A Motion Planning Strategy for Rapidly Finding an Object with a Mobile Manipulator in 3-D Environments

Alejandro Sarmiento[1], Judith Espinoza[2], Rafael Murrieta-Cid[2], and Seth Hutchinson[1]

[1] University of Illinois, Beckman Institute
Urbana, Illinois
{asarmien,seth}@uiuc.edu
[2] Centro de Investigación en Matemáticas, CIMAT
Guanajuato, México
{jespinoza,murrieta}@cimat.mx

Abstract. In this paper, we address the problem of searching for an object in a 3-D environment. We consider a mobile manipulator with an "eye-in-hand" sensor moving in the 3-D environment. In particular, we consider a static object whose location is modeled with a probability density function (pdf). We generate routes that minimize the expected value of the time until the object is first seen when following the route. We use 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 compromise between moving the manipulator base and moving the robotic arm.

1 Introduction

This work addresses the problem of finding a static object. In this paper, our goal is for the robot to find the object as quickly as possible on average. We claim that our work presents a new paradigm for search tasks, where it is important to gain as much new information in the shortest time as possible. This can be very useful in applications where 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 combinatoric one.

The possible applications have a wide range, from finding a specific piece of art in a museum to search and detection of injured people inside a building. This problem is closely related to the coverage problem [4] in the sense that any complete strategy to find an object must sense the whole environment.

In [1] we have investigated the problem of finding an object in a 3D environment *for the case of a point robot*. In that work, we have introduced a probabilistic sampling method to decompose the workspace into convex regions. In this paper we use that convex region partition. The research reported in this paper differs from our previous efforts in the following main points:

A. Gelbukh and E.F. Morales (Eds.): MICAI 2008, LNAI 5317, pp. 562–572, 2008.

1. We consider a mobile manipulator with an "eye-in-hand" sensor.
2. We investigate the use of different metrics to quantify the paths cost.
3. We use complete regions as sensing locations as opposed to a single location (point).

Note that current technology make feasible the assumption of equipping a mobile manipulator with an "eye-in-hand" sensor. For instance, using the very small omnidirectional camera described in [3]. Below, we present the definition of our problem.

2 Problem Definition

This work addresses the problem of minimizing the expected value of the time to find an object in a known 3-D workspace.

In general terms, we define the problem of searching for an object as follows: Given one mobile manipulator robot with sensing capabilities, a completely known 3D environment and an object somewhere in the world, develop a motion strategy for the robots to find the object in the least amount of time on average.

The environment W is known, and modeled as a set of polyhedrons. The obstacles generate *both motion and visibility constraints*. Furthermore, we assume that the probability of the object being in any specific point is uniformly distributed. Therefore, the probability of the object being in any subset $C_i \subset W$ is proportional to the volume of C_i. Where C_i is a convex region in a 3-D workspace.

The robot senses the environment at a set of locations L_i (also known as *guards*, from the art gallery problem [11]). The visibility region of location L_i is considered to be the volume of the convex region denoted C_i. Note that all points inside C_i can be connected by a clear line of sight from any location (point) L_i inside C_i. Note also that, it gives flexibility as to where to place the sensor, any point L_i inside C_i is a valid candidate.

The set $\{C\}$ is chosen so that the union of all C_i covers the whole environment, that is, $\bigcup_i C_i = W$. We do not require nor assume the set $\{C\}$ to be minimal.

Our exploration protocol is as follows: the robot always starts at a particular region in $\{C\}$ (associated to the starting point) and visits the other regions as time progresses. It follows the shortest paths for some given metric between them. We also investigate the use of different metrics to quantify the paths cost. We generate, metric-dependent trajectories, that is, trajectories that find a compromise between moving the base or moving the robotic arm.

The robot only gathers information about the environment (sensing) when it reaches one different convex region – it does not sense while moving. We describe the route followed by the robot as a series of convex regions C_{i_k} that starts with the robot's initial region and includes every other region at least once. Note that while C_i refers to region in the environment, C_{i_k} refers to the *order* in which those regions are visited. That is, the robot always starts at C_{i_0}, and the k-th region it visits is referred to as C_{i_k}.

For any route R, we define the time to find the object T as the time it takes to go through the regions – 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

$$P\left(T = t_j\right) = \frac{Volume\left(C_{i_j} \setminus \bigcup_{k<j} C_{i_k}\right)}{Volume(W)}. \tag{2}$$

Here, t_j is the time it takes the robot to go from its initial position – through all regions along the route – until it reaches the j-th *visited* region C_{i_j}, and $P\left(T = t_j\right)$ is the probability of seeing the object for the first time from region C_{i_j}. Since the robot only senses at specific regions, we also denote this probability of seeing the object for the first time from region C_{i_j} as $P\left(C_{i_j}\right)$.

