A Motion Planner for Finding an Object in 3D Environments with a Mobile Manipulator Robot Equipped with a Limited Sensor

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 searching an object with a mobile robot in a known 3-D environment. We consider a 7 degrees of freedom mobile manipulator with an "eye-in-hand" sensor. The sensor is limited in both range and field of view. 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 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 generate candidate sensing configurations over the robot configuration space using sampling. 3) We determine an order for visiting sensing configurations using a heuristic.

We have implemented all our algorithms, and we present simulation results in challenging environments.

1 Introduction

In this paper, we address the problem of searching a static object with a mobile robot in a known 3-D environment. We consider a 7 degrees of freedom mobile manipulator with an "eye-in-hand" sensor. The sensor is limited in both range and field of view. Fig. 1 (a) shows this robot and the visibility region of the limited sensor, which has the shape of a frustum.

Furthermore, our planner aims to diminishing the expected value of the time for finding the object. This could be very useful in applications in which the time assigned to the task is limited or not completely known. 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.

In [2,5] 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. The present work addresses a related problem but in a 3D workspace with a mobile manipulator robot equipped with a limited sensor. Hence, 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. In [4] an approach has been proposed for finding an object with a mobile manipulator, but in [4] the sensor used by the robot was omnidirectional (i.e. it has a field of view of 360 deg.) and without range limit. Furthermore, the environments considered in [4] were small and geometrically simple. In this paper we propose a motion planner for the more challenging case in which the robot is equipped with a sensor limited in both range and field of view, and the robot is moving in large and geometrically complex 3D environments (see Fig 2).

Our search problem is related to art gallery problems and coverage. The traditional art gallery problem is to find a minimal placement of guards such that their respective visibility regions completely cover a polygon [10].

In coverage problems (e.g., [7,1]), 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.

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 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 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 cost to move between 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.

We use numerical optimization methods for diminishing the cost of moving the robot during the object finding task. To reduce the computational running time of this procedure, we select the most important robot degrees of freedom (DOFs) and carry out the optimization procedure only considering this reduced number of DOFs.

2 General Approach and Notation

In our object finding 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*.

In our formulation, we assume that the environment is known, but that we do not have any 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 [6]. We believe 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 [10]). 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 $V(q_i)$. The interior of the workspace $int(W)$ is the free space inside the 3D environment.

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 [8]. For this reason, we decided to use a sampling scheme to calculate an approximated visibility region of the limited sensor.

Our searching strategy is as follows: first the whole environment is divided into a set of convex regions. To divide the environment into convex regions we use the probabilistic convex cover proposed in [3][1]. This method divides the environment into a set called $\{C(r)\}$, so that the union of all $C(r)$ covers the whole environment, that is $\bigcup_r C(r) = int(W)$. $C(r)$ 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 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.

We believe that there exists an analogy between our searching protocol and painting a house, first the order for painting rooms (equivalent to the order for visiting the convex regions) in the house is established. Then every wall in each room is painted, this is equivalent to visit our sensing configurations, whose associated view frustums cover each single convex region.

Since the expected value of the time depends also on the cost of moving the robot between sensing configurations, we need to find shortest paths to move the robot. The actual paths depend on the metric used to measure cost. One way to define the cost between two configurations X and Y in a D-dimensional configuration space is

$$\|X - Y\|_\Lambda \equiv (X - Y)^T \Lambda (X - Y), \tag{1}$$

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).

[1] Notice that in [3] the environment is 3D, but the robot is a point equipped with an omnidirectional sensor without range limit.

To find the shortest path between two convex regions we use the wavefront expansion (called NF1) proposed in [9].

The two orders: (1) for visiting convex regions, and (2) for visiting sensing configurations inside a single convex region are established based on the utility function proposed in [5]. This utility function measures how convenient it is to visit a determined configuration from another, and is defined as follows:

$$U\left(q_k, q_j\right) = \frac{P\left(q_j\right)}{Time\left(q_k, q_j\right)}. \tag{2}$$

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.

The utility function in (2) 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 the algorithm proposed in [5] to search a reduced space. This algorithm is able to explore several steps ahead without incurring a too high computational cost. This algorithm is described in detail in [5].

3 Selecting Sensing Configurations

The method that we propose to cover each convex region is based on sampling. There have been many approaches that use samples to capture connectivity of high dimensional spaces, for example, [13], [12] 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.

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 position of the robot's base and five angles. One angle measures the orientation of the robot's base, and the other 4 measure the orientations of the arm's links. 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 (see [3]). 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.

