2011 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 25-30, 2011. San Francisco, CA, USA

# Repairing Plans for Object Finding in 3-D Environments

Judith Espinoza and Rafael Murrieta-Cid
Centro de Investigación en Matemáticas, CIMAT
Guanajuato, México
{jespinoza,murrieta}@cimat.mx

*Abstract*— **In this paper, we address the problem of repairing previously computed plans for searching for an object. The object is sought with a 7 degrees of freedom mobile manipulator robot with an "eye-in-hand" sensor. The sensor is limited in both range and field of view. Our method computes a set of sensing configurations, which collectively cover the environment with the 3-D visibility region of the limited sensor. An order for visiting sensing configurations, which diminishes the expected value of the time for finding the object is generated. The search plan corresponds mainly to the set of sensing configurations to be visited and the order for visiting those configurations. In this paper, we show that whenever the environment changes locally our plan can also be repaired locally. We base our approach on a 3-D convex regions decomposition dividing the environment. The plan is repaired by generating a new sub-set of sensing configurations and a new order for visiting those configurations, only considering the convex regions related to the change in the map of the 3-D environment. We have implemented all our algorithms, and we present simulation results in realistic environments.**

## I. Introduction

Our search problem is related to robot motion planning, art gallery problems and coverage.

In robot motion planning [9], [10] the typical goal is to find a collision free path to move a robot (a mechanical system, which may have many degrees of freedom) from an initial to a final configuration. Efficient algorithms have been proposed to solve this problem. These algorithms use sampling to capture connectivity of high dimensional configuration or state spaces, for example, [8], [6], [11], [18], just to name some classical works. 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 3-D workspace for searching for an object* and not only in representing the configuration space for avoiding robot collision. Nevertheless, we need to find collision-free paths to the move the robot between sensing configurations. Our main interest is to address the problem of finding a static object. This adds a new aspect to our planning problem.

The traditional art gallery problem is to find a minimal placement of guards such that their respective visibility regions completely cover a polygon [12], [17], [7]. As we will see below in Section II, a set of sensing configurations that collectively see the environment could be used as part of a solution to our search problem. Notice that differently to the works presented in [12], [17] and [7] we consider a 3-D environment and not a polygon.

In coverage problems (e.g., [5], [1]), 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.

We have already addressed the problem of finding an object in a 3-D environment. In [15], we investigated the problem of finding an object in a 3-D environment *for the case of a point robot equipped with an omni-directional sensor*. In [15], we have also introduced a probabilistic sampling method to decompose the workspace into convex regions. In [2], [3] we have extended our work to a mobile manipulator robot equipped with a sensor *limited in both field of view and range*; a practical method to approximate the visibility region in 3-D of the limited sensor is proposed, convex regions are used to facilitate this approximation. In [2], we have also proposed the strategy of selecting the most important degrees of freedom (DOFs) to be optimized for minimizing the expected value of the time to find the object. This strategy significantly reduces the computational running time to generate a plan. The computational running time of our algorithm refers to the time taken by our software to generate a plan 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 one to the performance in average of this plan when the plan is executed.

The algorithm presented in [2], [3] receives as input a 3-D map of the environment and it returns as output a search plan. Once that a plan is generated, it can be used several times for finding an object as long as the environment does not change. The fact that the cost of planning will be amortized over many instances of a problem provides a justification for spending time in generating the plan. In our setting each instance of a problem corresponds to a different unknown location of the object that is sought. However, when the environment changes, the plan should be modified.

The motivation that drives the technique that we present in this paper is that if the environment only changes locally, there is not need to change the whole search plan. For example, imagine the following scenario: a plan has been generated for finding some object inside a house, but after the generation of the plan, some furniture inside the house has changed location, however the house building has not

978-1-61284-455-8/11/$26.00 ©2011 IEEE

changed. In this kind of situations, the method proposed in the paper is applicable and useful. In this paper, we propose a method to repair a previously computed plan, for dealing with local changes in the 3-D environment. We base our approach on a 3-D convex regions decomposition, in which the environment is divided. The plan is repaired by generating a new sub-set of sensing configurations and a new order for visiting those configurations, only considering the convex regions related to the change in the map of the 3-D environment.

The important advantage of repairing a plan instead of generating again the whole plan is that the time needed to repair the plan is in general significantly smaller than the time needed to generate again the whole plan. This time actually depends on the percentage of the environment that has changed.

