given portion of the environment from different sensing configurations (i.e. robot configurations).

The environment decomposition into regions provides flexibility on exactly where to place the end effector to sense each region – any point inside is a valid candidate. On the one hand, this simplifies the path planning problem, since now we have a *set* of goal configurations, as opposed to a single one. But on the other one, it invites a more challenging problem: given that we have more options, what is the *best* way to reach one of these regions? We are not only interested in finding a way to reach a region, but also the best way to do it.

We use our proposed algorithm (Section 4) to find the order of visiting sensing configurations. But, since the computation of the expected value of the time depends on two main factors 1) the cost of moving the robot between sensing configurations and 2) the probability mass of seeing the object (gain). We then also need to find shortest paths between configurations for a 7 degrees of freedom mobile manipulator.

We want to find shortest paths between configurations. The actual paths depend on the metric used to measure distance. One way to define the distance between two configurations X and Y in a D-dimensional configuration space is

$$\|X - Y\|_\Lambda \equiv (X - Y)^T \Lambda (X - Y), \tag{7}$$

where Λ is a diagonal matrix with positive weights $\lambda_1, \lambda_2, \ldots \lambda_D$ assigned to the different DOFs. In general, the matrix Λ is needed because joints might be not all equivalent (e.g., large links vs. small links), and also because different DOFs might not even be measuring the same movement (translation vs. rotation).

In our case, by weighting each DOF differently we can assign different priorities to the two main components of our system: the mobile base and the robotic arm. We divided the DOFs of the mobile manipulator into two groups. One group contains the two DOFs that translate the robot on the plane, and another the five rotations that further place the end effector at a specific position. This division is made without loss of generality since weights can be adjusted individually.

If we assign larger weights to one group and smaller weights to the other, it will be reflected in the

shortest paths for that metric. It is evident that these paths will have lower length when the DOFs that move the most are the ones with a lower cost. Since we can select which DOFs we would like to move the most (or the least), we can find trajectories that find a compromise between, for example, moving the base and moving the arm.

The problem is, of course, to find the shortest path to a set of configurations. Since it is inherently more complex than the classic path planning problem which only asks for any solution, it is unlikely that it can be solved by an algorithm efficiently. For this reason, we decided to simply implement a wavefront expansion [28] (called NF1). This algorithm is exponential in the number of dimension (robot degrees of freedom), but if we restrict ourselves to a *particular* robot, then it becomes polynomial.

The wavefront expansion algorithm uses the weights $\lambda_1, \lambda_2, \ldots \lambda_D$ assigned to the DOFs to expand the wave in the different dimensions, so that all configurations at the growing boundary are at the same distance from the start configuration.

The advantage of finding a shortest path is most evident when there is an option as exactly where to go; when the robot can decide on a more "convenient" goal configuration. In this case, a convenient configuration is one that can be reached by moving the low cost DOFs more than the others. For our case, the task of sensing a region can be accomplished by reaching any robot configuration having the sensor inside the selected convex 3-D region.

7 Searching for an Object with a Mobile Manipulator Equipped with a Limited Sensor

We have extended our strategies for searching motion plans to a mobile manipulator equipped with a sensor limited in both field of view and range. The visibility region of the limited sensor has the shape of a frustum, see figure 1.

We propose a method to approximate the 3-D visibility region of a limited sensor. This method has as its objective to determine which portions of the environment are visible with the limited sensor of the robot based on the point samples generated over the 3-D workspace. Our method to approximate the 3-D visibility region (frustum) of the limited sensor works as follows: first the whole environment is divided into a set of convex regions. These convex regions are used to facilitate the 3-D visibility computation of the limited sensor. Thanks to these convex regions, we avoid the use of complex data structures to update the portion of the environment that has been covered (perceived with the limited sensor). A convex region is covered if the point samples used to approximate its volume have been sensed, or equivalently if the union of the frustums associated to robot's sensing configurations has collectively covered all the point samples inside a convex region. We compute which points are visible from a sensor's configuration based on the intersection of the sensor's frustum and a given convex region.

Our motion strategy is as follows: First, an order to visit every convex region $C(r)$ is established. This order is the same as the one for an omnidirectional sensor. Once this order is planned, another order is computed to visit sensing configurations $q_{(i,r)}$ that collectively cover a specific convex region, i indexes the sensing configuration and r the label of the convex region. Both orders are computed using the algorithm described in Section 4.

The selection of sensing configurations to cover each convex region considers the portion of the convex region that has already been covered, while sensing convex regions previously visited (according to the order to visit the convex regions).

