This is an accepted version of the article: Faigl, J., Kulich, M., and Přeučil, L. (2011). **A Sensor Placement Algorithm for a Mobile Robot Inspection Planning**. *Journal of Intelligent and Robotic Systems*, 62(3-4), which has been published in a final form at https://doi.org/10.1007/s10846-010-9449-0.

A Sensor Placement Algorithm for a Mobile Robot Inspection Planning

Jan Faigl · Miroslav Kulich · Libor Přeučil

Received: 3 December 2009 / Accepted: 12 July 2010 / Published online: 30 July 2010

Abstract In this paper, we address the inspection planning problem to "see" the whole area of the given workspace by a mobile robot. The problem is decoupled into the sensor placement problem and the multi-goal path planning problem to visit found sensing locations. However the decoupled approach provides a feasible solution, its overall quality can be poor, because the sub-problems are solved independently. We propose a new randomized approach that considers the path planning problem during solution process of the sensor placement problem. The proposed algorithm is based on a guiding of the randomization process according to prior knowledge about the environment. The algorithm is compared with two algorithms already used in the inspection planning. Performance of the algorithms is evaluated in several real environments and for a set of visibility ranges. The proposed algorithm provides better solutions in both evaluated criterions: a number of sensing locations and a length of the inspection path.

Keywords Sensor placement · Mobile robotics · Inspection path planning · Art gallery problem

1 Introduction

The inspection planning, a problem to "see" the workspace $\mathcal{W}$, is a robotic instance of the *Watchman Route Problem* (WRP) [1], which deals with finding a shortest path in a polygon P such that all points of P are visible from at least one point at the path.

The work has been supported by the Ministry of Education of the Czech Republic under program "National research program II" by the project 2C06005.

J. Faigl (✉) · M. Kulich · L. Přeučil
Department of Cybernetics, Faculty of Electrical Engineering,
Czech Technical University in Prague, Technická 2,
166 27, Prague 6, Czech Republic
e-mail: xfaigl@labe.felk.cvut.cz

The problem is NP-hard for a polygon with holes and the first heuristic approach has been proposed relatively recently in [2]. An alternative approach is based on the problem decomposition into the set cover problem and the multi-goal path planning problem. The problems can be formulated as the *Art Gallery Problem* (AGP) and the well-known *Traveling Salesman Problem* (TSP) [3]. The AGP stands to find a minimal number of guards to cover a polygonal environment. The guard is a point for which a star-shaped visibility polygon is computed to cover a part of the environment. Both problems are known to be NP-hard, even in the case of a polygon without holes [4].

The visibility of real sensors is limited, therefore restricted visibility should be considered. Two variants of the WRP are studied for a visibility range limited to a fixed distance d [5]. The *d-watchman route problem* is a variant to see only the boundary of the polygon, while the *d-sweeper route problem* aims to sweep a polygonal floor using a circular broom of the radius d, such that the total travel of the broom is minimized [6]. These problems are studied for restricted classes of polygons in the computational geometry domain, an overview of particular results can be found in [7, 8].

For the decoupled approach, the authors of [9] considered three visibility constraints and called the problem the *sensor placement* rather than the AGP. In addition to the basic constraint that a line-of-sight does not intersect the boundary of the workspace, they considered visibility range and incidence constraints. The decoupled approach is motivated by the practical sensing limitation, where a cost of the sensing is greater than a cost of moving, e.g. a high quality measurement needs a significant amount of time and it cannot be taken during the robot movement. Therefore the number of sensing locations have to be minimized. Despite the fact that the decoupled approach provides a feasible solution of the inspection planning, it was criticized in [10], because even in the case that both sub-problems are solved optimally, due to its independent solution, the overall performance can be poor.

Table 1 Used symbols and notation

Symbol	Description
$\boldsymbol{G}$	A set of sensing locations
$g \in \boldsymbol{G}$	A sensing location (a guard)
$\mathcal{C}_{\text{free}}$	Free space (all robot configurations not colliding with obstacles)
$A \oplus B$	Minkowski sum
$A \ominus B$	Minkowski difference
$\mathcal{W} \subset \mathbb{R}^2$	A robot workspace, represented by the polygonal domain
n_v, n_g	The number of polygon (map) vertices, the number of guards
k	The number of vertices of a polygonal representation of a disk
$G(V, E)$	A graph defined by a set of vertices V and a set of edges E
$K(v)$	The relevance measure, see Eq. 1
r	A value of minimal allowed relevance
$\|.\|$	An area of a polygon
visible$(\mathcal{W}, d, p)$	A visibility polygon, i.e. a part of $\mathcal{W}$ d-visible from the point p
$\delta\boldsymbol{B}$	Boundary (border of a shrunk free space)
$\boldsymbol{I}, \boldsymbol{E}$	A collection of polygons in the interior, resp. exterior, of $\mathcal{W}$ according to $\delta\boldsymbol{B}$
m	The number of random samples in the RDS algorithm
$\mathcal{N}(0, \sigma^2)$	Two dimensional normal (Gaussian) distribution with zero mean and σ variance

The author noted that such decomposition works well if a coverage of considered views do not overlap or those with large coverage overlap are close to each other. However the critique makes sense, a systematic study of sensor placement algorithms according to the related route planning problem has not been found in the literature. In this paper, a new algorithm for the sensor placement problem is presented and compared with two algorithms. The performance of the algorithms is evaluated with respect to the number of found sensing locations and the length of the inspection path to visit these locations in a set of real environments.

The paper is organized as follows. An overview of related work and applications of the sensor placement are presented in the next section. The problem is specified in Section 3 and three examined algorithms are described in Section 4. The experimental results are presented in Section 5. Finally, the presented results are discussed and further possible extensions are proposed in the conclusion, Section 6. Symbols and notation used in the rest of the paper are shown in Table 1.

2 Related Work

Randomized sampling based approaches are suitable for consideration of real sensor limitations as they are able to address different visibility constraints. Two sampling based algorithms determining a set of sensing locations to cover the workspace boundary with consideration of the visibility constraints were proposed in [11]. The first algorithm is a greedy approach to find a near-optimal subset of sampled points to cover the boundary. The second algorithm at first places a point p on the boundary of the unseen part of the polygon and then selects a point with the highest coverage from sampled points in the visibility polygon of p. The algorithm is called *Randomized Dual Sampling* (RDS) and it has been utilized in the inspection planning [3] and in the surveillance systems [12].

The surveillance systems are related to the sensor placement problem, because restricted visibility has to be considered in these systems: cameras are typically mounted on walls, have limited field of view and limited range for sufficient sharp focus in an image. A solution of the problem based on a discretization of the environment into a two dimensional grid was presented in [13]. The problem formulated as the Binary Integer Programming (BIP) has been presented in [12]. Authors compared the performance of the RDS and greedy heuristic with the exact solution. They reported suitability of heuristic approaches and their good approximation of the BIP solution. However due to the lack of computational resources only small problems have been examined. A similar problem was addressed in [14], the problem is the optimal sensor placement to create a wireless sensor network where each sensor has three parameters: sensing range, field of view and orientation. A limited bandwidth of a wireless links was also considered. The problem is formulated as the Integer Linear Programming (ILP) and solved by CPLEX 10.1 solver. The largest solved problem had 200 placement sites and 70 control points and was solved in 30 826 seconds on 3 GHz CPU.

A deterministic algorithm based on a decomposition of a polygon with holes into a set of convex sub-polygons was presented in [15]. Authors considered a panoramatic camera with 360° field of view and restricted visibility range to capture an image with a sufficient level of details. The algorithm is able to address both variants of

the inspection planning: to "see" whole area of the workspace or only its borders. The authors presented an analysis of the required computational time that increases proportionality to a number of found guards and reported solved problems up to eight thousands of guards for the restricted visibility range.

The sensor placement problem is also studied in the so-called View Point Planning (VPP). A solution of the VPP aims to provide a plan for an automated 3D object recognition and inspection. To provide a "good" view it is necessary to consider particular sensing constraints, which depend on a sensing device and recognition capabilities [16]. Two costs can be considered in the VPP, the sensing cost and travel cost from one view point to the next view point. The combination of the view and travel costs is addressed in the formulated Traveling VPP [10], which is defined on a given set of view points. The solution (probably the first unified approach in robotics) is based on the ILP formulation and rounding algorithm called Round and Connect. The watchman route problem is reduced to the Traveling VPP by a finite number of viewpoints that are found by the proposed sampling algorithm. The number of viewpoints does not depend on geometric parameters of the polygon (with holes), the viewpoints are found in $O(n^{12})$, where n is the number of polygon vertices.

A two dimensional sensor placement is also useful for the full 3D environment reconstruction to provide an initial plan where measurements should be taken [17] and in the exploration task [18] with the *next-best-view* navigation strategies [19].