## II. The Original Plan Generation

In our formulation, we assume that the environment is known, but that we do not have information about the location of the static object being searched. This is equivalent to defining an uniform probability density function (pdf) modeling the object location. We believe that this reasoning is general given that we do not need to assume a relation between a particular type (class) 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.

The robot senses the environment at discrete configurations $q_i$ (also known as *guards*, from the art gallery problem [12]). Let's call $V(q_i)$ the visibility region associated to the limited sensor. Our searching strategy is as follows: first the whole environment is divided into a set of convex regions. To split the environment into convex regions we use the probabilistic convex cover proposed in [15]. That method divides the environment into a set called $\{C_r\}$, so that the union of all $C_r$ covers the whole environment, that is $\bigcup_r C_r = int(W)$. The interior of the workspace $int(W)$ is the free space inside the 3-D environment, $C_r$ denotes a convex region in a 3-D environment, and $r$ 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 $C_r$. Second, 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 also 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. These sensing configurations are linked in a graph and perform a graph search to establish the order to visit the configurations associated to a single convex region.

In [14] it has been shown that the problem of determining the global order for visiting sensing locations, which minimizes the expected value of the time to find an object is NP-hard, even in a 2-D polygonal workspace with a point robot. Hence, in [14], [16], we have proposed an efficient algorithm, which aims just to diminish the expected value of the time. In this paper, we use that algorithm to establish the orders for visiting convex regions and for visiting sensing configurations inside a single convex region.

Below, we briefly describe the main concepts that found the algorithm proposed in [14], [16].

The route followed by the robot corresponds to an order of visiting sensing configurations $q_{i,k}$ that starts with the robot's initial configuration and includes every other configuration. 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. 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, which is the elapsed time, and 2) the probability mass of seeing the object, which is equivalent to the *gain*.

The expected value of the time that a route takes to find the object is defined as follows:

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

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, $P\left(T = t_j\right)$ 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 Eq. 2.

We use the utility function defined below to measure how convenient it is to visit a determined configuration from another:

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

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. $P\left(q_j\right)$ is equal to $P\left(T = t_j\right)$ defined in Eq. 2.

The utility function in Eq. 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. So, we use this utility function to drive a partially greedy algorithm. Our algorithm is able to explore several steps ahead without incurring a too high computational cost. In the worst case, our algorithm has a running time complexity of $O\left(n^3 \log n\right)$. A description of this algorithm can be found in [16], together with a comparison between the performance of the algorithm (in terms of the expected value of the time to find the object) vs. the optimal path. We stress that our algorithm often reduces in 3 orders of magnitude the computational running time compared with the algorithm needed to find the optimal solution, which is exponential since the optimization problem to be solved is NP-hard.

### *A. Paths to move between convex regions*

Since the expected value of the time depends on the cost (time) of moving the robot between sensing configurations, we need to find shortest paths to move the robot between convex regions.

Our convex cover gives flexibility about the first sensing configuration to be visited associated to a given convex region $C_r$. Any configuration $q_i$ which places the robot's sensor inside $C_r$ is a valid configuration. The actual paths depend on the metric used to measure cost to move between convex regions. One way to define the cost between two configurations $X$ and $Y$ in a $D$-dimensional configuration space is

$$\|X - Y\|_\Lambda \equiv (X - Y)^T \Lambda (X - Y), \tag{4}$$

where $\Lambda$ is a diagonal matrix with positive weights $\lambda_1, \lambda_2, \ldots \lambda_D$ assigned to the different DOFs.

Our planner coordinates the translation of the robot base and the rotations of base and the arm's links, such that both translation and rotations happen at once, then the cost of moving the arm is zero (in terms of elapsed time), since the motions (translation and rotations) are simultaneous.

Consequently, we consider that the coordinates $(x, y)$ defining the position are the DOFs determining the cost of moving the whole system. Hence, we optimize only these two DOFs. To find the shortest path between one given convex region and all the others, we use the wavefront expansion (called NF1) proposed in [9]. We determine the other DOFs using a randomized sampling procedure. Thus, a robot path to move between convex regions is a sequence of robot's configurations, in which some DOFs are planned optimally and the others do not produce collisions between the robot and the obstacles (see [2]).