7.1 Selecting Sensing Configurations

In our method to cover each convex region, sensing configurations $q_{(i,r)}$ are generated with a uniform probability distribution in a configuration space $\mathcal{C}$ over 7 dimensions: Two coordinates (x, y) defining the robot base position and five angles. A sensing configuration $q_{(i,r)}$ is associated to a given region $C(r)$. Each convex region has associated a set $S(r)$ of point samples $s(r) \in S(r)$ originally used to compute the size and shape of a convex region. Each point sample $s(r)$ lies in the 3D space, and is defined by a 3-dimensional vector $p(x, y, z)$. We use this previously computed set of points to determine the coverage of a convex region with a limited sensor.

The need of selecting effective robot configurations can be found in several problems. For instance in [29], the authors propose an approach to detect an effective pose of a mobile manipulator for grasping an object. In our searching problem, for diminishing the expected value of the time to find an object, the set $\{q_{(i,r)}\}$ must have two properties: a low cardinality of sensing configurations and a small distance separating them.

Notice that *a set with minimal cardinality might not be the optimal one*, since the cost for reaching the configurations might be large if they are far from one another. Thus, our problem is different from the problem of only choosing the minimum number of configuration that collectively sense the whole environment: The art gallery problem. Since, even the easier art gallery problem is NP-hard [30], it is unlikely that our problem can be solved by a deterministic algorithm efficiently. For this reason, we decided to use a sampling scheme to compute $\{q_{(i,r)}\}$.

Our algorithm for selecting sensing configurations has been inspired from the algorithm presented in [31]. That method computes a "good" set of sensing locations that collectively see the whole environment. The algorithm works only for a 2D environment. The environment W is represented as a polygon. Sensing locations are generated based on uniform sampling. The method in [31] is designed to cover the boundary of the workspace ∂W. In our problem, we aim to cover the free space inside the polyhedral environment representation, and moreover we deal with a 3D environment.

In our method, the point samples lying inside the frustum associated to a sensing configuration $q_{(i,r)}$ are used to approximate the actual visibility region $V(q_{(i,r)})$. The robot configurations used to cover a convex region have the property that all of them place the sensor inside the convex region being sensed. This property allows us to approximate the visibility region of the limited sensor without complex 3D visibility computations. The visibility region of the limited sensor at configuration $q_{(i,r)}$ is approximated by:

$$V(q_{(i,r)}) = \bigcup_{s} s(r) \in int(\mathcal{F} \cap C(r)) \tag{8}$$

In figure 1 (a), the white dots are used to show the point samples $s(r) \in S(r)$ covered by the view frustum $\mathcal{F}$ and inside a convex region $C(r)$.

Notice that the sensor can be inside several convex regions at once, in such a case, the visibility region is approximated as the point samples lying simultaneously inside the frustum and the convex regions having the sensor in its interior.

As mentioned above, we use the order of visiting convex regions to select sensing configuration to cover each convex region progressively. While covering region $C(r)$, we also mark as sensed and logically remove, all samples $s(t)$ belonging to region $C(t), t \neq r$, if $s(t) \in int(\mathcal{F} \cap C(r) \cap C(t))$. It is guaranteed, that these samples are not occluded from configuration $V(q_{(i,r)})$.

In figure 1 (b) dark (blue) dots are used to show the set $S(t)$, and white dots represent the set of point samples $S(r)$ belonging to the region in which the sensor resides and inside the frustum.

A convex region $C(r)$ is totally covered if:

$$\bigcup_{s} s(r) \in int(C(r)) = S(r) \tag{9}$$

Similar to [31], we select sensing configurations based on the cardinality of its point samples. Iteratively, we select the configurations with the largest cardinality of point samples $s(r)$ until all the set $S(r)$ is sensed. The point samples associated to selected sensing configurations are logically *removed*. Thus, redundant sensing configurations, with low point samples cardinality are avoided, yielding a reduced set containing only sensing configurations with high cardinality of point samples and a small number of redundant point samples.

Additionally, in our sensing configuration sampling scheme, we reject candidate sensing configurations in whose view frustum is in collision with the robot itself, thus, avoiding occlusions generated by the robot body. We also reject sensing configurations, that produce a collision of the robot with the obstacles and robot self-collisions.

Following the order of visiting convex regions, each convex region is covered. The procedure of selection of sensing configurations described above is repeated several times for each convex region $C(r)$, yielding different sets $\{q_{(i,r)}\}$ of sensing configurations. We select one of those sets based on the average value of its graph edges. We describe this last selection below.

7.2 Connecting Sensing Configurations

Since we want to have options to move the robot between sensing configurations, and thus further reduce the expected value of the time to find the object, we connect the sensing configuration of each set $\{q_{(i,r)}\}$ into a fully connected graph with ne edges. We assign weights $w_{(i,j)}$ to the graph edges according on the metric used to measure the cost of moving the robot between sensing configurations q_i and q_j.