Practical applications of the randomized sampling based algorithm have been reported by several authors, however the related path planning problem is solved independently. In this paper, we evaluate three algorithms according to a number of sensing locations and a length of the path over found set of sensing locations. Two state-of-the-art algorithms have been selected: the deterministic *Convex Polygon Partitioning* (CPP) algorithm [15] and the RDS algorithm [20]. The main advantage of these two approaches is their simplicity and efficiency, which allows real-time planning in robotic applications. The third algorithm is a new algorithm called *Boundary Placement* (BP), which tries to incorporate the related path planning problem into the sensor placement part.

3 Problem Statement

An environment is a priori known and it is represented in a form of a geometrical map as the polygonal domain (polygon with holes). The problem is to find a set of sensing locations $\boldsymbol{G}$ to cover the environment. A sensing location have to be reachable by the mobile robot, therefore $g \in \boldsymbol{G}$ have to be in the free space of the C-space, $g \in \mathcal{C}_{\text{free}}$. A rigid body of the robot with a differential nonholonomic drive can be modeled by a disk that allows representation of the reachable free space by a shrunk polygon determined by the Minkowski sum operation. The obstacles are grown according to the Minkowski sum definition $A \oplus B = \{x+y \mid x \in A, y \in B\}$ while the border polygon is shrunk by the Minkowski difference that can be defined as $A \ominus B = A \oplus (-B)$ [21]. In such a shrunk free space a point robot can be assumed and $\mathcal{C}_{\text{free}}$ can be represented by the polygonal domain $\mathcal{W}$ denoting the robot workspace.

The visibility range of the sensor is considered to be limited to the distance d and two points p and q in a polygon $\boldsymbol{P}$ are called *d-visible*, if the line segment joining them

is contained in $\boldsymbol{P}$ and the segment length is less or equal to d. The visibility restricted to the range d is modeled by a disk having radius d and consisting of a given number of vertices. The sensor placement problem can be formulated as follows.

Sensors Placement Problem - For a given workspace $\mathcal{W} \subset \mathbb{R}^2$ reachable by the mobile robot, find a set of sensing locations $\boldsymbol{G}$ such that every point of $\mathcal{W}$ is d-visible from at least one point of $\boldsymbol{G}$.

To follow a notation of the AGP, a sensing location is also called a guard in the rest of this paper. Once guards are found, the inspection planning is formulated as the routing problem to find an order to visit the guards such that the total length of the path is minimized. Shortest paths between guards can be found as the shortest path roadmap constructed from the visibility graph, e.g. in $O((n_v + n_g)^2)$ [22], where n_v denotes the number of map vertices and n_g is the number of guards. The path planning problem is formulated as the TSP on a graph $G(V, E)$, where V denotes guards and E is a set of edges with costs derived from the lengths of the shortest paths between guards, the TSP can be than solved by a TSP solver. Without loss of generality $G(V, E)$ is assumed to be complete.

4 Sensor Placement Algorithms

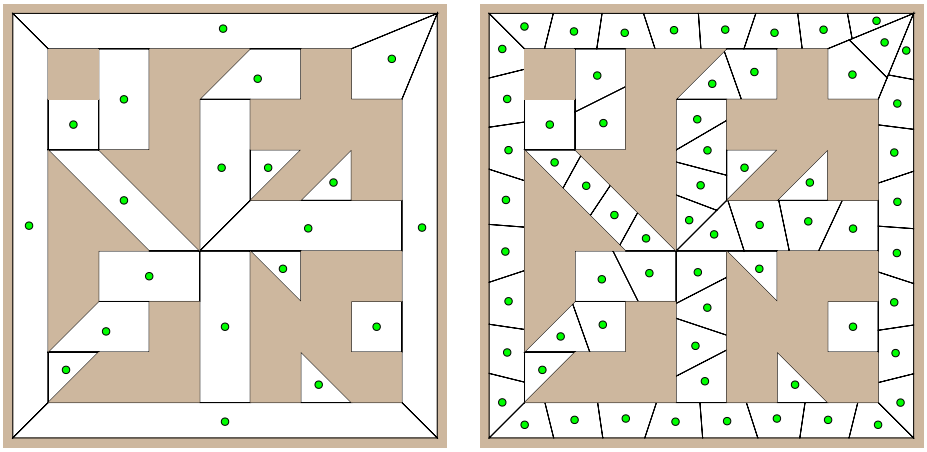
4.1 Convex Polygon Partitioning - CPP

A deterministic sensor placement algorithm based on the decomposition of a polygonal environment representation into a set of convex polygons has been proposed in [15]. Each convex polygon is covered by one guard and to satisfy the restricted visibility range constraint, a distance from a guard to vertex of the convex polygon has to be less than the visibility range d. If a convex polygon is too large, it is divided into convex sub-polygons until each sub-polygon is not covered by one guard with the omnidirectional view. The primal convex partition is found by Seidel's algorithm [23]. The total complexity is linear with the number of found guards [15].

The number of found guards depends on the partition to convex polygons. For a polygon with very small segments, vertices can cause additional convex polygons, which have to be covered by additional guards. This issue can be partially addressed by the polygon filter technique described in Section 4.1.1. Examples of found guards and particular convex sub-polygons are shown in Fig. 1.

4.1.1 Polygon Filtering

However a geometric representation is memory efficient in comparison to a grid based representation, the number of vertices can be still unnecessarily high. The number of map vertices affects the performance of the used algorithm. For example if two vertices of a non-convex polygon are very close, removing one of them can lead to a convex polygon, which can be easily covered by one guard placed at any position inside the polygon. These reasons lead to reduce a number of unnecessary vertices by a pre-processing of the polygonal representation of the free space, while the polygon shape is preserved.



(a) (b)

Fig. 1 Found guards by the CPP algorithm

A polygon filter technique based on the relevance measure [24] is one of the suitable algorithms to remove unnecessary vertices. It is depicted in Algorithm 1.

Algorithm 1: Polygon filter based on relevance measure

Input: $\boldsymbol{P}$ - polygon
Input: r –*minimal allowed value of relevance*
Result: A filtered polygon $\boldsymbol{P}$
Require: $|P| > 3$.

```
do
    v_i ← argmin_{v∈P} K(v)
    r_m ← K(v_i)
    if r_m < r then
        P ← P \ v_i
while r_m < r AND |P| > 3
```

The $K(v)$ is a relevance measure based on the Euclidean distance [25]

$$K(v_i) = |v_{i-1}, v_i| + |v_i, v_{i+1}| - |v_{i-1}, v_{i+1}|, \tag{1}$$

where v_{i-1} and v_{i+1} are neighbouring vertices of the vertex v_i in the polygon $\boldsymbol{P}$.

The polygonal representation $\mathcal{W}$ consists of a border polygon and a collection of hole polygons. Because of each polygon is filtered independently, the consistence of the filtered polygon with holes has to be validated. To ensure consistence of the resulting polygon, the final polygon with holes can be constructed by the Boolean operations, i.e. all filtered holes are subtracted from the filtered border polygon.

4.2 Randomized Dual Sampling Schema—RDS

The idea of the RDS algorithm to find a minimal set of guards is based on sampling the constraints of the problem (the points to be covered) instead of its domain [20].

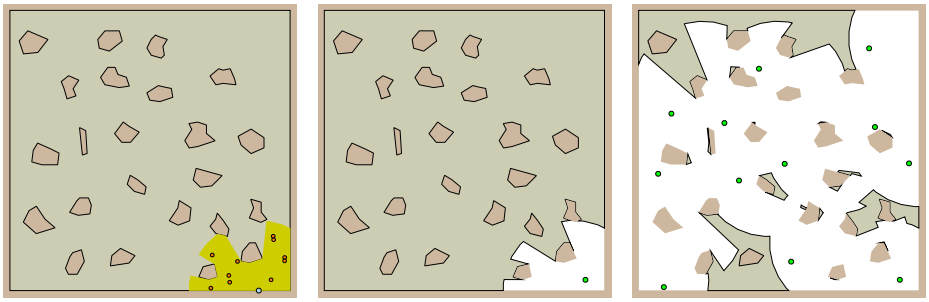
The algorithm has been proposed to find a set of guards to cover the boundary of the free space, therefore it needs to be modified to address covering of the workspace interior. The extension is straightforward, instead of the total length of the uncovered boundary an area of the uncovered part of the free space is considered. The sampling procedure is summarized in Algorithm 2, where $|.|$ denotes an area of the particular polygonal part of the workspace $\mathcal{W}$. The algorithm performs in two steps. At first, the boundary is sampled by a point and its visibility polygon is computed. After that, the polygon is sampled m times and a point with the highest coverage, i.e. the largest area of the visibility polygon, is denoted as a new guard. The visibility polygon of the new guard is subtracted from the uncovered free space and the process is repeated until the whole free space is covered. The algorithm is complete and it is terminated after a finite number of iterations, because at each iteration a random point is generated at the border of $\boldsymbol{U}$ and the visibility polygon of the new guard is subtracted from $\boldsymbol{U}$. An example of the algorithm performance is shown in Fig. 2.