### *B. Selecting and connecting sensing configurations inside a single convex region*

The method that we propose [2], [3] to cover each convex region with a limited sensor is based on sampling. A video showing some simulation results reported in [2] can be found at:
http://www.cimat.mx/%7Emurrieta/Papersonline/VideoIber.wmv

Sensing configurations $q_{(i,r)}$ are generated with a uniform probability distribution in a configuration space $\mathcal{C}$ of 7 dimensions: 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 $s_r \in S_r$. Each point sample $s_r$ lies in the 3-D space, and is defined by a 3-dimensional vector $p(x, y, z)$. We use $S_r$ to cover the convex region $C_r$ with a limited sensor.

Our algorithm for selecting sensing configurations has been inspired from the algorithm presented in [7], that method was designed to cover the boundary $\partial P$ of a polygon $P$, we have extended the method to cover the interior of the polyhedral representation of a 3-D environment $int(W)$.

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's 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 3-D visibility computations. The visibility region of the limited sensor at configuration $q_{(i,r)}$ is approximated by:

$$V(q_{(i,r)}) = \bigcup_s s_r \in int(\mathcal{F} \cap C_r) \tag{5}$$

Where $s$ indicates sample points.

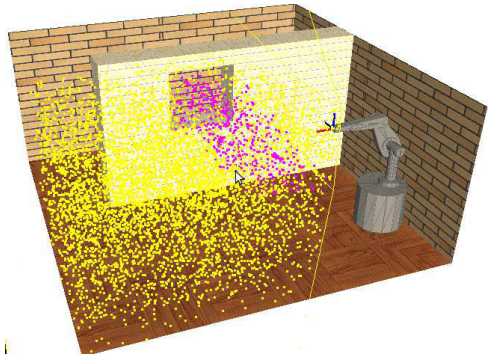


Fig. 1. Sets $\{s_r\}$ and $\{s_v)\}$

While covering region $C_r$, we also mark as sensed and logically remove, all samples $s_v$ belonging to region $C_v, v \neq r$, if $s_v \in int(\mathcal{F} \cap C_r \cap C_v)$. It is guaranteed, that these samples are not occluded from configuration $q_{(i,r)}$. In figure 1 dark (magenta) dots are used to show the set $S_v$, and light gray (yellow) 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:

$$\bigcup_s s_r \in int(C_r) = S_r \tag{6}$$

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. 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.

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. For reducing the computational time to cover the environment with a limited sensor, we *estimate* the cost to move between sensing configuration as a straight line in the configuration space $\mathcal{C}$.

In the motion planning problem of generating collision free paths to move between configurations, we use a lazy collision checking scheme [13]. 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. Evidently, sometimes the fully connected graph splits into two connected components, if so, we use an RRT [11] to find a collision-free path between the two components. We stress that we have found in our experiments that only $\frac{1}{10}$ of the total number of paths to sense convex regions are computed with an RRT. All other times, a straight line in $\mathcal{C}$ was enough to find collision free paths.

To cover a single convex region the robot travels a tour, the first sensing configuration and the last one is the same, this allows to preserve the path and its cost of moving between convex regions, and consequently the order to visit them, which has been previously planned.

## III. Repairing the plan

### A. Dividing the environment into convex regions

In [15] we have proposed an algorithm for a convex cover. That algorithm is based on sampling and divides the environment into overlapping convex regions. Roughly, the algorithm works as follows: first, 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. 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 $g \in G$. Second, guard kernels are divided into convex region by using convex hulls. The resulting convex regions are expanded by adding sample points as long as doing so does not generate a collision with the obstacles. 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 (A detailed description of this algorithm can be found in [15]).

### B. Modifying the convex cover to deal with changes in the environment

The changes in the environment are detected using the original convex cover. Indeed, the change of location of an obstacle in the environment will produce the following modifications over the convex regions originally generated. 1) The regions related to the original location of the obstacle must be modified 2) the regions related to the new position of the obstacle must also be modified. Let us call the first set of regions $\{C\}_t$ and the second set $\{C\}_{t+1}$.

To define which regions are members of $\{C\}_t$, it is necessary to detect the regions which are adjacent to the original obstacle position. A way (among others) of detecting these regions is by measuring the distance between the convex regions and the obstacle. All regions which are closer than a given small $\epsilon$ to the obstacle are members of $\{C\}_t$. The set of regions $\{C\}_t$ has associated a set of point samples called $S_t$. The point samples in the interior of the union of all the regions in $\{C\}_t$ forms the set $S_t$, that is $S_t = \bigcup_s s \in int(\{C\}_t)$.