We select the set that minimizes the average value of the graph edges, that is $\min(\frac{1}{ne}\sum_{<i,j>} w_{(i,j)})$. At the end, we obtain a set having both: low cardinality of sensing configurations and small distances among sensing configurations.

To cover a region with a limited sensor several sensing configuration are required. The cost of moving a robot between sensing configurations can be computed using the wavefront expansion described in Section 6. However, the execution of that algorithm can take significant amount of time for a large number of sensing configurations. To reduce the running time of planning robot paths to cover a region with a limited sensor, we consider that the cost of moving the robot between sensing configurations corresponds to a straight line in the configuration space. That is, for reducing the computational time to cover the environment with a limited sensor, we *pre-estimate* the cost to move between sensing configuration as a straight line in the configuration space. To cover a single convex region the robot travels a tour, the first sensing configuration and the last one is the same.

In the motion planning problem of generating collision free paths to move between configurations, we use a lazy collision checking scheme [32]. The sensing configurations are connected in the graph without checking if a collision free path can connect them.

Since we proceed visiting convex regions one by one, it is likely to find collision free paths among configurations to cover the same convex region. Often a small region can be covered with small robot motions, and big regions offer large open space to move the robot. We postpone the collision checking until an order of sensing configurations is established. If some edge of the route is in collision then a new sub-order is generated using other edges in the graph. The new sub-route starts from the last configuration that was connected in the route, and includes all the remaining sensing configurations. Evidently, sometimes the fully connected graph splits into two connected components, if so, we use an

RRT [33] to find a collision-free path between the two components. In this later case, the cost assigned to the graph edge connecting the two associated sensing configurations is the cost of the path, which has been obtained with the RRT and checked for collision. In our experiments, we have found that this strategy significantly reduces the time to generate collision-free paths to cover convex regions. The order of visiting the regions is established based on a path of optimal cost between the regions. This optimal path is computed with the wavefront expansion algorithm described in [28].

We use the convex cover, even in the case of a robot equipped with a limited sensor (i.e. not omnidirectional). There are three main reasons to do so: First, we avoid the use of complex data structures to update the portion of the environment that has been covered. A convex region is covered if the samples used to approximate its volume have been sensed, or equivalently if the union of the frustums associated to sensing configurations has collectively covered all the samples inside a convex region. Second, we take advantage of our convex covering to postpone the collision checking until an order of sensing configurations is established. To cover a convex region, it is often possible to find a collision free path between sensing configurations by simply moving the robot in a straight line path in the configuration space. Third, the robot does not move to cover another convex region until a given convex region has been totally covered. It avoids unnecessary robot motion to visit sensing configurations far away from a previous one.

8 Simulation Results

In this section simulation results are presented. First, we present results for metric dependent trajectories. We assign different weights to the degrees of freedom of the robot. Second, we consider the case of large 3D environments, in which the cost of rotation is the same that the cost of translation. Finally, we present results for a mobile manipulator equipped with a sensor whose visibility region has the form of a frustum.

All the results presented in this paper were obtained with a regular PC running Linux OS, equipped with an Intel Pentium IV microprocessor and 512 megabytes of RAM, the processing speed of the CPU is 2.2 GHz. The programming language used to obtain our simulation result was C++.

8.1 Simulation Results for Metric-dependent Trajectories

We divided the DOFs of the robot into two groups: two translations and five rotations. First we assigned a larger weight to one group and then to the other. Fig. 5 (a-c) shows the case when rotating the links is better than translating the robot. Similarly, Fig. 5 (d-f) shows the same setup but when we seek to translate the base as opposed to rotate the links. Parts (a) and (d) show the initial configuration and the region that needs to be visited. Parts (b) and (e) show the initial (dark) and final (light) configurations. Finally, parts (c) and (f) show the actual path followed by the robot. In the example the resulting trajectories are very different. When rotations are better (Fig. 5 (c)), the robot goes straight to the obstacle and places its sensor over it. The robot must rotate the base (and compensate with the "shoulder") and also rotate the middle link (and compensate with the end effector) to be able to reach beyond the obstacle. So there are a total of four rotations in this movement. In contrast, when translation is preferred (Fig. 5 (f)), the robot does not rotate any link and simply moves around the obstacle to reach the region behind.