2.1 Expected Value vs. Worst Case

It is important to note the difference between minimizing the expected value of the time to find an object and minimizing the time it would take in the worst case.

To minimize the worst case time, the robot must find the shortest path (for instance, in euclidean sense) that completely covers the environment (the shortest watchman tour problem [5]). This usually means that no portions of the environment are given any priority over others and the rate at which new portions of the environment are seen is not important.

On the other hand, to minimize the expected value of the time, the robot must gain probability mass of seeing the object as quickly as possible. For a uniform probability density function (pdf), this translates into sensing large portions of the environment as soon as possible, even if this means spending more time later to complete covering the whole environment. We believe this represents another paradigm for search tasks, where it is important to gain as much new information in the shortest time possible.

The trajectories that satisfy the previous two criteria are not the same. In fact, for a given environment, the route that minimizes the distance traveled may not minimize the expected value of the time to find an object along it.

Consider the example in Fig. 1. The robot starts in the corridor at location L_0. Assume that the object will always be in one of two rooms, and the probability of it being in either is related to the size of the room. These rooms have a narrow door and the entire room is visible from the threshold. The room to the right – seen from location L_1 – is smaller but lies closer to the initial location, while the room to the left – seen from L_2 – is larger but farther from initial position. There are only two routes the robot might take to solve this problem: Go to the smaller room first ($L_{0_0} \rightarrow L_{1_1} \rightarrow L_{2_2}$), or go to the larger room first ($L_{0_0} \rightarrow L_{2_1} \rightarrow L_{1_2}$). For the following analysis, we assume that the robot moves at a constant speed of 1 unit per second.

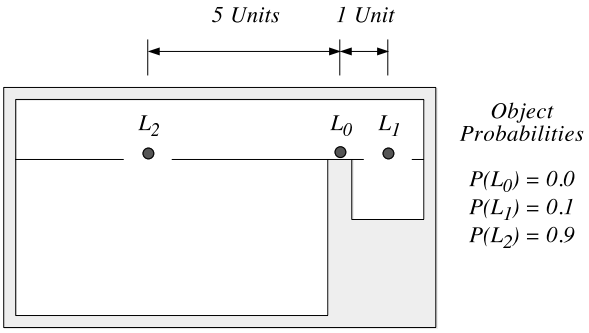


Fig. 1. Example with a simple environment

Route 1 – If the robot goes to the smaller room first and then moves on to the larger room, it reaches L_1 at time 1 and L_2 at time 7. The expected value of the time it takes to find the object following this route is

$$E\left[T\right|\left(L_0, L_1, L_2\right)] = (0.1)(1) + (0.9)(7) = 6.4.$$

The robot always completes its search after 7 seconds.

Route 2 – If the robot moves to the larger room first and then goes to the smaller room, it reaches L_2 at time 5 and L_1 at time 11. The expected time in this case is

$$E\left[T\right|\left(L_0, L_2, L_1\right)] = (0.9)(5) + (0.1)(11) = 5.6.$$

In the worst case, it will take the robot 11 seconds to find the object.

A robot following route 1 always finishes searching after 7 seconds, while a robot following route 2 takes 11 seconds. Route 1 minimizes the distance traveled. However, the average time it takes for a robot following route 1 to find the object is 6.4 seconds whereas for route 2 it is only 5.6 seconds. Route 2 minimizes the expected value of the time to find an object.

Thus, *a trajectory that is optimal in the distance traveled does not necessarily minimize the expected value of the time to find an object along it.*

3 Approach Overview

This work builds on our previous research on expected value search with mobile robots. In [2] 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 finding an object in a 2-D polygonal workspace with a point robot is NP-hard. 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 too high a computational cost. The present work builds on the same problem but in a three dimensional workspace with a mobile manipulator robot.

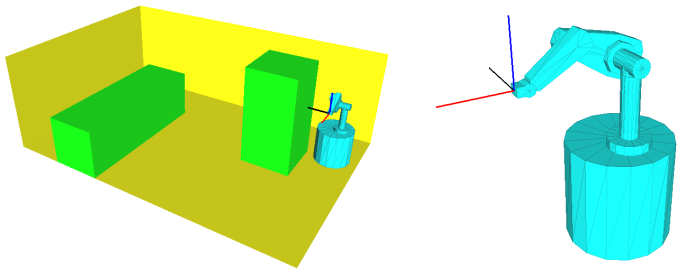
The expected value of the time depends on two factors: 1) The distance traveled (equivalent to the time if the robot moves to constant saturated speed), quantified in some given metric, which represents the *cost* and 2) the probability mass of seeing the object, which is equivalent to the *gain*.