Defining which regions are members of $\{C\}_{t+1}$ is simple. Merely all original convex regions are tested for collision with the obstacle at its new location, those regions in collision belong to $\{C\}_{t+1}$. The set of regions $\{C\}_{t+1}$ has also associated a set of point samples called $S_{t+1}$, which is defined by $S_{t+1} = \bigcup_s s \in int(\{C\}_{t+1})$ We also need to determine the point samples in collision with the original and new obstacle locations. Let's call the set of point samples in collision with the obstacle at its original location $Obs_t$, and the set of samples in collision with the obstacle at its new location $Obs_{t+1}$.

The key idea to compute the new convex regions needed to take into account of the change of the map is the following: only a subset of the samples used to compute the original convex cover are given as input to the algorithm that generates the local convex cover decomposition considering the change in the map. Let's call this subset $S_\Delta$. The set $S_\Delta$ is the union of the point samples of the sets $S_t$, $S_{t+1}$ and $Obs_t$ minus the samples of the set $Obs_{t+1}$, that is:

$$S_\Delta = \bigcup_s s \in \left(S_t \bigcup S_{t+1} \bigcup Obs_t\right) \setminus Obs_{t+1} \qquad (7)$$

The polyhedral representation of the environment considering the new obstacle location, and the set of point samples $S_\Delta$ are given as inputs to the convex cover algorithm proposed in [15]. The algorithm returns as outputs the new set of convex regions needed to take into account the change in the environment. We call this set $\{C\}_\Delta$.

Let's call $\{F\}$ to the set of all the original convex regions, and $\{N\}$ to the set of all convex regions after having modified the environment. This new set of convex regions is composed by the original regions that have not

been eliminated (i.e. $\{F\} \setminus (\{C\}_t \bigcup \{C\}_{t+1}$ ) plus the set $\{C\}_\Delta$, that is: $\{N\} = (\{F\} \setminus (\{C\}_t \bigcup \{C\}_{t+1})) \bigcup \{C\}_\Delta$. $\{N\}$ totally covers the modified environment.

### C. Modifying the orders to visit convex regions and sensing configurations

As it was mentioned above, the problem of generating an order to visit sensing configurations was planned in two steps: first an order to visit convex regions is computed (we call it the global plan). Second, for every convex region an order to visit sensing configurations is established. Originally, this heuristic was proposed to reduce the computational running time to generate a search route. We will show now that this heuristic of dividing the large problem into several smaller sub-problems also facilities to repair a previously computed plan.

In order to repair the previously computed search path, several options are possible. For instance, one is just to recompute the order for visiting convex regions and try to preserve as much as possibly the local orders for visiting sensing configurations inside each convex region. However, under the assumption that the change in the environment is local, it makes sense to modify the global plan only locally, preserving as much as possible the order to visit convex regions. This approach has the advantage that the computational running time to repair the plan is typically smaller than the one needed to recompute the whole global plan, adding reactivity to the re-planning process.

For repairing the plan, the spatial location of the new convex regions is taken into account. Furthermore, the new convex regions will appear at spatial locations related to the site occupied by the regions belonging to $\{C\}_t$ and $\{C\}_{t+1}$.

The goal of algorithm 1 is to compute a new order for visiting regions, preserving as much as possible the original global plan. Let's call $\{O\}_F$ to the ordered set of original convex region, and $\{O\}_N$ to the new ordered set of convex regions.

First, the set $\{F\}$ is ordered according to which region is visited earlier (by the robot) in the original global plan (line 1 in algorithm 1). The approach to order $\{F\}$ is briefly described in section II, more details can be found in [14], [2].

Regions $C_{r,j}$ index the elements of this set; $r$ refers to the region's label (region's identifier), and $j$ refers to the $j^{th}$ visited region. Regions $C_v \in \{C\}_\Delta$ are the new generated regions, $v$ refers to the region's label.

Regions $C_{r,k}$ index the elements of $\{O\}_N$ ($k$ refers to the $k^{th}$ visited region in the new set $\{O\}_N$).