Algorithm 2: Randomized dual sampling schema

Input: $\mathcal{W}$ - the workspace to be covered
Input: d - the maximal sensing range
Input: m - the number of samples
Result: $\boldsymbol{G}$ - a set of found guards (sensing locations)

$\boldsymbol{U} \leftarrow \mathcal{W}$ `// set the uncovered free space`
while $|\boldsymbol{U}| > 0$ **do**
  $p_b \leftarrow$ *select random point at border of* $\boldsymbol{U}$
  $\boldsymbol{V} \leftarrow \text{visible}(\mathcal{W}, d, p_b)$
  $\{p_1, p_2, \ldots, p_m\} \leftarrow$ *random points in* $\boldsymbol{V}$
  $p^\star \leftarrow \text{argmax}_{p_i \in \{p_1, p_2, \ldots, p_m\}} |\boldsymbol{U} \cap \text{visible}(\mathcal{W}, d, p_i)|$
  $\boldsymbol{G} \leftarrow \boldsymbol{G} \cup \{p^\star\}$
  $\boldsymbol{U} \leftarrow \boldsymbol{U} \setminus \text{visible}(\mathcal{W}, d, p^\star)$

The complexity of the algorithm depends on the computation of the visibility polygon that can be done in $O(n_v \log n_v)$, where n_v denotes the number of vertices



(a) (b) (c)

Fig. 2 An example of the RDS performance, small disks denote random points; **a** an initial random point p_b (in *blue*) at the border of uncovered free space $\boldsymbol{S}$, its restricted visibility polygon $\boldsymbol{V}$ and a set of random points, **b** the reduced uncovered free space $\boldsymbol{S}$ after the first guard has been found, **c** a covered free space by several found guards

of $\mathcal{W}$. The restricted visibility is computed as an intersection of the visibility polygon and a disk. The disk is formed from k edges and its radius is the visibility range d. Newly covered portion of the free space is subtracted from U that can increase a number of vertices up to $kn_v n_g$, where n_g is the number of found guards. The overall complexity can be bounded by $O(mn_v n_g \log(n_v n_g))$, where m is the number of random samples.

It should be noted that the RDS algorithm has been developed to consider two types of constraints, the visibility range and the incident constraint for the laser rangefinder sensing beam. The incident constraint models a situation when a laser beam is not reflected from the surface, because the incident angle of the beam with the surface normal is too wide. The constraint cannot be applied for covering the whole free space and it is not considered, however it can be used for the boundary cover.

4.3 Boundary Placement—BP

The BP algorithm tries to consider a length of the inspection path during the sensor placement that is based on the randomization process of the RDS algorithm. The randomized sampling is guided by a priori knowledge about the environment structure and positions of already found guards. The main idea follows greedy principle and suggestion to do not place guards unnecessary far from each other. Such a procedure can potentially lead to a higher number of guards, but the guards should be placed close to each other and the total traveled path to visit all guards is expected to be shorter. Let us review the ideas behind the algorithm design.

From the path planning point of view, it is not necessary to move the robot closer to walls (or obstacles) than in a perimeter of the visibility range. This consideration leads to place guards firstly in a pre-specified distance from the obstacles and then place additional guards to cover the rest of the uncovered free space. The primal guards positions are at the boundary of a shrunk free space by the distance b, which is close to the visibility range. The boundary represents prior knowledge how to sample the free space. The main idea is demonstrated in Fig. 3a. The boundary is represented by inscribed line segments, its distance to the obstacle is very close to the restricted visibility range. After placing guards at the boundary (notice the guards positions also satisfy possible incident constraint) the uncovered free space is divided into two sets of regions. The first set contains polygons inside the boundary, therefore it is called

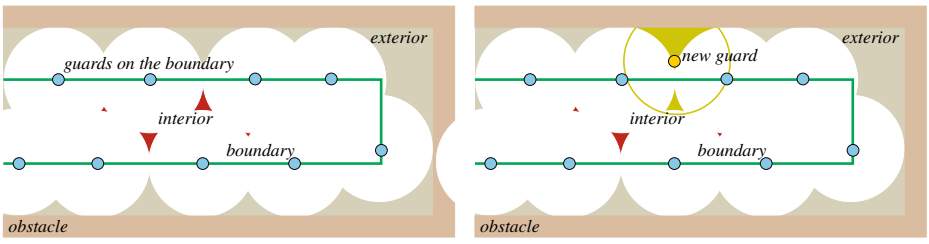


Fig. 3 Principle of the boundary placement algorithm

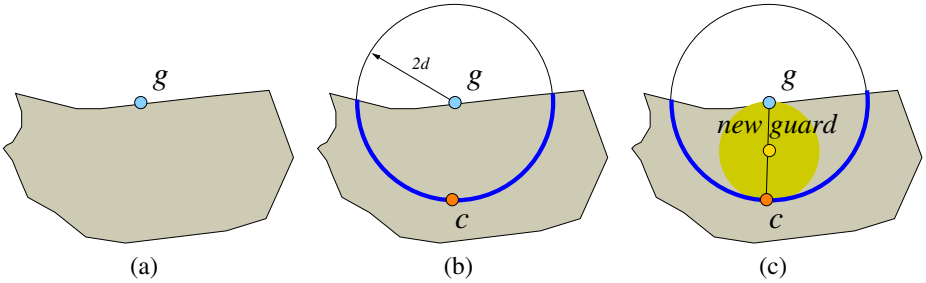


Fig. 4 Large region cover strategy

interior, the set is represented by dark polygons. The important property of these polygons is that they are not in contact with obstacles. Regions outside the boundary represent the second set called *exterior*. To cover a part of the exterior a new guard can be placed in a certain distance from an already placed guard, so the new guard is placed close to the guard. A path connecting found guards is mainly affected by the guards at the boundary, while added guards do not lead to significant change of the path direction. This idealized case demonstrates the main idea behind the algorithm design.

The Boundary Placement algorithm consists of four parts. The first three parts correspond to covering the boundary, interior and exterior sets. The fourth part is a post-processing procedure to adjust the number of found guards by replacing two very close guards by one guard with the same coverage.

The sensor placement procedure follows the randomized schema of the RDS algorithm, but the second sampling is replaced by two heuristic strategies.

The large region cover strategy firstly selects a random point g at the border of the uncovered region, which is not a part of an obstacle, see Fig. 4a. Then the midpoint c of the longest part of the circle (with the radius $2d$, where d is the visibility range) lying inside the uncovered region is determined, Fig. 4b. Finally a new guard is placed in the middle of the segment (g, c), see Fig. 4c.

The small region cover strategy also starts with a random sample point p at the border, which is not a part of an obstacle, see Fig. 5a. Then the closest already found guard g, which is directly visible from p, is determined. If such a guard is not found, the point p is used as a new guard. Otherwise a new guard is placed at the segment

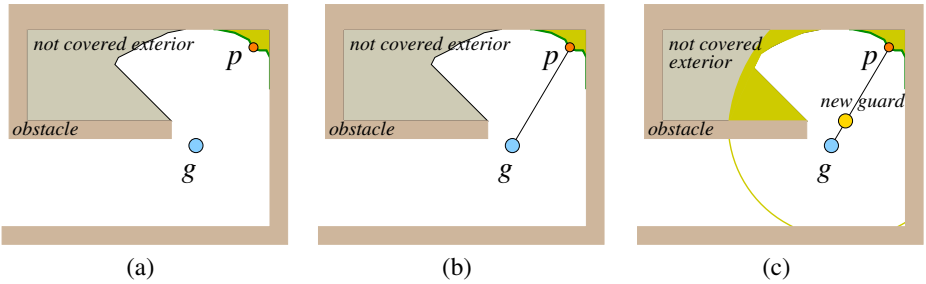


Fig. 5 Small region cover strategy

(g, p) close to g as much as possible, like in Fig. 5c. The new guard should cover same portion of the uncovered free space as the point p. To be precise, p lies at a border of one of the region that is a part of the set of regions. Only this particular region is considered to be covered, i.e. only its area covered by p is considered.

The BP algorithm is summarized in Algorithm 3. The second and third parts of the algorithm are identical, except the set of uncovered free space $\boldsymbol{I}$, resp. $\boldsymbol{E}$. It is expected that the covering interior will also cover a part of the exterior, that is why the covering interior precedes the covering exterior. The symbol δ denotes a border of the polygon. The polygon is an open set, thus the border is a difference between the closure of the polygon and the polygon. Both the interior and exterior are collections of polygons, hence a particular polygon from the collection is denoted as $\boldsymbol{I}_p \in \boldsymbol{I}$, resp. $\boldsymbol{E}_p \in \boldsymbol{E}$.