8.2 Simulation Results with an Omnidirectional Sensor

Figures from 6 (a) to 6 (c) show snapshots of the searching task in an indoor environment. This environment cannot be searched by using a 2D projection; notice the stairs, the windows, the lamp

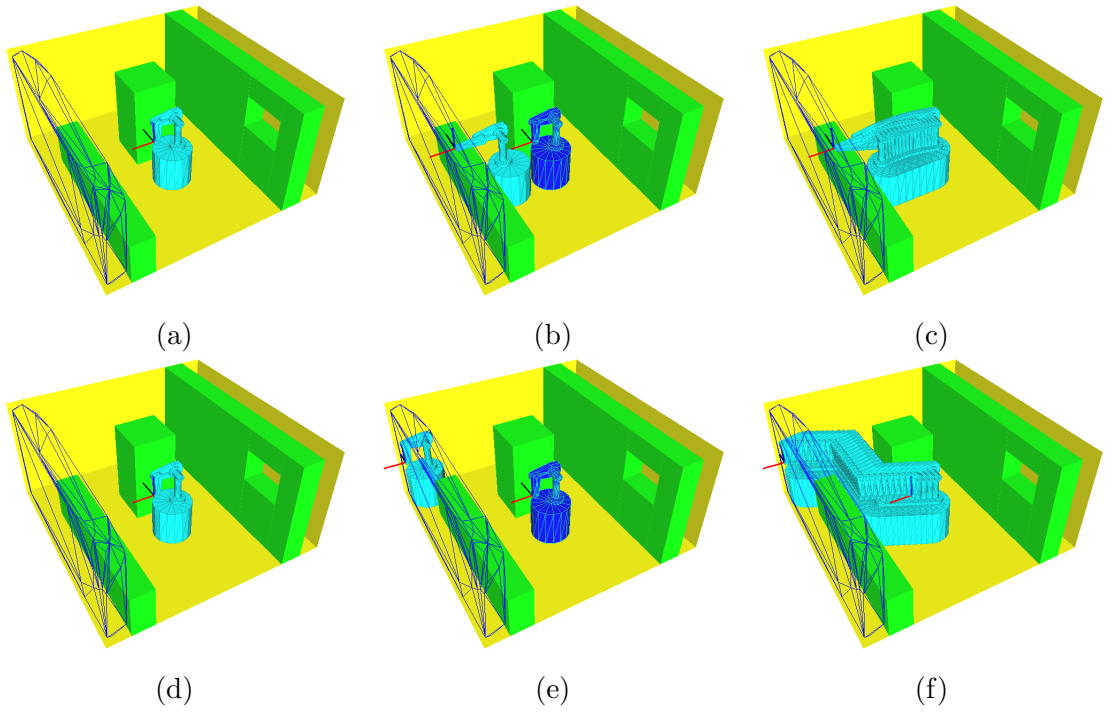


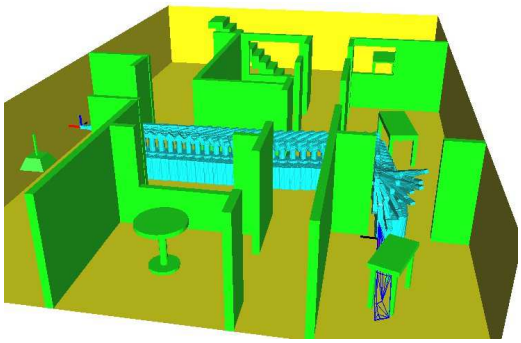
Figure 5: Visiting one convex region: Metric dependent trajectories

and the furniture. For this environment, our convex cover produced 50 convex regions. The convex cover decomposition was computed by our software in 7 minutes 39 seconds. Figure 6 (d) shows the global robot's path to sense all regions.

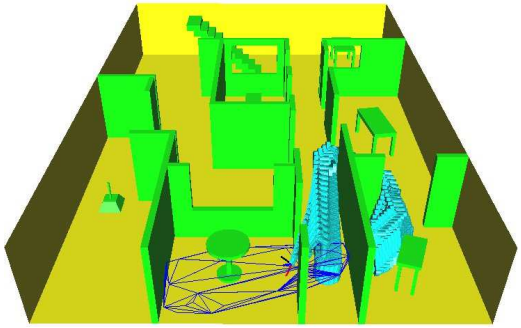
The process of computing the paths of optimal cost to move the robot between convex regions, and an order for diminishing the expected value of the time to visit the convex regions was almost two days. This example allows us to find the bottleneck limitation regarding the running time of our algorithms. This bottleneck corresponds to compute the optimal paths to move our 7 DOFs robot between sensing configurations. The optimal order for visiting convex regions is a NP-Hard problem, but we reduce the computational running time using the heuristic presented in Section 4. As a result we obtain a practical procedure with the cost of losing global optimality and obtaining only local optimality. However, another complex problem has to be confronted, the problem of computing optimal paths between pairs of sensing configurations (locally optimal) for a robot with 7 DOF and moving in a complex and large 3D environment (for instance the one shown in figure 6). The NF1 wavefront expansion algorithm [28] that we use requires considerable running time for computing all the locally optimal sub-paths to be concatenated in a global path, specially for a global path including a large number of inter-medial sensing configurations. This algorithm also strongly depends on the number of cells (resolution of the partition) in which the configuration space is divided. We use a resolution in which robot collisions with the obstacles are detected. Globally optimal search paths for this kind of environments cannot be computed with a regular PC, due to computational complexity.