Second, all regions $C_{r,j} \in (\{C\}_t \bigcup \{C\}_{t+1})$ are eliminated in the new plan. Every region $C_{r,j} \in (\{C\}_t \bigcup \{C\}_{t+1})$ is checked for collision with every region $C_v \in \{C\}_\Delta$. Notice that more than one region $C_v$ might intersect the same region $C_{r,j}$. All regions in $\{C\}_\Delta$, which intersect the same $C_{r,j}$ are stored in the set $\{C\}_{aux}$ (line 7 in algorithm 1). Let's call $a_j$ the cardinality of the set $\{C\}_{aux}$, for each region $C_{r,j}$.

**Algorithm 1** Computing the new order to visit convex regions

**Input**: Sets: $\{F\}$, $(\{C\}_t \bigcup \{C\}_{t+1})$, $\{C\}_\Delta$.
**Output**:$\{O\}_N$ new ordered set of convex regions.
1. $\{O\}_F \leftarrow$ Order$\{F\}$;
2. $k = 1$, $e = a = 0$;
**for** $j = 1$ to $|\{O\}_F|$ **do**
  3. $C_{r,j} \leftarrow \{O\}_F$;
  **if** $C_{r,j} \in (\{C\}_t \bigcup \{C\}_{t+1})$ **then**
    4. $\{C\}_{aux} \leftarrow \emptyset$;
    5. $e = e + 1$;
    **for** $n = 1$ to $|\{C\}_\Delta|$ **do**
      6. $C_v \leftarrow \{C\}_\Delta$;
      **if** $(C_{r,j} \bigcap C_v \neq \emptyset)$ **then**
        7. $\{C\}_{aux} \leftarrow C_v$;
        8. $a = a + 1$;
      **end if**
    **end for**
    9.$\{C\}_\Delta \leftarrow \{C\}_\Delta \setminus \{C\}_{aux}$;
    10. Local-Order($\{C\}_{aux}$,$q_{robot} \in C_{r,k}$);
    11. $\{O\}_N \leftarrow \{C\}_{aux}$;
  **else if** $C_{r,j} \notin (\{C\}_t \bigcup \{C\}_{t+1})$ **then**
    **if** $e = 0$ **then**
      12. $\{O\}_N \leftarrow C_{r,j}$;
      13. $k = j$;
    **else if** $e \neq 0$ **then**
      14. $k = j + a - e$;
      15. $C_{r,k} = C_{r,j}$;
      16. $\{O\}_N \leftarrow C_{r,k}$;
    **end if**
  **end if**
**end for**

In algorithm 1 (line 10), the method Local-Order is used to establish the order for visiting regions in $\{C\}_{aux}$; $q_{robot} \in C_{r,k}$ denotes the first visited robot configuration in convex region $C_{r,k}$. Local-Order does the following. Assuming that the robot is located at $q_{robot} \in C_{r,k}$, 1-step ahead evaluation of the utility function in Eq. 3 is used to establish the region $C_{v,k+1}$. The region which maximizes Eq. 3 is chosen to be the $k+1$ region to be visited in the new order. Assuming now, that the robot is located at $q_{robot} \in C_{v,k+1}$ and again using 1-step ahead utility function evaluation, the remaining $(a_j - 1)$ regions in $\{C\}_{aux}$ are evaluated to determine the visited region $C_{v,k+2} \in \{O\}_N$, and so forth until all $a_j$ regions are ordered. For establishing an order for visiting regions in $\{C\}_{aux}$, the case in which the robot is already located at region $C_{v,k} \in \{C\}_{aux}$ must be considered. This mean that the cost to travel to this region is zero, and therefore there is not need to evaluate the Eq. 3.

Third, the regions $C_{r,j} \notin (\{C\}_t \bigcup \{C\}_{t+1})$ are included in the new order with the following simple rules: If no region $C_{r,j}$ has been eliminated then $k = j$, and region $C_{r,k}$ has the same place in the order as region $C_{r,j}$ (line 12 in algorithm 1). If at least one region $C_{r,j}$ has been eliminated from the

original plan then $k = j + a - e$, in which $a$ is the number of new regions added to the new plan until that $j$, and $e$ is the number of regions which have been eliminated also until that $j$ (lines 14, 15 and 16 in algorithm 1).

Note that each region $C_v \in \{C\}_\Delta$ is included only once in the new plan. The location of the first region $C_{r,j}$ which intersect region $C_v$ is taken into account to establish the order of region $C_v$ in the new plan. Once a region $C_v$ is included in the new plan, it is eliminated from $\{C\}_\Delta$ (line 9 in algorithm 1). This procedure avoids redundancy in the new plan. Since regions $C_v \in \{C\}_\Delta$ replace regions $C_{r,j} \in (\{C\}_t \bigcup \{C\}_{t+1})$, there is not need of including a region $C_v$ more than once. The new order is complete when all original $C_{r,j}$ regions have been considered.