The cover strategy is the *large region cover strategy* or *small region cover strategy*. It is selected according to the area of the particular polygon $\boldsymbol{I}_p$, resp. $\boldsymbol{E}_p$,

$$|\boldsymbol{I}_p| \geq \begin{cases} \mu_I \pi d^2 & \text{large region cover strategy,} \\ \text{otherwise} & \text{small region cover strategy.} \end{cases} \tag{2}$$

Algorithm 3: Boundary Placement Algorithm

Input: $\mathcal{W}$ - the workspace to be covered
Input: d - the visibility range
Input: $\delta\boldsymbol{B}$ - the boundary (border of shrunk free space)
Result: $\boldsymbol{G}$ - a set of found guards (sensing locations)

$\boldsymbol{U} \leftarrow \mathcal{W}$ // set the uncovered free space
while $|\delta\boldsymbol{B}| > 0$ **do**
  $g \leftarrow$ *select random point at* $\delta\boldsymbol{B}$
  $\delta\boldsymbol{B} \leftarrow \delta\boldsymbol{B} \setminus \text{visible}(\mathcal{W}, d, g)$
  $\boldsymbol{G} \leftarrow \boldsymbol{G} \cup \{g\}$ // add the boundary guard to the set of guards
  Part I - *Boundary Cover*
$\boldsymbol{T} \leftarrow \boldsymbol{U} \setminus \bigcup_{g_j \in \boldsymbol{G}} \text{visible}(\mathcal{W}, d, g_j)$
$\boldsymbol{I} \leftarrow \{\boldsymbol{I}_i | \boldsymbol{I}_i \in \boldsymbol{T} \wedge (\boldsymbol{I}_i \cap \delta\mathcal{W} = \emptyset)\}$ // the interior set
while $|\boldsymbol{I}| > 0$ **do**
  $p \leftarrow$ *select random point at* $\delta\boldsymbol{I}, p \subset \boldsymbol{I}_p, \boldsymbol{I}_p \in \boldsymbol{I}$
  $g \leftarrow \text{cover\_strategy}(\mathcal{W}, d, p, |\boldsymbol{I}_p|)$
  $\boldsymbol{I} \leftarrow \bigcup_{\boldsymbol{I}_i \in \boldsymbol{I}} \boldsymbol{I}_i \setminus \text{visible}(\mathcal{W}, d, g)$ // cover the interior set
  $\boldsymbol{G} \leftarrow \boldsymbol{G} \cup \{g\}$ // add interior guard to the set of guards
  Part II - *Interior Cover*
$\boldsymbol{E} \leftarrow (\boldsymbol{U} \setminus \bigcup_{g_j \in \boldsymbol{G}} \text{visible}(\mathcal{W}, d, g_j)) \setminus \delta\mathcal{W}$ // the exterior set
while $|\boldsymbol{E}| > 0$ **do**
  $p \leftarrow$ *select random point at* $\delta\boldsymbol{E}, p \subset \boldsymbol{E}_p, \boldsymbol{E}_p \in \boldsymbol{E}$
  $g \leftarrow \text{cover\_strategy}(\mathcal{W}, d, p, |\boldsymbol{E}_p|)$
  $\boldsymbol{E} \leftarrow \bigcup_{\boldsymbol{E}_i \in \boldsymbol{E}} \boldsymbol{E}_i \setminus \text{visible}(\mathcal{W}, d, g)$ // cover the exterior set
  $\boldsymbol{G} \leftarrow \boldsymbol{G} \cup \{g\}$ // add the exterior guard to the set of guards
  Part III - *Exterior Cover*
$\boldsymbol{G} \leftarrow \text{reduce}(\mathcal{W}, d, \boldsymbol{G})$ // part IV

The parameter μ_I, resp. μ_E, represents multiplication of the visibility disk area and it is used to estimate size of the particular uncovered region $\boldsymbol{I}_p$, resp. $\boldsymbol{E}_p$.

The first three parts of the algorithm are similar to the randomized approach of the RDS algorithm, which uses a set of random points to select the best random point as a new guard. In these three parts, the same RDS schema of the random point

selection can be used in similar manner. Such selection is not a part of Algorithm 3, the extension is straightforward.

The fourth part of the algorithm is a post-processing optimization to reduce a number of found guards. A guard can be placed close to previously found guards, therefore guards can cover large portion of the same space. Two such guards can be possibly replaced by one guard with the same coverage. A difference between guards coverage can be covered by other guards, so only particular not covered part of the free space can be considered. The optimization procedure is following.

1. Let $\mathcal{W}$ is a polygonal map and $\boldsymbol{G}$ is a set of guards.
2. Create pairs of mutually visible guards $\{G_1, \ldots, G_n\}$ that are closer than the visibility range d, $G_i = \{g_i, g_i'\}$, $g_i \neq g_i'$, $|(g_i, g_i')| \leq d$, $g_i \in \boldsymbol{G}$, $g_i' \in \boldsymbol{G}$.
3. Sort the pairs according to distance between guards and select pairs with the shortest distance between guards such that each guard is only in one such pair, $\boldsymbol{G}_P = \{G_1, \ldots, G_k\}$, $g_i \in G_i$, $g_i \notin G_j$, $i \neq j$, $i, j \in \{1, \ldots, k\}$.
4. Compute coverage of guards, which are not in the selected pairs, the uncovered free space is denoted as $\boldsymbol{U}$.
5. Select pair G_s from the $\boldsymbol{G}_P$ with the closest guards. Compute coverage of the guards $G_s = \{g_s, g_s'\}$ as $\boldsymbol{P}_s = \text{visible}(\mathcal{W}, d, g_s) \cap \boldsymbol{U}$, $\boldsymbol{P}_s' = \text{visible}(\mathcal{W}, d, g_s') \cap \boldsymbol{U}$ and coverage of the midpoint $p = \text{midpoint}(g_s, g_s')$, $\boldsymbol{P_p} = \text{visible}(\mathcal{W}, d, p) \cap \boldsymbol{U}$. Select guards according to following criterions.
   (a) If $(\boldsymbol{P}_s \cup \boldsymbol{P}_s') \setminus \boldsymbol{P}_p = \emptyset$ then replace guards g_s, g_s' by the new guard p, $\boldsymbol{G} \leftarrow \{p\} \cup \boldsymbol{G} \setminus \{g_s, g_s'\}$ and update uncovered free space $\boldsymbol{U} \leftarrow \boldsymbol{U} \setminus \boldsymbol{P}_p$, go to step 6.
   (b) If $|\boldsymbol{P}_s| > |\boldsymbol{P}_s'| \wedge \boldsymbol{P}_s' \setminus \boldsymbol{P}_s = \emptyset$ then use g_s, $\boldsymbol{G} \leftarrow \boldsymbol{G} \setminus \{g_s'\}$ and update uncovered free space $\boldsymbol{U} \leftarrow \boldsymbol{U} \setminus \boldsymbol{P}_s$, go to step 6.
   (c) If $\boldsymbol{P}_s \setminus \boldsymbol{P}_s' = \emptyset$ then use g_s', $\boldsymbol{G} \leftarrow \boldsymbol{G} \setminus \{g_s\}$ and update uncovered free space $\boldsymbol{U} \leftarrow \boldsymbol{U} \setminus \boldsymbol{P}_s'$, go to step 6.
   (d) use both guards g_s and g_s', update uncovered free space $\boldsymbol{U} \leftarrow \boldsymbol{U} \setminus (\boldsymbol{P}_s \cup \boldsymbol{P}_s')$, go to step 6.
6. Remove the processed pair G_s from the set of pairs $\boldsymbol{G}_P \leftarrow \boldsymbol{G}_P \setminus \{G_s\}$.
7. Repeat step 5 if $\boldsymbol{G}_P$ is not empty.

The performance of the BP algorithm mostly depends on the initial boundary selection. The original idea of the BP algorithm expects the boundary at a distance close to the visibility range. If a boundary is created at a very small distance to the polygon border the guards will be placed unnecessarily close to obstacles. On the other side, if a distance is relatively high, the boundary created from the shrunk free space can be degenerated. In a degenerated case a large portion of the free space is a part of the exterior and the heuristic approach is used. A boundary can be determined as the border of the shrunk free space by a distance b. During experimental verification of the proposed algorithm a set of b values have been found for each particular environment and visibility range, see Section 5. An example of the Boundary Placement algorithm performance is shown in Fig. 6.