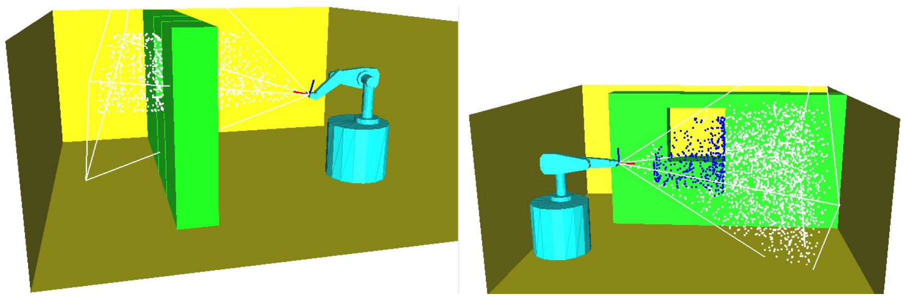
Notice that *a set of sensing configurations with minimal cardinality might not be the optimal one for our problem*, since the cost for reaching the configurations might be large if they are far from one another. Thus, our problem is different from the problem of only choosing the minimum number of configuration that collectively sense the whole environment: The art gallery problem. Since, even the easier art gallery problem is NP-hard [11], 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 [14]. That method is designed to cover the boundary of a polygonal –2D– workspace ∂W. In our problem, we aim to cover the free space inside the polyhedral –3D– environment representation.

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

$$V(q_{(i,r)}) = \bigcup_{s} s(r) \in int(\mathcal{F} \cap C(r)) \tag{3}$$

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 $C(r)$.



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

Fig. 1. Visibility Region and Set $\{s(t)\}$

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.

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:

$$\bigcup_s s(r) \in int(C(r)) = S(r) \tag{4}$$

Similar to [14], 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.

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.

4 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 cost to move between sensing configurations.

To cover a region with a limited sensor several sensing configuration are required. 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.

In the motion planning problem of generating collision free paths to move between configurations, we use a lazy collision checking scheme [15]. The sensing configurations are connected in the graph without checking whether or not 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 [16] 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.

Note that we use the convex regions to facilitate the object finding task. There are four 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 point 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. Fourth, once a global plan is generated if the environment changes locally then the global plan can also be modified *locally*, only considering the convex regions related to the change in the map.

4.1 Optimizing a Reduced Number of DOFs

The order of visiting regions is established based on a path of optimal cost between regions. This optimal path is computed with the wavefront expansion algorithm described in [9]. Regardless the optimization algorithm used, the problem of finding the path of locally optimal cost (between two robot's configurations) for a robot with several DOFs can take significant amount of time, specially for a global path including a large number of inter-medial sensing configurations.

Indeed, in our experiments we have found that bottleneck limitation regarding the running time of our algorithms corresponds to compute the optimal paths for a large number (4 or more) of DOFs. To mitigate this problem, we select some

DOFs and carry out the optimization procedure over this reduced number. 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 other do not produce collisions between the robot and the obstacles.

In terms of energy and time is more costly to translate a mobile manipulator robot than rotate its base or its arm links. In fact, if the planner is able to coordinate the translation of the robot base and the rotations of 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. Furthermore the environment in which the robot resides is large, and the robot has to translate a long distance (equivalent to time when the robot moves with saturated speed) to search the object. Consequently, we consider that the coordinates (x, y) defining the position of the robot's base are the most important DOFs for this problem. Hence, we optimize only these two DOFs.

5 Simulation Results

All the results presented in this paper were obtained with a regular PC running Linux OS, equipped with an Intel Pentium IV microprocessor and 1 Gigabyte of RAM. Figures from 2 (a) to 2 (d) show snapshots of the object finding task in an indoor environment. This environment cannot be searched by using a 2D projection; since information about the 3D structure will be lost. Notice the stairs, the windows, the lamp and the furniture.

For this environment, our convex cover algorithm, produced 47 convex regions. This convex cover decomposition was computed by our software in 2 minutes and 40 seconds. 552 sensing configurations were needed to cover the whole environment with the limited sensor. The running time of our software to generate these 552 sensing configurations, link them in a graph in C, and computing an order to visit them was 1 hour and 33 minutes.

Figure 2 (a) shows with two (blue) meshes 2 convex regions, called respectively region $C(A)$ and $C(B)$. Convex regions $C(A)$ and $C(B)$ are consecutive in the order to visit convex regions. Figure 2 (a) also shows the path to move between regions $C(A)$ and $C(B)$.

Figure 2 (b) shows the 19 sensing configurations needed to collectively cover region $C(A)$. Figure 2 (c) 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 C of 7 dimensions.

Only $\frac{1}{10}$ of the total number of paths to sense the whole map was computed with a RRT. All the other times, a straight line in C was enough to find collision free paths.

Figure 2 (d) shows the global robot's path to visit all the convex regions. In this global path only the DOFs (x, y) defining the robot's base position are optimized. The time to compute this global path was 15 minutes and 9 seconds.