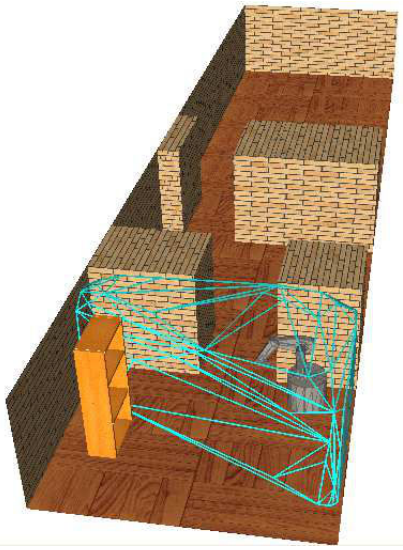
The collision free paths to move the robot between new convex regions in $\{C\}_\Delta$, and between a region in $\{F\}$ and a region in $\{C\}_\Delta$ is a sequence of robots configurations, in which some DOFs are planned optimally (using the wavefront expansion proposed in [9]) and the others do not produce collisions between the robot and the obstacles, as we have described in Section II-A. Finally, new local plans are computed for visiting sensing configurations associated to every new region in $\{C\}_\Delta$. These local plans are computed with the approach described in Section II-B.

Since in our original global plan for covering the environment with the limited sensor, while covering region $C_r$, all samples $s_p$ belonging to region $C_p$ ($p \neq r$) are marked and logically removed (whenever $s_p \in int(\mathcal{F} \cap C_r \cap C_p)$), then it is necessary to recompute new local plans for regions $C_{r,j} \notin (\{C\}_t \bigcup \{C\}_{t+1})$, which intersected regions that have been eliminated and were before in the original order of visiting convex regions. Those new local plans are also computed with the approach described in Section II-B.

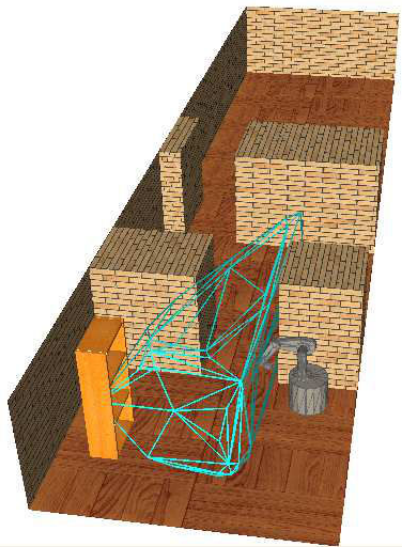
## IV. Simulation Results

All the results presented in this paper were obtained with a regular PC running Linux OS, the processing speed of the CPU is 2.2 GHz. The programming language used to obtain our simulation results was C++. In the figures a (cyan) mesh is used to show the convex regions. Initially, the environment was divided in 11 convex regions, 44 sensing configurations were needed to cover the environment with the limited sensor. The original order to visit convex regions was the following: $C_{id9,1} \rightarrow C_{id7,2} \rightarrow C_{id6,3} \rightarrow C_{id11,4} \rightarrow C_{id1,5} \rightarrow C_{id2,6} \rightarrow C_{id3,7} \rightarrow C_{id8,8} \rightarrow C_{id5,9} \rightarrow C_{id10,10} \rightarrow C_{id4,11}$. The total computational running time to generate the original plan (for covering the whole environment with the limited sensor) was 1 minute and 58 seconds. The expected value of the time for finding the object associated to this plan was 796.02 units.

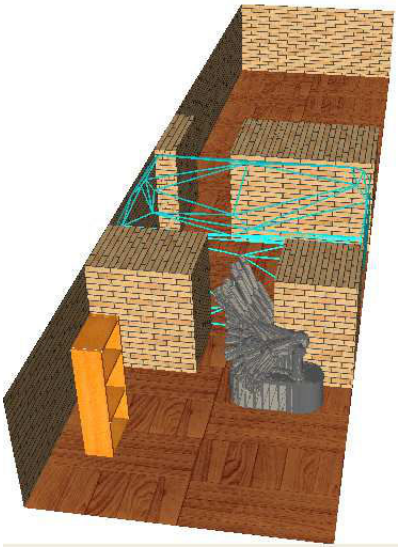
Figure 2 (a) shows the initial robot configuration and the convex region $C_{id9,1}$. Figures 2 (b) shows the robot having the sensor inside region $C_{id7,2}$. Figure 2 (c) shows the path to move between regions $C_{id1,5}$ and $C_{id2,6}$. Figure 3 (a) shows the global path to visit convex regions. Figure 3 (b) shows with light gray (yellow) the point samples used to approximate the visibility region of the limited sensor.