Complexity of the algorithm is very similar to RDS. Assume that the number of random samples in the first three parts is one and $\mathcal{W}$ is represented by n_v vertices. The first part of the BP algorithm requires computation of a visibility polygon for each new guard, which can be done in $O(n_v \log n_v)$. The second and third parts

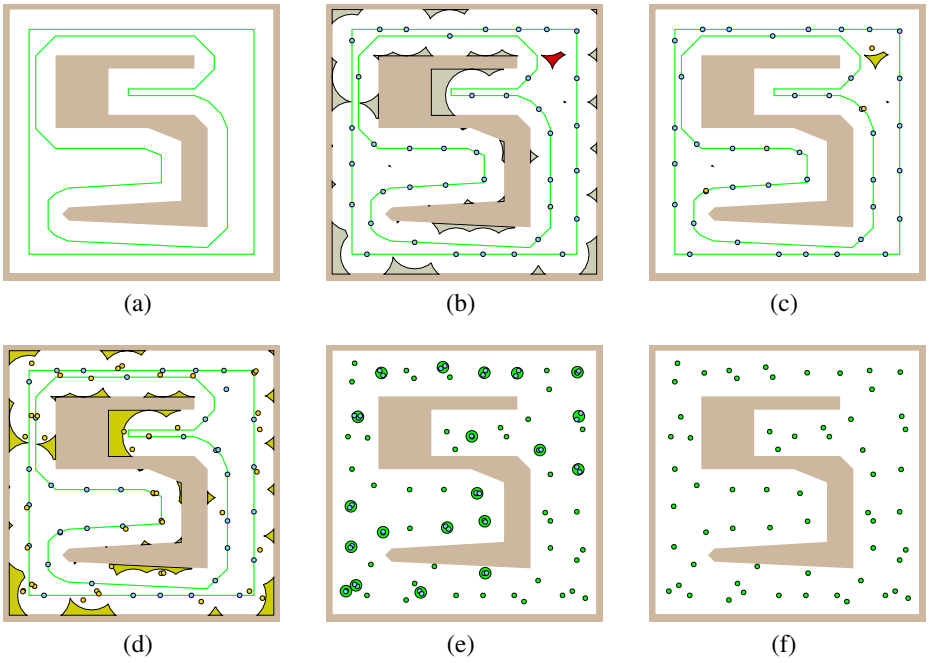


Fig. 6 An example of the BP algorithm performance, the visibility range d is 2 m; **a** polygonal environment and its boundary at distance 1.5 m, **b** boundary coverage and uncovered interior and exterior parts of free space, **c** interior coverage, **d** exterior coverage, **e** guards reduction, **f** the final set of guards

are a little bit complicated as they require computation of the closest guard and determination of the new point to cover the region. The closest visible guard g_c can be found as an intersection of the full visibility region of the point p with the set of guards.[1] The point p belongs to the particular polygon of the polygon collection $\boldsymbol{I}$, resp. $\boldsymbol{E}$, the furthest vertex v of the particular polygon $\boldsymbol{I}_p$, resp. $\boldsymbol{E}_p$, is determined according to its Euclidean distance to g_c. The distance between g_c and v is used to estimate the position of the new guard g_c according to the current visibility range d. Then, a visibility polygon for the new guard candidate is determined. The small region cover strategy is applied for small polygons $\boldsymbol{I}_p$, resp. $\boldsymbol{E}_p$, therefore its number of vertices is expected to be less than n_v and determination of the vertex v is negligible according to computation of the required visibility polygon from p. It means the time complexity can be bounded by $O(n_v \log(n_v))$. The large region cover strategy also requires additional computation of a visibility polygon. At first, a midpoint of the longest circle part is determined in $O(k)$ steps, where k is the number of disk vertices, then the guard candidate is placed at the center of the segment from the random point at the border of the uncovered polygon and the midpoint. Similarly to the RDS, coverage of the new guard is subtracted from the uncovered free space, which increases a number of vertices. The complexity of the first three parts depends on the visibility polygons and can be bounded by $O(n_v n_g \log(n_v n_g))$.

[1] E.g. by Boolean operation on Nef polyhedra representation.

Table 2 Basic properties of used maps

Map name	Dimensions (m × m)	No. holes	No. vertices	Reduced no. vertices
jh	20.5 × 23.1	9	424	212
ta	39.7 × 46.9	2	174	87
pb	133.3 × 105.0	3	192	92

The last optimization procedure requires computation of mutually visible guards, which can be done in $O((n_v + n_g)^2)$. All pairs can be sorted in $O(n_g^2 \log(n_g^2))$, but only $n_g/2$ pairs can be selected at maximum and only $n_g/2$ new guards can be determined, therefore complexity of the determination of visibility regions is not increased. The overall algorithm complexity can be bounded by $O(n_v n_g \log(n_v n_g) + n_v^2 + n_g^2 \log(n_g^2))$.

The boundary, used in the first part of the algorithm, can be obtained as a border of the shrunk free space. The shrunk free space can be found by the Minkowski sum of the free space polygon and a convex disk, therefore the boundary determination can be bounded by $O((n_v k)^2)$, where n_v is the number of polygon vertices and k is the number of disk vertices.

5 Experiments

The performance of the three presented algorithms have been experimentally evaluated within three maps of real environments.[2] The assumed robot has differential nonholonomic drive and free space has been shrunk by a radius of the circumscribed circle around the robot polygonal shape, therefore a robot is modeled as a point robot. All maps[3] have been filtered by the polygon filter technique presented in Section 4.1.1 for the relevance value $r = 5$ cm. Changes of the free space area is less than one percent after filtering. The environment properties are presented in Table 2 and they are visualized in Fig. 7.

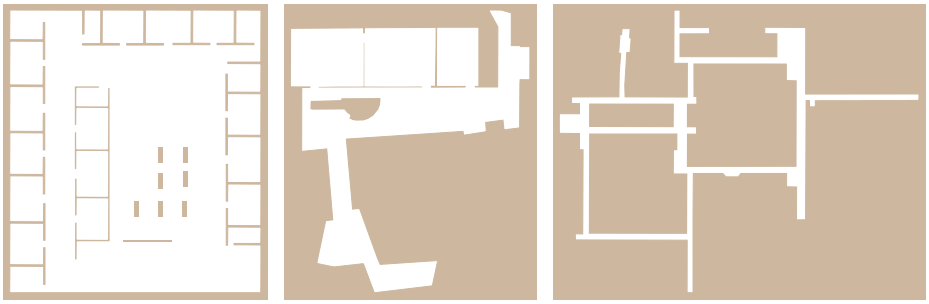
The performance of algorithms is evaluated for the set of visibility ranges {*inf*, 10.0, 5.0, 4.0, 3.0, 2.0, 1.5, 1.0} meters, where *inf* denotes the unrestricted visibility range. The restricted visibility is modeled by a disk with 24 vertices.

Two qualities of the sensor placement solutions are examined, a number of found guards and a length of the inspection path. The path is found as a solution of the related TSP. The TSP is solved exactly by the Concorde solver [26] for all problems, except solutions for the map *pb* and visibility ranges $d \in \{1.0, 1.5\}$ (all algorithms) and the map *ta*, $d = 1.0$ only in the case of the CPP algorithm. In these particular cases a number of guards is too high and the exact solution is too computationally demanding. That is why these problems are solved by the Chained Lin-Kernighan heuristic [27].

The BP and RDS algorithms are randomized, therefore 20 solutions are found by each algorithm for each particular configuration (map and visibility range) and average values are determined. To compute overall algorithm performance ratios of

[2]These environments were used as testing environments for real experiments in the search and rescue missions during solution of the IST-2001-FET project number 38873—PeLoTe Building Presence through Localization for Hybrid Telematic Systems.

[3]The maps are available at http://purl.org/faigl/planning.



(a) map *jh* (b) map *ta* (c) map *pb*

Fig. 7 Testing environments *jh*, *ta* and *pb*

the number of guards and the length of the path are used. Before the algorithms comparison the most suitable settings for each algorithm have been found.

The CPP algorithm does not have any special parameters. The polygon filtering technique is useful, because the reduction of vertices leads to less number of guards. A number of vertices after the polygon filtering is shown in the last column of Table 2.

The RDS algorithm depends on the number of random samples m. An overall results according to reference for $m = 1$ are presented in Table 3. The $m = 25$ provides the best performance while it is less computation intensive than higher values of m.

The BP depends mainly on the boundary distance. For each map and visibility range the boundary distance b has been experimentally found, see Table 4. For small visibility ranges the boundary is in one meter from obstacles for all used maps, while higher visibility ranges require particular value of b for each map to get better results. Presented values do not provide the smallest number of found guards, but they have been selected to have small the total number of b values and the length of inspection paths. The number of random samples has been set to 1 and the heuristic parameters to $\mu_I = \mu_E = 0.66$.

A comparison of the algorithms is made according to reference solution found by the CPP algorithm, it means that for each particular solution a number of guards and a length of the tour are divided by appropriate values of the CPP solution. The ratios are denoted as GR for *Guards Ratio* and LR for *Length Ratio*. The relative

Table 3 RDS performance according to the number of random samples m

m	Guards ratio	Length ratio
1	1.00	1.00
5	0.88	0.95
10	0.87	0.94
25	0.87	0.93
50	0.87	0.93
75	0.88	0.93
100	0.89	0.93

Table 4 Selected values of the boundary parameter b

Map	Boundary distance b (m)	
	Visibility range $d \leq 2$ m	Visibility range $d > 2$ m
jh	1.0	1.5
ta	1.0	3.0
pb	1.0	2.0

values allow presentation of overall results as average values and sample standard deviations σ_{GR}, σ_{LR}. The overall results of the RDS and BP algorithms are presented in Table 5. In Fig. 8 the first two bars denote guards ratios and the next two bars denote length ratios for the RDS and BP algorithms according to the reference solution found by the CPP algorithm. Detail results are dedicated to Appendix A.