We provide algorithms to compute the shortest paths (for a given metric) for a 7 degrees of freedom mobile manipulator to move between configurations, –cost. We also provide visibility-based decompositions of the environment to compute the probability mass of seeing the object, –gain. We use a set of sensing locations. Then we link them in a graph and perform a graph search to generate the global trajectory.

We propose a sample-based convex cover algorithm that allows us to partition the environment into convex pieces and also to estimate the volume of each piece. Using our convex cover decomposition, we propose an approach to our search problem, for the mobile manipulator with an "eye-in-hand" sensor like the one shown in Fig. 2, moving in a 3-D environment.

We use the complete regions as sensing locations, that is, as long as the end effector (which is equipped with a sensor) is inside one of them, we know that the vast majority of the whole region will be visible to the robot. This gives flexibility as to where to place the sensor, and is also helpful in the generation of what we call "metric-dependent trajectories." These trajectories take into account weights assigned to the different degrees of freedom and therefore, can be good compromises between moving the base or moving the robotic arm.

Finally, we use our previously proposed greedy approach [2] to search the resulting graph and generate trajectories that minimize the expected value of the time to find an object in the environment. Unlike our previous work, the proposed algorithms are no longer complete nor deterministic, but we do provide probabilistic bounds on how good the coverage is.



(a) (b)

Fig. 2. a) 3-D workspace b)A mobile manipulator with an "eye-in-hand" sensor

4 Sample-Based Sensing Strategy in 3-D

Our convex cover and subsequent graph abstraction capture the connectivity of the workspace. There have been many related approaches that connect samples to capture connectivity of high dimensional spaces, for example, [8,7,10,12] just to name a few. Our work is different in several aspects. First, we are interested in representing the workspace for searching an object, not in abstracting the configuration space for path planning purposes. Second, most techniques throw away "useless" samples, we keep all of them since we need to estimate the size of volumes. Finally, we can determine a clear-cut threshold of when it is no longer useful to continue sampling.

Due to the high computational complexity of visibility computation in 3-D [11], we decided to use sampling to decompose the environment into convex regions and to estimate the size of those regions. In particular, we implemented the sampling strategy in [12] originally proposed to construct a probabilistic roadmap in the configuration space. The difference in our case is that we will sample the workspace (not the configuration space) and keep all samples, as opposed to throwing away those that do not join important configurations.

In order to capture the size and shape of the workspace W, we generate a set of independent, uniformly distributed collision free samples S in 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 member 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,

$$\bigcup_{g_i \in G} Vis\,(g_i) = S, \tag{3}$$

where $Vis\,(g_i)$ is the set of points in S whose line segment to g_i does not intersect the workspace W. Also, since guards must not be mutually visible, $g_i \notin Vis\,(g_k) \quad \forall k \neq i$.

To determine how well the workspace has been covered, consider an environment of unit volume like the one shown in Fig. 3. 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.

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,

$$(1-\epsilon)^m \leq \alpha; \quad m\log(1-\epsilon) \leq \log(\alpha); \quad m \geq \frac{\log(\alpha)}{\log(1-\epsilon)}\ . \tag{4}$$

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.

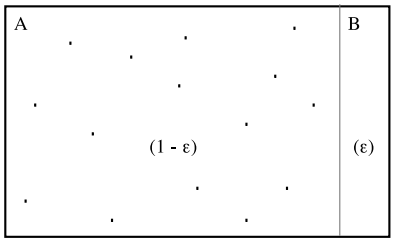


Fig. 3. An environment of unit volume and the sizes of covered (A) and uncovered (B) regions

4.1 Convex Cover Algorithm

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 denote it as

$$Ker\left(g_i\right) = Vis\left(g_i\right) \setminus \bigcup_{k \neq i} Vis\left(g_k\right). \tag{5}$$

In any minimum convex cover $\{C\}$, each convex region C_i has a set of points only contained in that particular region. Otherwise, region C_i 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. More details about our algorithm to generate a convex cover environment partition can be found in [1].

5 Application to a Mobile Manipulator

We have applied our search strategy to a mobile manipulator with an "eye-in-hand" sensor, like the one shown in Fig. 2 b). 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 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, see figure 2 (b). The figure also shows the coordinate frame for the end effector, where our sensor resides.