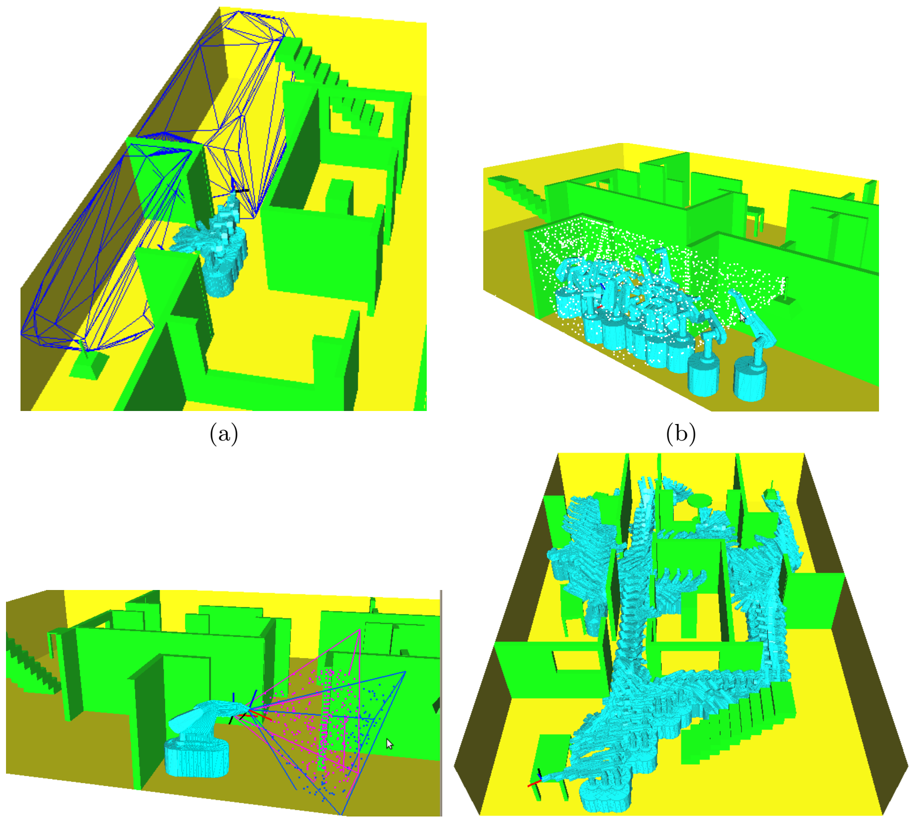

(a) (b) (c) (d)

Fig. 2. Finding an object with a limited sensor

The total running time of our software to compute a plan for sensing the whole environment with a limited sensor was 1 hour and 50 minutes. The expected value of the time is 100.44 units.

To our knowledge, this is the first method able to solve this problem with a 7 DOFs robot equipped with a limited sensor in a realistic 3D map.

6 Conclusion

In this paper, we have proposed a motion planner for finding an object in 3-D environments with a robot equipped with a limited sensor. Our main contributions are: (1) We have proposed 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 have proposed a method to select candidate sensing configurations having convenient properties for diminishing the average time to find the searched object. (3) We have proposed the strategy of selecting the most important DOFs to be optimized. This strategy significantly reduces the computational running time to find the object. We have implemented all our algorithms, and we have presented simulation results in challenging environments.

References

1. Acar, E.U., Choset, H., Atkar, P.N.: 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. Sarmiento, A., Murrieta-Cid, R., Hutchinson, S.: A Multi-robot Strategy for Rapidly Searching a Polygonal Environment. In: Lemaître, C., Reyes, C.A., González, J.A. (eds.) IBERAMIA 2004. LNCS (LNAI), vol. 3315, pp. 484–493. Springer, Heidelberg (2004)
3. 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, IEEE-ICRA 2005 (2005)
4. Sarmiento, A., León-Espinoza, J., Murrieta-Cid, R., Hutchinson, S.: A Motion Planning Strategy for Rapidly Finding an Object with a Mobile Manipulator in 3-D Environments. In: Gelbukh, A., Morales, E.F. (eds.) MICAI 2008. LNCS (LNAI), vol. 5317, pp. 562–572. Springer, Heidelberg (2008)
5. Sarmiento, A., Murrieta, R., Hutchinson, S.A.: An Efficient Motion Strategy to Compute Expected-Time Locally Optimal Continuous Search Paths in Known Environments. Advanced Robotics 23(12-13), 1533–1569 (2009)
6. LaValle, S.: Planning Algorithms. Cambridge University press, Cambridge (2006)
7. Hert, S., Tiwari, S., Lumelsky, V.: A terrain-covering algorithm for an auv. Autonomous Robots 3, 91–119 (1996)
8. Goodman, J.E., O'Rourke, J. (eds.): Handbook of Discrete and Computational Geometry. CRC Press, Boca Raton (1997)
9. Latombe, J.-C.: Robot Motion Planning. Kluwer Academic Publishers, Dordrecht (1991)
10. O'Rourke, J.: Art Gallery Theorems and Algorithms. Oxford University Press, Oxford (1987)
11. Shemer, T.: Recent Results in Art Galleries. Proc. IEEE 80(9), 1384–1399 (1992)
12. Hsu, D., Latombe, J.C., Motwani, R.: Path planning in expansive configuration spaces. In: Proc. IEEE Int. Conf. on Robotics and Automation, IEEE-ICRA 1997 (1997)
13. 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)
14. González, H.H., Latombe, J.-C.: A Randomized Art-Gallery Algorithm for Sensor Placement. Proc. 17th ACM Symp. on Computational Geometry, SoCG 2001 (2001)
15. Sanchez, G., Latombe, J.C.: A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking. In: Jarvis, R.A., Zelinsky, A. (eds.) ISRR 2001. STAR (2003)
16. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. International Journal of Robotics Research 20(5), 378–400 (2001)