The RDS algorithm provides less number of guards than the CPP and also the path is about 10% shorter. Similarly the BP outperforms the CPP. The ratios increase with decreasing visibility range, but paths are more than about twenty percents shorter. For the unrestricted visibility range performances of the BP and RDS algorithms are pretty much similar, however for the visibility range 10 m the BP provides shorter tour about more than 10%. For higher visibility ranges the difference in the numbers of guards is not so significant, but the BP still outperforms the RDS in the length of the inspection tour.

During the evaluation of the BP performance, it has been observed that the guards optimization procedure significantly reduces the number of required guards. The procedure can be used for any guard set, therefore solutions of the CPP and RDS have been optimized. The overall comparison as ratios according to the CPP algorithm is presented in Table 6. The post-processed solutions are denoted as CPP-opt and RDS-opt and the BP algorithm without optimization as BP-notopt. The optimization procedure decreases numbers of guards for the CPP and RDS and tour lengths are also decreased. For the RDS the tour reduction is about three percents. An interesting observation provides results for the BP without optimization. However the number of guards is reduced about 10% the length of the tour is shortened only about one percent after the optimization. This experimental results support the main idea of the BP algorithm to guide the sampling process and to place guards close to each other to reduce the final length of the tour. The inspection path is still shorter

Table 5 Algorithms performance

d (m)	RDS algorithm				BP algorithm			
	GR	LR	σ_{GR}	σ_{LR}	GR	LR	σ_{GR}	σ_{LR}
inf	0.38	0.76	0.05	0.05	0.38	0.77	0.05	0.05
10.0	0.49	0.85	0.08	0.07	0.45	0.74	0.11	0.13
5.0	0.61	0.89	0.13	0.11	0.54	0.75	0.12	0.14
4.0	0.65	0.89	0.15	0.11	0.60	0.76	0.15	0.14
3.0	0.67	0.91	0.09	0.05	0.62	0.78	0.11	0.13
2.0	0.75	0.93	0.04	0.03	0.59	0.82	0.04	0.04
1.5	0.72	0.88	0.03	0.04	0.56	0.75	0.02	0.02
1.0	0.70	0.85	0.01	0.04	0.60	0.74	0.01	0.02

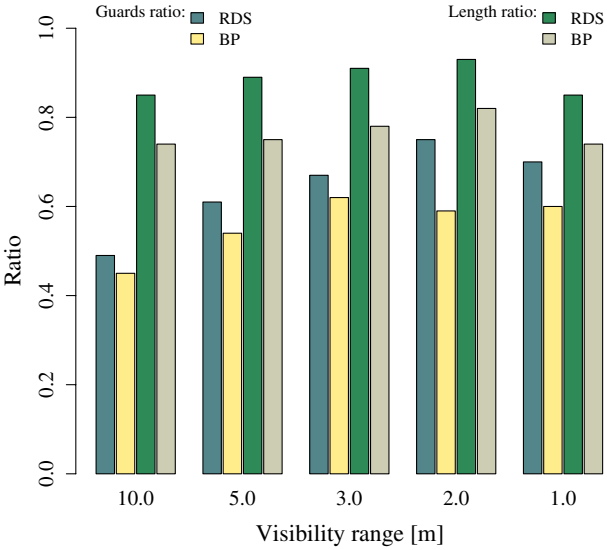


Fig. 8 The algorithms comparison relatively to the CPP algorithm

for solutions found by the BP-notopt, however the number of guards is higher than for the RDS-opt and CPP-opt in several cases.

To demonstrate capabilities of the presented algorithms several examples of found solutions for additional environments are presented in Appendix B.

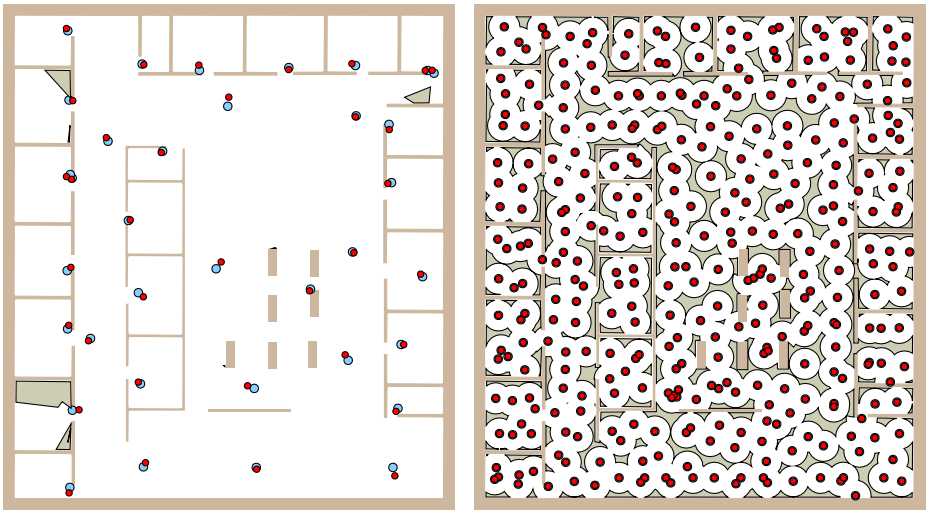
5.1 Effect of Disturbances

A robot performing an inspection does not have precise localization nor sensors are perfect. The imperfect localization leads to miss the exact guards positions. Also particular sensor coverage can be less than the expected one. These disturbances can lead to imperfect coverage. It is clear that a solution with less number of sensing locations for particular visibility range will be more sensitive to sensing locations disturbances or changes in the sensor visibility range. To examine the sensitivity of solutions found by the particular algorithm two experiments have been performed as follows.

The effect of the localization error has been examined by a random perturbation of each sensing location of found solutions presented in the previous section. The perturbation has been performed in two steps to reflect position estimation accuracy and repeatability. At first, the position of the guard has been modified according to

Table 6 Overall algorithms performance

Method	GR	LR	σ_{LR}	σ_{GR}
CPP	1.00	1.00	0.00	0.00
RDS	0.62	0.87	0.08	0.15
BP	0.54	0.76	0.10	0.12
CPP-opt	0.74	0.91	0.04	0.03
RDS-opt	0.53	0.84	0.08	0.12
BP-notopt	0.64	0.77	0.11	0.18



(a) 98.9% coverage caused by localization imperfections

(b) 87.8% coverage caused by 25% reduced visibility range

Fig. 9 Examples of disturbances to coverage, small disks represent guards, the guards before disturbances are in *blue*

two dimensional Gaussian distribution $\mathcal{N}(0, \sigma_a^2)$, where σ_a represents accuracy of the localization. Such positions have been then modified according to $\mathcal{N}(0, \sigma_r^2)$, where σ_r represents repeatability of the localization method. Computed coverage from these new positions has been expressed as the percentage of seen area of free space, denoted as $\%Coverage$. An example of imperfect coverage is illustrated in Fig. 9a. To get statistically meaningful results each found solution has been randomized 20 times and average values of the coverage have been computed.[4] Regarding to presented results of localization methods in [28], two sets of values of localization accuracy and repeatability have been considered for indoor environments. The methods are based on visual landmarks and the accuracy is lower than for a method based on laser range finder presented in [29], thus the selected values of coverage change represent the upper bounds. The total number of performed experiments for the position perturbations is 39,360. The overall results are presented in Table 7. It is shown that the guard optimization procedure increases sensitivity of the solution to the position perturbations. The coverage error is not significant for higher visibility ranges, even for the $d = 1$ m the error is less than 1.5% for more accurate localization method ($\sigma_a = 14$ cm). For $\sigma_a = 30$ cm the proposed BP algorithm without the optimization procedure provides competitive results to the RDS algorithm.

Sensitivity of the algorithm to disturbances of the visibility range d has been examined by a systematic decreasing of the sensor visibility range. A coverage of a solution for the particular value of d has been computed for the decreased d and expressed as the percentage of seen area similarly to the previous examination. An

[4]That means 480 sets of sensing localizations for the RDS, resp. BP, algorithm for each map and visibility range.