The 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 one hand, this simplifies the path planning problem, since we have a *set* of goal

configurations, as opposed to a single one. But on the other, 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 want to find the path, which minimizes the expected value of the time to find an object. But, recall that the computation of the expected value of the time depends on two main factors 1) the distance traveled (cost) and 2) the probability mass of seeing the object (gain). Thus, we also need to find shortest paths for a given metric (cost) between configurations for a 7 degrees of freedom mobile manipulator. The choice of metric is interesting because it can generate different "behaviors" for the robot, as we will explain in the next subsection.

5.1 Metric-Dependent Trajectories

We would like 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

$$\|X - Y\|_{\Lambda} \equiv (X - Y)^T \Lambda (X - Y), \tag{6}$$

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 not all joints might be 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, the two DOFs that translate the robot on the plane and the five rotations that further place the end effector at a specific position.

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 the low 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 (the convex region). To find that shortest path, we implemented a wavefront expansion [9]. This algorithm restricted to a *particular* robot has polynomial complexity .

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.

5.2 Simulation Results

To make the results more evident, we divided the DOFs of the robot into two groups, two translations and five rotations. First we assigned a larger weight

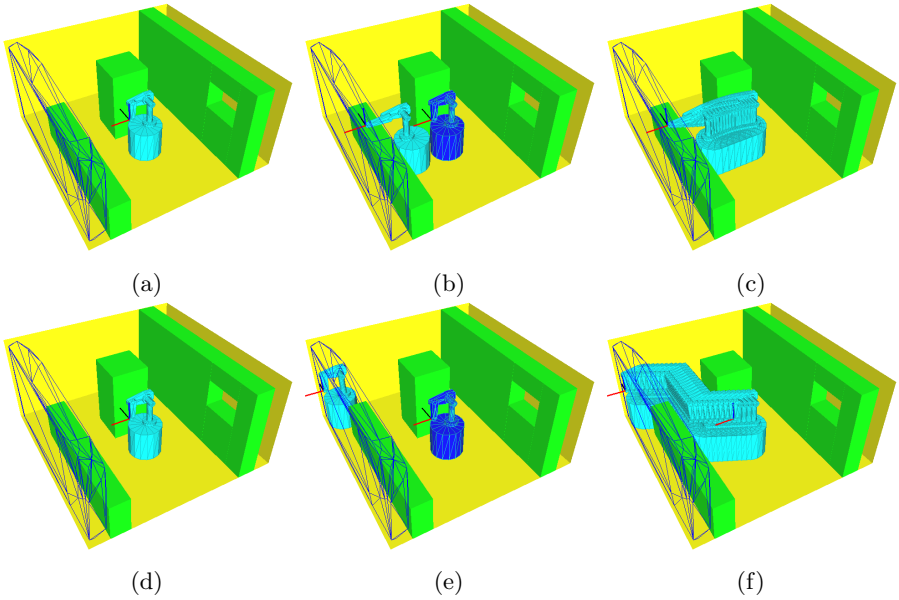


Fig. 4. Visiting one region: Low-Cost Translation vs Low-Cost Rotation

to one group and then to the other. Figure 4 Parts (a) and (d) show the initial position and one region that need 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 shown in figure 4, the robot only visits one region, but we assign different weights to the rotation and translation DOFs. Parts (a) to (c) show the case when the rotation's DOFs have a small weight, in contrast, parts (d) to (f) show the case when the translation's DOFs have a small weight. Note that, the resulting trajectories are very different. When rotations are better (Fig. 4 (c)), the robot goes straight to the obstacle and places its sensor over it. Note that 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 it is better to translate (Fig. 4 (f)), the robot does not rotate any link and simply moves around the obstacle to reach the region behind.

Figure 5 shows an example where the robot must visit several regions to see the whole environment. Here, the global plan corresponds to the order of visiting the regions, such order minimizes the expected value of the time to find an object, and is computed using our algorithm proposed in [2]. Figure 5, parts (a), (b) and (c) show the case of low-cost rotation. Figures 5 (a) and (b) show snapshots of the robot path and the regions covered through the path. Figure 5 (c) shows the whole path. In this case, the robot chooses to see first the region associated to the hole in the wall. It requires several arm's rotations, then the