(a) Region $C_{id9,1}$



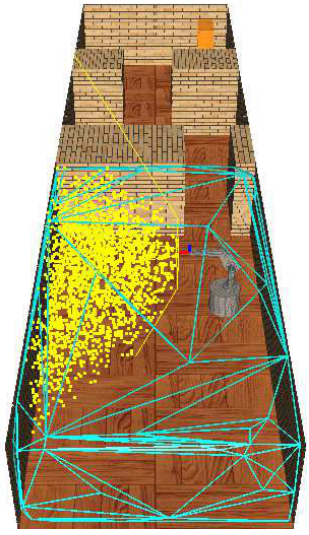
(b Region $C_{id7,2}$



(c) Path $C_{id1,5} \rightarrow C_{id2,6}$

Fig. 2. Finding an object

(a) Global path

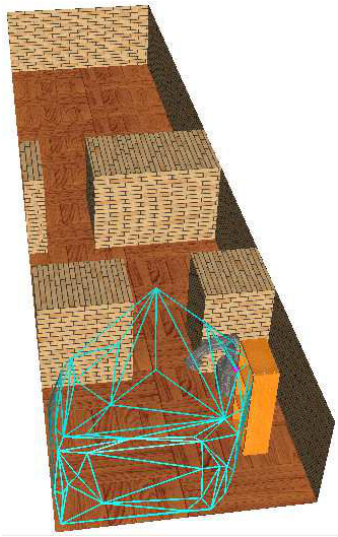


(b) Covering region $C_{id2,6}$ with the limited sensor

Fig. 3. (a) Global path and (b) covering region $C_{id2,6}$

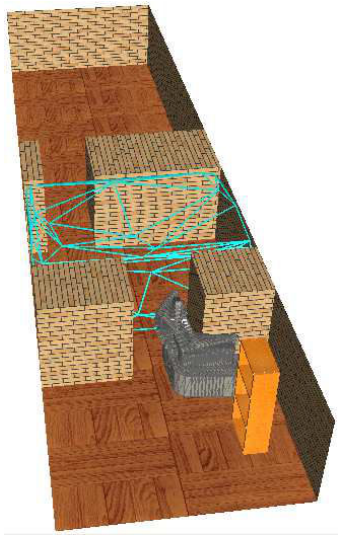
The original plan was modified as follows: $(\{C\}_t \bigcup \{C\}_{t+1})$=$\{C_{id4,11}, C_{id5,9}, C_{id6,3}, C_{id7,2}, C_{id8,8}, C_{id9,1}, C_{id10,10}, C_{id11,4}\}$.
$C_\Delta = \{C_{id12}, C_{id13}, C_{id14}, C_{id15}, C_{id16}, C_{id17}, C_{id18}, C_{id19}\}$. Thus, 8 regions were removed of the original plan and 8 new regions were generated. The new order for visiting convex regions is: $C_{id17,1} \rightarrow C_{id18,2} \rightarrow C_{id16,3} \rightarrow C_{id14,4} \rightarrow C_{id13,5} \rightarrow C_{id19,6} \rightarrow C_{id15,,7} \rightarrow C_{id12,8} \rightarrow C_{id1,9} \rightarrow C_{id2,10} \rightarrow C_{id3,11}$. 43 sensing configurations were needed to cover the modified environment. The computational running time to modify the original plan was 21.5 seconds. The expected value of the time for finding the object associated to the new plan is 771.20 units.