8.3 Simulation Results with a Limited Sensor

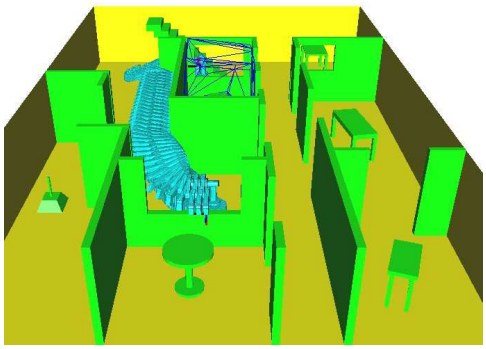
We finish our presentation of simulation results with a house type environment, which is searched with a limited sensor mounted in our 7 DOF mobile manipulator robot. Figure 7 shows snapshots of this



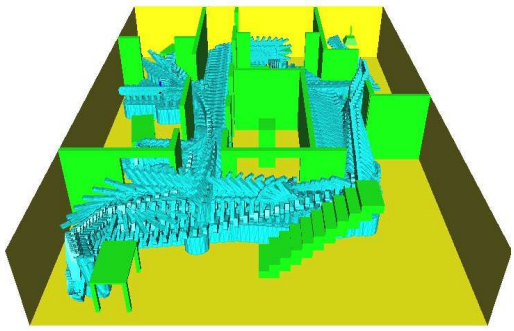
Convex region a) and the robot's path to reach region a) (a)



Convex region b) and the robot's path to reach region b) (b)



Convex region c) and the robot's path to reach region c) (c)



The global path to cover the whole environment (d)

Figure 6: House

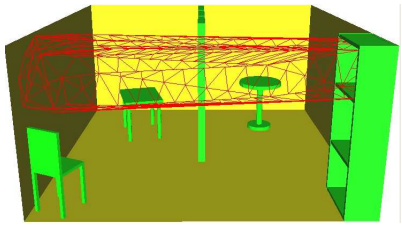
experiment. Our convex cover algorithm generated 23 convex regions and 113 sensing configurations were needed to cover the whole environment with the limited sensor.

Figures 7 (a), 7 (b) and 7 (g) show with a (red) mesh 3 convex regions, called respectively region $C(A)$, $C(B)$ and $C(C)$. Convex regions $C(A)$ and $C(B)$ are consecutive in the order to visit convex regions.

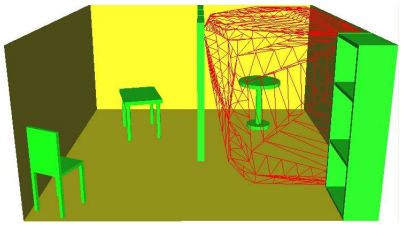
Figures 7 (c) and 7 (d) show the sensing configurations needed to collectively cover regions $C(A)$ and $C(B)$ respectively. Figure 7 (e) shows the path to move between two sensing configurations associated to region $C(A)$, this path corresponds to a straight line in a configuration space of 7 dimensions. Figure 7 (f) shows the path to move the mobile manipulator robot between the last sensing configuration of region $C(A)$ and the first sensing configuration of region $C(B)$.

Figure 7 (h) shows the 3 sensing configurations used to cover the convex region $C(C)$, which is under a table.

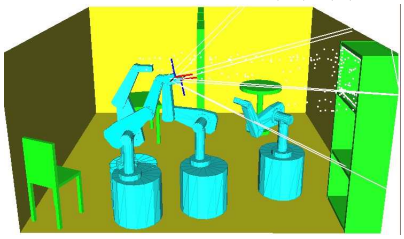
The running time of our software to generate the 113 sensing configurations and computing an order to visit them to cover the environment with a limited sensor was 2 hours and 2 minutes. This time has to be added to the time to compute the 23 convex regions, which is 1 minute and 33 seconds, and the time to compute optimal paths between every pair of convex regions, and the time to evaluate the heuristic to define an order to visit all sensing configurations, which is 13 minutes and 18 seconds. Thus, the total running time of our software to sense the whole environment with a limited sensor was 2 hours 15 minutes and 51 seconds. The expected value of the time is 36.17 units. The computational running time of our algorithm refers to the time taken by our software to generate a plan to cover the