Table 7 An effect of position disturbances

Method	$d = 1.0$ m			$d = 5.0$ m			$d = 10.0$ m		
	$\%Coverage$	LR	GR	$\%Coverage$	LR	GR	$\%Coverage$	LR	GR
Localization accuracy $\sigma_a = 14$ cm, localization repeatability $\sigma_r = 8$ cm									
CPP	99.95	0.97	1.00	100.00	1.00	1.00	100.00	1.00	1.00
RDS	99.42	0.83	0.70	99.93	0.88	0.61	99.93	0.85	0.49
BP	98.72	0.73	0.60	99.52	0.75	0.54	99.64	0.74	0.45
CPP-opt	99.79	0.87	0.75	99.99	0.91	0.72	100.00	0.91	0.72
RDS-opt	98.84	0.79	0.57	99.91	0.86	0.54	99.91	0.83	0.44
BP-notopt	99.24	0.78	0.77	99.58	0.76	0.64	99.69	0.74	0.51
Localization accuracy $\sigma_a = 30$ cm, localization repeatability $\sigma_r = 18$ cm									
CPP	99.28	0.92	1.00	99.99	1.00	1.00	100.00	1.00	1.00
RDS	97.05	0.81	0.70	99.50	0.88	0.61	99.46	0.84	0.49
BP	95.45	0.73	0.60	98.56	0.75	0.54	98.78	0.74	0.45
CPP-opt	98.05	0.83	0.75	99.86	0.91	0.72	99.95	0.91	0.72
RDS-opt	94.94	0.76	0.57	99.26	0.86	0.54	99.33	0.83	0.44
BP-notopt	97.44	0.79	0.77	98.69	0.76	0.64	98.87	0.75	0.51

example of such decreased coverage is shown in Fig. 9b. Results for the selected visibility ranges and particular reductions are presented in Table 8. A shortened visibility range about 10 cm (or less), which is higher disturbance than an accuracy of available laser range finders, does not really affect the quality of coverage. Moreover, the accuracy can be considered in the requested visibility range and a solution of the sensor placement problem can be found for such reduced visibility range.

The presented results confirm the expectation that a solution with smaller number of guards is more sensitive to disturbances. However, the cost of the solution (the number of guards and the length of the inspection path) have to be taken into account in selection of the most suitable algorithm for particular robot configuration. In addition, the imperfections can be considered in the visibility range, thus a solution can be found for particularly smaller d. Hence, the disturbances are not a real issue and overall comparison of the algorithms, presented in Table 6, provides an overview of expected quality of a found solution by the particular algorithm.

5.2 Required Computational Time

Beside the quality of solutions the required computational time is important for the real applicability of the algorithms in the mobile robotics. The theoretical bounds provide estimation of the complexity, however the real computational requirements

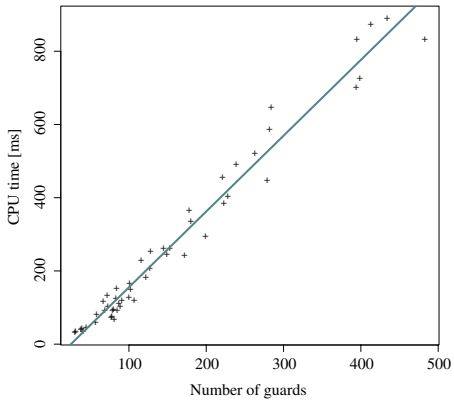
Table 8 An effect of visibility range disturbances, values of $\%Coverage$

Method	$d = 1.0$ m			$d = 5.0$ m			$d = 10.0$ m		
	$d_{95\%}$	$d_{90\%}$	$d_{75\%}$	$d_{98\%}$	$d_{90\%}$	$d_{80\%}$	$d_{99\%}$	$d_{90\%}$	$d_{80\%}$
CPP	100.00	99.99	99.40	100.00	99.95	99.73	100.00	100.00	99.99
RDS	99.90	99.16	89.81	100.00	99.62	97.48	100.00	99.60	98.19
BP	99.15	97.37	85.92	99.97	99.19	96.51	99.99	99.29	97.48
CPP-opt	99.94	99.78	96.90	100.00	99.82	99.19	100.00	99.98	99.74
RDS-opt	99.40	97.84	85.17	99.98	99.36	96.50	100.00	99.47	97.70
BP-notopt	99.58	98.40	89.39	99.99	99.38	97.15	100.00	99.42	97.85

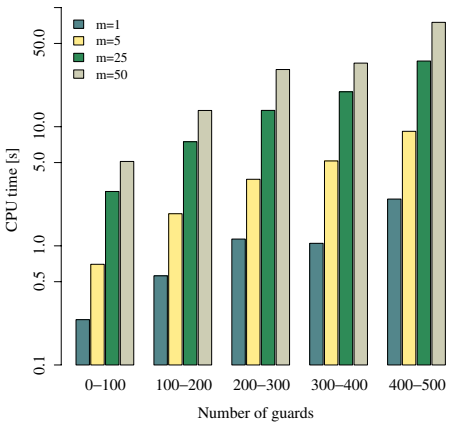
mainly depend on used geometric models. To compare the real requirements all experiments have been performed in the same computational environment with the Athlon X2@2 GHz CPU, 1 GB RAM running FreeBSD 7.1 with only one CPU core dedicated to computations.

The CPP algorithm has been implemented in C++ and CGAL library version 3.3.1 [30]. The used geometric kernel has been *Exact_predicates_exact_-constructions_kernel_with_sqrt*, which provides sufficient precision and good real-time performance. The program has been compiled by the G++ 4.2 with the −O2 optimization flag. The required computational time linearly increases with a number of found guards, see Fig. 10a. For problems with less than 500 guards the computation takes hundreds of milliseconds. The largest solved problem with 1,852 guards has been solved in 3.3 s.

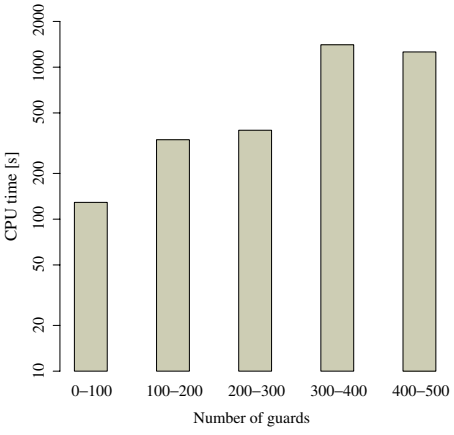
The RDS algorithm has been implemented in Java with the geometric library JTS [31] and the diablo-jdk1.6.0 [32] java runtime machine. The real-time per-



(a) CPP



(b) RDS



(c) BP

Fig. 10 The algorithms required computational time of the examined problems

formance is presented in Fig. 10b as a histogram of required computational time according to the number of found guards, average values are visualized. For 25 random samples a solution is found in several seconds, while for $m = 1$ it is found in hundreds of milliseconds (solutions with less than 300 guards). The used geometric representation is double precision floating point (IEEE 754), which provides only limited precision of geometric operations. During the experiments, the total number of failures (mainly in the intersection operation) has been 3,867, which represents 0.560% of the total number of found guards or 0.015% of the total number of performed guards selections. The most problematic map is *jh* with 2,120 failures. For the map *ta* 1,378 failures has been recorded. The advantage of the randomized incremental algorithm is safe recovery from such fails. If an operation cannot be performed, a new random point is generated and the algorithm can continue.

An implementation of the BP algorithm is based on the Planar Nef polyhedra representation, from the CGAL library version 3.1, which does not support simple double precision, therefore the precise `CGAL::Gmpz` kernel has been used. This representation allows Boolean operations on polygons, which can be partially closed, i.e. an edge of the polygon border can be opened while the next edge can be closed. However the theoretical bound for the first three parts of the algorithm is less than for the RDS (only one random sample is used), the real computational time is heavily affected by the used geometric kernel. During subtraction of polygons a lot of Steiner points are created and due to the used arbitrary precision representation the polygonal operations are incredibly slow and the largest problem with more than one thousands guards has been solved in two hours. Small problems with less than hundred guards are solved in tens of seconds. Running times are shown in Fig. 10c, solutions with 300–400 guards are mostly found for small visibility ranges, therefore more Steiner points are created. The post-processing optimization takes about twenty percent of the total required computational time. It should be noted that all visibility polygons of found guards are re-computed in the optimization procedure.

6 Conclusion

The new randomized sensor placement algorithm called the Boundary Placement has been presented. The proposed randomized sampling procedure explicitly uses knowledge about the environment shape to guide the randomization process. Performance of the proposed algorithm has been experimentally compared with two state-of-the-art algorithms. The algorithms have been evaluated according to the number of found guards and the length of the inspection path. The proposed BP algorithm outperforms both algorithms and provides solution with less number of guards and about 10% shorter path. The experimental results show that length of the path can be decreased by more sophisticated sampling even in cases that the decoupled problems are solved independently.

However only restricted visibility range has been considered in the experiments, the proposed BP algorithm can be easily extented to consider the incident constraints like the RDS. Applications of the RDS algorithm in the related VPP have been reported in the literature, hence the applicability of the BP algorithm to all related problems is expected, as it provides better solutions than the RDS.

The randomized approach provides solution of the sensor placement problem in less than few seconds, therefore it is suitable for real applications. Even though the real computational requirements of the proposed BP algorithm are higher than the RDS, due to used precise geometry representation, the complexity of the BP algorithm is lower then the RDS, because only one random sample and one point from the heuristic is used for each found guard. The post-processing optimization increases the complexity, but overall performance of the BP without optimization is similar to the RDS regarding to the number of guards and it provides shorter inspection paths. Also the optimization procedure is useful for the RDS as well as for the CPP algorithm.