Figure 4 (a) shows the new obstacle (a bookshelf) location and the robot having the sensor inside region $C_{id18,2}$. Figure 4 (b) shows the path between regions $C_{id18,2}$ and region $C_{id16,3}$. Figure 4 (c) shows the path between region $C_{id12,8}$ and region $C_{id1,9}$. Figure 5 (a) shows the 4 sensing



(a) New obstacle location and region $C_{id18,2}$

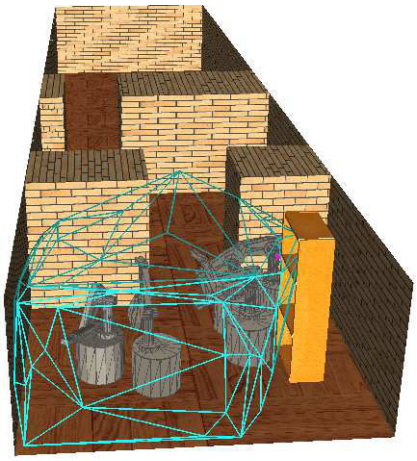
(b) Path $C_{id18,2} \rightarrow C_{id16,3}$.



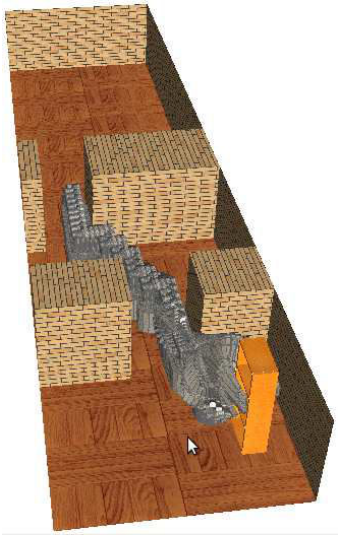
(c) Path $C_{id12,8} \rightarrow C_{id1,9}$

Fig. 4. New order

configuration that collectively covers with the limited sensor the convex region $C_{id18,2}$. Figure 5 (b) shows the new global path to visit all convex regions.



(a) Sensing configurations, region $C_{id18,2}$



(b) New global path

Fig. 5. (a) Sensing configurations, $C_{id18,2}$, and (b) new global path

The sub-order, sensing configurations and paths to cover the room in which the obstacle has changed location were modified. However, the sub-order, the paths and sensing configurations to cover the other two rooms were preserved. In the experiment included in this paper, the expected value of the time was improved with the new plan. We have found that in general, the expected value of the time for finding the object is almost the same after having update the plan.

## V. Conclusion and Future Work

In this paper, we have proposed an approach for repairing previously computed plans for searching for an object in 3-D environments. The object is sought with a 7 degrees of freedom mobile manipulator robot with an "eye-in-hand" sensor. The sensor is limited in both range and field of view. We have shown that whenever the environment changes locally our plan can also be repaired locally. We base our approach on a 3-D convex regions decomposition dividing the environment. The plan is repaired by generating a new sub-set of sensing configurations and a new order for visiting those configurations, only considering the convex regions related to the change in the map of the 3-D environment. The important advantage of repairing a plan instead of generating again the whole plan is that the computational running time needed to repair the plan is in general significantly smaller than the time needed to generate again the whole plan, while the expected value of the time for finding the object almost remains the same. We have implemented all our algorithms, and we present simulation results in realistic environments. As a future work we want to test our approach in a real robot together with a computer vision algorithm to detect the object.

## References


[1] 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*, IEEE/RSJ-IROS 2001.

[2] 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.

[3] J. Espinoza, A. Sarmiento, R. Murrieta-Cid and Seth Hutchinson, A Motion Planning Strategy for Finding an Object with a Mobile Manipulator in 3-D Environments, *To appear in Journal Advanced Robotics,* 2011.

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

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

[6] D. Hsu, J. C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proc. IEEE Int. Conf. on Robotics and Automation*, IEEE-ICRA 1997.

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

[8] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, Jun 1996.

[9] J.-C. Latombe. Robot Motion Planning. Kluwer A.P., 1991.

[10] S. LaValle, *Planning Algorithms*, Cambridge University press, 2006.

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

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

[13] G. Sanchez and J.C. Latombe, A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking. *ISRR'01, R.A. Jarvis and A. Zelinsky eds. STAR*, 2003.

[14] A. Sarmiento, R. Murrieta-Cid, 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, USA, 2003.

[15] A. Sarmiento, R. Murrieta-Cid and S. Hutchinson, A Sample-based Convex Cover for Rapidly Finding an Object in a 3-D environment, *Proc. IEEE Int. Conf. on Robotics and Automation*, IEEE-ICRA 2005.

[16] A. Sarmiento, R. Murrieta-Cid, 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.

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

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