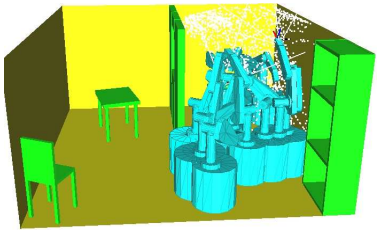
Convex Region $C(A)$ (a)



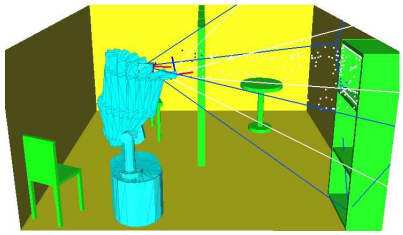
Convex Region $C(B)$ (b)



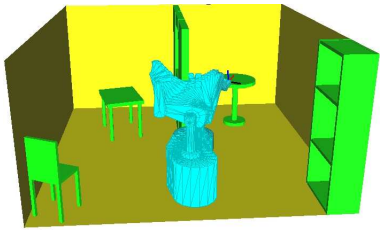
Robot configurations
to cover Region $C(A)$ (c)



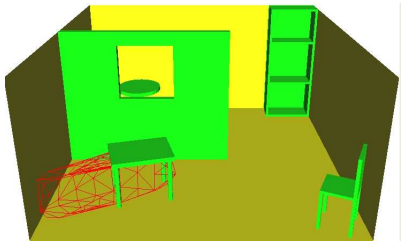
Robot configurations
to cover Region $C(B)$ (d)



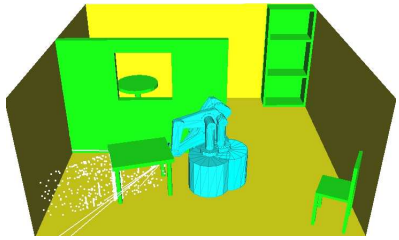
Path between two configurations to cover Region $C(A)$:
Moving the robot in a straight line in $\mathcal{C}$ (e)



Path from the last configuration in Region $C(A)$
to the first configuration in Region $C(B)$ (f)



Convex Region $C(C)$
under a table (g)



Robot configurations to
cover region under the table (h)

Figure 7: Finding an object with a limited sensor

environment and consequently to find the object. The expected value of the time refers to the average time in which the object will be found by executing that plan. So the first time refers to the generation of the plan, and the second to the performance in average of this plan when the plan is executed.

Recently, in [34], we have proposed the strategy of selecting the most important DOFs to be optimized. This strategy significantly reduces the computational running time to generate a plan. For instance, for the environment shown in figure 6, the time to generate a plan for a robot equipped with an omni-directional sensor, was reduced from almost 2 days to 1 hour and 50 minutes, and using a *robot equipped with a limited sensor*. We stress that in general, it is more complex to generate a plan for a robot equipped with a limited sensor, since in this case, the robot needs to visit more sensing configurations to cover the whole environment. The time for generating a plan, in the environment

shown in figure 6 was reduced from 2 hours and 13 minutes to only 11 minutes. More details can be found in [34].

9 Conclusions and Future Work

We addressed the problem of searching for an object with a mobile manipulator in a known 3-D environment. Due to the high computational complexity of visibility queries in 3-D, we decided to use a sample-based approach to approximate the size and shape of visibility regions. We have also studied the plan's dependency on the field of view and range of the sensor used to search the object.

In our experiments, first, we have considered an omnidirectional sensor. The decomposition into convex regions gives us flexibility on exactly where to place the sensor, and their convexity guarantees that the entire region will be visible from every point inside using an omnidirectional sensor. We have presented an approach to generate metric-dependent trajectories, that is, trajectories that compromise between moving the base and moving the robot arm. Second, we have shown that our approach can be extended to a sensor limited in both field of view and range. Unsurprising, if the sensor is limited then the robot has to move more and has to visit more sensing configurations.

Recently, we have proposed in [34], the strategy of selecting the most important DOFs to be optimized, which significantly reduces the computational running time to find the object.

Finally, in this work, we assumed that the pdf modeling the object location was uniform, i.e. the probability of finding the object in a given region was directly proportional to the size of the region. We believe that it is a good general a priori, given that to consider other types of pdf's, the type of object should be taken into account. However, in a scenario where one or more objects are sought several times, it should be possible to start the search assuming an uniform pdf and modify it according to the places where the objects were previously found. We also leave this problem for future work.

Acknowledgments: This work was partially funded by CONACYT project 106475 and NSF-CONACYT project J110.534/2006.

References

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

[2] W. P. Chin and S. Ntafos, "Optimum watchman routes," *Information Processing Letters*, vol. 28, pp. 39–44, 1988.