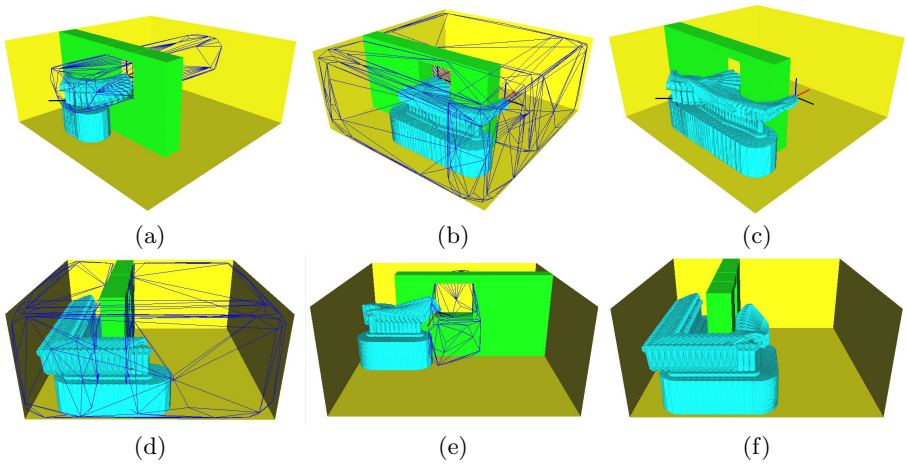


Fig. 5. Visiting several regions: Low-Cost Translation vs Low-Cost Rotation

robot translates as few as possible (because the translation is expensive) to cover the remaining regions. Note that the robot goes, just around the corner to cover the whole environment.

Figure 5, parts (d), (e) and (f) show the case of low-cost translation, the whole path is shown in figure 5 (f). Now, the robot chooses a path which requires a long translation around the wall, the robot see the region associated to the hole in the wall up to the end. In this second case, the robot avoids to stick its arm through a hole in the wall, because it requires several expensive rotations. The expected value of the time depends on both: 1) the volume of the regions and 2) the distance traveled (and hence the metric used to quantify it). Therefore, the order for visiting regions may change according to the metric used to quantified the path cost, thus generating robot's *behaviors*.

6 Conclusions and Future Work

We addressed the problem of searching for an object 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. The decomposition into regions gives us flexibility on exactly where to place the sensor, and their convexity guarantees that the entire cell will be visible from every point inside.

Based on this covering, we presented approaches for a mobile manipulator with an "eye-in-hand" sensor. We have also presented an approach to generate metric-dependent trajectories, that is, trajectories that compromise between moving the base and moving the robotic arm.

There are several extensions to the proposed problems that we think would be interesting to research. 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 is a good general a priori, given that to consider other types of pdfs, the type of object should be taken into account. However, in a scenario where one or more objects are searched several times, it should be possible to start the search assuming an uniform pdf and modify it according to the places where the objects are found. We leave this problem for future work.

Acknowledgments

This research was partially funded by CONACyT project 56754-F1, NSF-CONACyT Project J110.534 and CONCYTEG project 07-02-K662-097.

References

1. Sarmiento, A., Murrieta-Cid, R., Hutchinson, S.: A Sample-based Convex Cover for Rapidly Finding an Object in a 3-D environment. In: Proc. IEEE Int. Conf. on Robotics and Automation (2005)
2. Sarmiento, A., Murrieta-Cid, R., Hutchinson, S.: An Efficient Strategy for Rapidly Finding an Object in a Polygonal World. In: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (2003)
3. Ishiguro, H., Sogo, T., Barth, M.: Baseline Detection and Localization for Invisible Omnidirectional Cameras. International Journal of Computer Vision 58(3), 209–226 (2004)
4. Hert, S., Tiwari, S., Lumelsky, V.: A terrain-covering algorithm for an auv. Autonomous Robots 3, 91–119 (1996)
5. Chin, W.P., Ntafos, S.: Optimum watchman routes. Information Processing Letters 28, 39–44 (1988)
6. Goodman, J.E., O'Rourke, J. (eds.): Handbook of Discrete and Computational Geometry. CRC Press, Boca Raton (1997)
7. Hsu, D., Latombe, J.C., Motwani, R.: Path planning in expansive configuration spaces. In: Proc. IEEE Int. Conf. on Robotics and Automation (1997)
8. Kavraki, L.E., Svestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4) (June 1996)
9. Latombe, J.-C.: Robot Motion Planning. Kluwer Academic Publishers, Dordrecht (1991)
10. Leven, P., Hutchinson, S.: A Framework for Real-time Path Planning in Changing Environments. Int. Journal Robotic Res. 21(2), 999–1030 (2003)
11. O'Rourke, J.: Art Gallery Theorems and Algorithms. Oxford University Press, Oxford (1987)
12. Simeon, T., Laumond, J.P., Nissoux, C.: Visibility based probabilistic roadmaps. Advanced Robotics Journal 14(6) (2000)