[3] S. Hert, S. Tiwari, and V. Lumelsky, "A terrain-covering algorithm for an auv," *Autonomous Robots*, vol. 3, pp. 91–119, 1996.

[4] E. U. Acar, H. Choset, and P. N. Atkar, "Complete sensor-based coverage with extended-range detectors: A hierarchical decomposition in terms of critical points and voronoi diagrams," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1305-1311, Maui, HI, USA, 2001.

[5] B. Tovar, L. Muñoz, R. Murrieta-Cid, M. Alencastre, R. Monroy, and S. Hutchinson, "Planning exploration strategies for simultaneous localization and mapping," *Robotics and Autonomous Systems*, vol. 54, no. 4, pp. 314–331, 2006.

[6] P. Wang and K. Gupta, View planning for exploration via maximal C-space entropy reduction for robot mounted range sensors, *Advanced Robotics*, vol.21, no.7, pp.771-792, 2007.

[7] Y. Yu and K. Gupta, Sensor-based probabilistic roadmaps: experiments with an eye-in-hand system *Advanced Robotics*, vol.14, no.6, pp.515-536, 2000.

[8] A. Sarmiento, R. Murrieta, and S. A. Hutchinson, "An efficient strategy for rapidly finding an object in a polygonal world," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1153-1158, Las Vegas, NV, USA, 2003.

[9] A. Sarmiento, R. Murrieta, and S. A. Hutchinson, "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. 3486-3491, Barcelona, Spain, 2005.

[10] A. Sarmiento, J. León-Espinoza, R. Murrieta-Cid and S. Hutchinson: A Motion Planning Strategy for Rapidly Finding an Object with a Mobile Manipulator in 3-D Environments. in Proc MICAI 2008, LNAI, Vol 5317, pages 562-572, Atizapán de Zaragoza, México, 2008.

[11] A. Sarmiento, R. Murrieta-Cid and S. Hutchinson, Planning Expected-time Optimal Paths for Searching Known Environments, in Proc *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 872-878, Sendai, Japan, 2004.

[12] A. Sarmiento, R. Murrieta, and S. A. Hutchinson, "An Efficient Motion Strategy to Compute Expected-Time Locally Optimal Continuous Search Paths in Known Environments," *Advanced Robotics*, Vol 23, No 12-13, 1533-1569, October, 2009.

[13] H. Ishiguro, T. Sogo, and M. Barth, "Baseline Detection and Localization for Invisible Omnidirectional Cameras," in *International Journal of Computer Vision*, 58(3), 209-226, 2004.

[14] P.-S. Laplace, Théorie Analityque des Probabilités. *Courceir*, Paris, 1812.

[15] J. E. Goodman and J. O'Rourke, Eds. in *Handbook of Discrete and Computational Geometry*, CRC Press, 1997.

[16] S. M. Ross, *Introduction to Probability and Statistics for Engineers and Scientists.* Wiley, 1987.

[17] 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*, vol. 12, no. 4, Jun 1996.

[18] D. Hsu, J. C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp 2719-2726, Albuquerque, NM, USA, 1997.

[19] T. Simeon, J. P. Laumond, and C. Nissoux, "Visibility based probabilistic roadmaps," *Advanced Robotics Journal*, vol. 14, no. 6, 2000.

[20] R. Murrieta-Cid, B. Tovar, and S. Hutchinson, "A Sampling Based Motion Planning Approach to Maintain Visibility of Unpredictable Moving Targets," *Autonomous Robots*, vol. 19, no. 5, 285-300, December, 2005.

[21] S. Leonard, E. A. Croft, and J. J. Little, "Dynamic visibility checking for vision-based motion planning," in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2283-2288, Pasadena, CA, USA, 2008.

[22] A. Chan, E. A. Croft, and J. J. Little, "Trajectory specification via sparse waypoints for eye-in-hand robots requiring continuous target visibility," in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 3082-3087, Pasadena, CA, USA, 2008.

[23] M.A. Baumann, D. C. Dupuis, S. Leonard, E. A. Croft, J. J. Little, "Occlusion-Free Path Planning with a Probabilistic Roadmap," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2151-2156, Nice, France, 2008.

[24] M. Evans and T. Swartz, *Approximating Integrals via Monte Carlo and Deterministic Methods*. Oxford University Press, 2000.

[25] J. O'Rourke, "The complexity of computing minimum convex covers for polygons," in *20th Annual Allerton Conf. on Communication, Control, and Computing*, pp. 75-84, Allerton, IL, USA, 1982.

[26] J. O'Rourke, *Computational Geometry in C*. Cambridge University Press, 1994.

[27] Q. Huang, K. Tanie and S. Sugano, Stability compensation of a mobile manipulator by manipulator motion: feasibility and planning, *Advanced Robotics*, vol.13, no. 6-8, pp.25-40, 1999.

[28] J.-C. Latombe. Robot Motion Planning. Kluwer Academic Publishers, 1991.

[29] K. Yamazaki, M. Tomono and T. Tsubouchi, Pose Planning for a Mobile Manipulator Based on Joint Motions for Posture Adjustment to End-Effector Error, *Advanced Robotics*, vol.22, no.4, pp.411-431, 2008.

[30] T. Shemer, Recent Results in Art Galleries, Proc. IEEE, 80(9), pages 1384-1399, September 1992.

[31] H.H. González and J.-C. Latombe, A Randomized Art-Gallery Algorithm for Sensor Placement. Proc. 17th ACM Symp. on Computational Geometry (SoCG'01), pp. 232-240, Medford, MA, USA, 2001

[32] G. Sanchez and J.C. Latombe, A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking. Int. Symposium on Robotics Research (ISRR'01), Lorne, Victoria, Australia, November 2001. Published in Robotics Research: The Tenth Int. Symp., R.A. Jarvis and A. Zelinsky (eds.), STAR, pp. 403-417, 2003.

[33] S. M. LaValle and J. J. Kuffner, Randomized kinodynamic planning. International Journal of Robotics Research, 20(5):378–400, May 2001.

[34] J. Espinoza and R. Murrieta-Cid, A Motion Planner for Finding an Object in 3D Environments with a Mobile Manipulator Robot Equipped with a Limited Sensor, In Proc *IBERAMIA-2010, LNCS Vol. 6433*, pages 532-541, Bahia Blanca, Argentina.

Judith Espinoza obtained a B.E in Computer Science (2004) and a M.S. in Computer Science (2006) both from Beneménerita Universidad Autónoma de Puebla. In 2007 she was awarded a fellowship from the National Council for Science and Technology of México, to pursue graduate study in Computer Science at the Centro de Investigación en Matemáticas (CIMAT), Guanajuato, México, in which she is currently a PhD student. She is mainly interested in motion planning.

Alejandro Sarmiento obtained a B.E in Electronic Systems Engineering (1995) and a M.S. in Automation - Intelligent System (1998) from Monterrey Institute of Technology and Higher Education. In 1998 he was awarded a fellowship from the National Council for Science and Technology to pursue graduate study in Computer Science at the University of Illinois. He obtained his Ph.D. in computer Science from the University of Illinois in 2004.

Rafael Murrieta-Cid received his Ph.D. from the Institut National Polytechnique (INP) of Toulouse, France (1998). His Ph.D research was done in the Robotics and Artificial Intelligence group of the LAAS-CNRS. In 1998-1999, he was a postdoctoral researcher in the Computer Science Department at Stanford University. In 2002-2004, he was working as a postdoctoral research associate in the Beckman Institute and Department of Electrical and Computer Engineering of the University of Illinois at Urbana-Champaign. From August 2004 to January 2006 he was a professor and director of the Mechatronics Research Center in Tec de Monterrey, Campus Estado de México, México. Since March 2006, he has been working in the Mathematical Computing Group at the CIMAT –Centro de Investigación en Matemáticas, in Guanajuato México. He is mainly interested in robotics and robot motion planning, and has publised more than 50 papers in Journals and International Conferences on these topics.

Seth Hutchinson received his Ph.D. from Purdue University in 1988. In 1990 he joined the faculty at the University of Illinois in Urbana-Champaign, where he is currently a Professor in the Department of Electrical and Computer Engineering, the Coordinated Science Laboratory, and the Beckman Institute for Advanced Science and Technology. Hutchinson is Editor-in-Chief of the IEEE Transactions on Robotics, and previously served as the first Editor-in-Chief for the RAS Conference Editorial Board. He currently serves on the editorial boards of the International Journal of Robotics Research and the Journal of Intelligent Service Robotics. In 1996 he was a guest editor for a special section of the Transactions devoted to the topic of visual servo control, and in 1994 he was co-chair of an IEEE Workshop on Visual Servoing. In 1996 and 1998 he co-authored papers that were

finalists for the King-Sun Fu Memorial Best Transactions Paper Award. He was co-chair of IEEE Robotics and Automation Society Technical Committee on Computer and Robot Vision from 1992 to 1996, and has served on the program committees for more than fifty conferences related to robotics and computer vision. He has published more than 150 papers on the topics of robotics and computer vision, and is coauthor of the books "Principles of Robot Motion: Theory, Algorithms, and Implementations," published by MIT Press, and "Robot Modeling and Control," published by Wiley. Hutchinson is a Fellow of the IEEE.