In the majority of the examined problems, more than 80% guards are placed on the boundary in the initial (the boundary cover part), which is very efficient, just one random sample is used. The boundary is created from the shrunk free space, and the performance of the algorithm can be probably increased by structures like the Visibility-Voronoi diagram [33] or Saturated Generalized Voronoi Diagram (SGVD) [34].

Appendix A: Experimental Results

Particular results for each examined environment are presented in Tables 9, 10 and 11.

Table 9 Algorithms performance for the map *jh*

d (m)	No. guards			Length (m)		
	CPP	RDS	BP	CPP	RDS	BP
inf	94	33	31	208	154	149
10.0	95	37	32	207	158	118
5.0	101	44	37	216	160	124
4.0	106	47	41	220	164	125
3.0	115	63	53	226	190	138
2.0	175	126	94	282	269	217
1.5	282	200	160	350	321	258
1.0	552	388	340	471	414	359

Table 10 Algorithms performance for the map *ta*

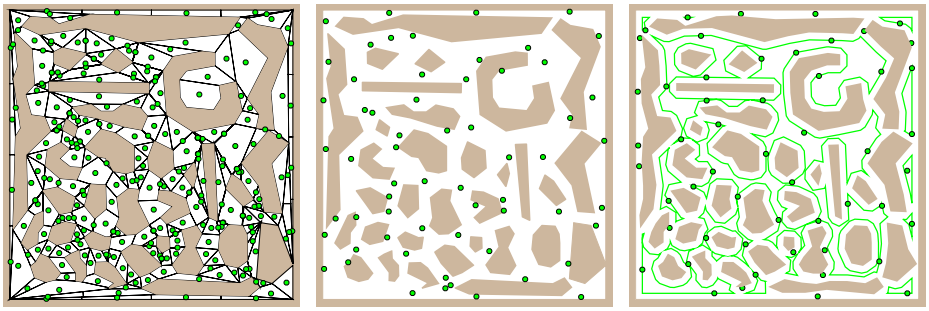
d (m)	No. guards			Length (m)		
	CPP	RDS	BP	CPP	RDS	BP
inf	34	14	13	204	167	167
10.0	35	18	15	203	176	159
5.0	58	41	34	254	239	197
4.0	72	54	52	272	260	224
3.0	118	83	83	315	292	266
2.0	231	170	139	408	366	330
1.5	405	280	231	522	436	386
1.0	865	598	526	746	595	528

Table 11 Algorithms performance for the map *pb*

d (m)	No. guards			Length (m)		
	CPP	RDS	BP	CPP	RDS	BP
inf	45	16	20	533	389	407
10.0	73	41	44	613	567	533
5.0	131	90	84	683	667	619
4.0	160	120	109	720	697	650
3.0	235	178	160	775	741	696
2.0	434	349	271	902	851	787
1.5	821	616	439	1,116	1,001	858
1.0	1,809	1,289	1,067	1,565	1,346	1,151

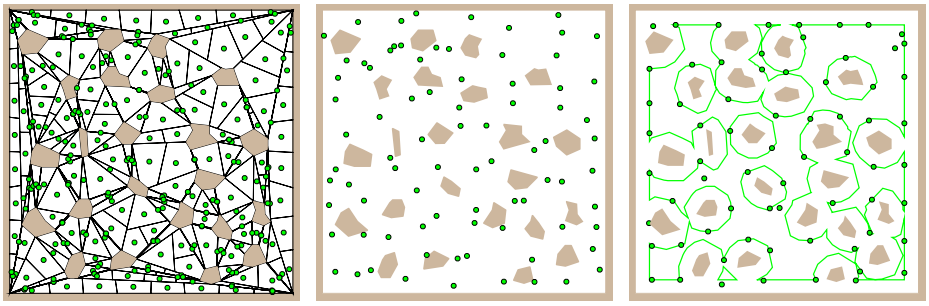
Appendix B: Example of Solutions

Solutions for environments denoted as *dense*, *potholes*, *warehouse* and *h2* are presented in following figures (Figs. 11, 12, 13 and 14). All maps are processed by the polygon filter with the relevance value 5 cm and the RDS has been run with $m = 25$.



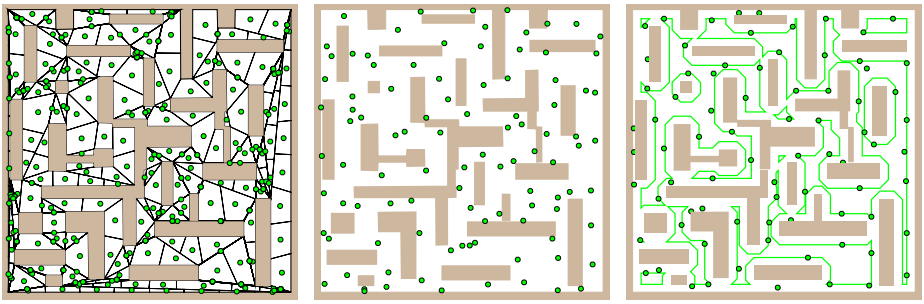
(a) CPP, 285 guards, length 270 m (b) RDS, 61 guards, length 181 m (c) BP, 53 guards, length 180 m

Fig. 11 Example of solution for the map *dense*, 21 × 21 m, visibility range 4 m and boundary at distance 0.5 m



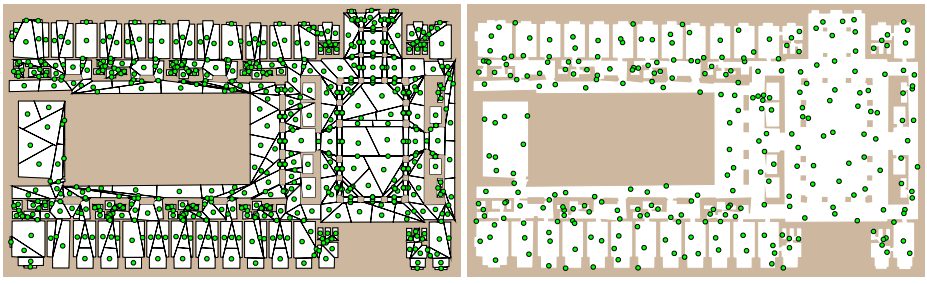
(a) CPP, 306 guards, length 234 m (b) RDS, 81guards, length 159 m (c) BP, 68 guards, length 155 m

Fig. 12 Example of solution for the map *potholes*, 20 × 21 m, visibility range 2 m and boundary at distance 1 m

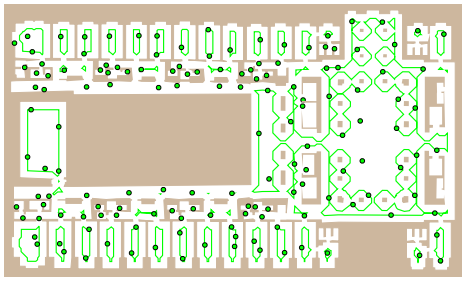


(a) CPP, 361 guards, length 496 m (b) RDS, 99 guards, length 383 m (c) BP, 78 guards, length 363 m

Fig. 13 Example of solution for the map *warehouse*, 40 × 40 m, visibility range 4 m and boundary at distance 1.2 m



(a) CPP, 361 guards, length 1353 m (b) RDS, 99 guards, length 1132 m



(c) BP, 78 guards, length 887 m

Fig. 14 Example of solution for the map *h2* representing real building with dimensions approximately 70 × 40 m, visibility range 5 m and boundary at distance 1.5 m

References

1. Chin, W.-P., Ntafos, S.: Optimum watchman routes. In: SCG '86: Proceedings of the Second Annual Symposium on Computational Geometry, pp. 24–33, Yorktown Heights, New York. ACM (1986)
2. Packer, E.: Robust geometric computing and optimal visibility coverage. PhD thesis, Stony Brook University, New York (2008)

3. Danner, T., Kavraki, L.E.: Randomized planning for short inspection paths. In: Proceedings of The IEEE International Conference on Robotics and Automation (ICRA), pp. 971–976, San Francisco, CA. IEEE (2000)
4. Culberson, J.C., Reckhow, R.A.: Covering polygons is hard. J. Algorithms **17**(1), 2–44 (1994)
5. Tan, X., Hirata, T.: Finding shortest safari routes in simple polygons. Inf. Process. Lett. **87**(4), 179–186 (2003)
6. Ntafos, S.C.: Watchman routes under limited visibility. Comput. Geom. **1**, 149–170 (1992)
7. Li, F., Klette, R.: An approximate algorithm for solving the watchman route problem. In: RobVis, pp. 189–206 (2008)
8. Goodman, J.E., O'Rourke, J. (eds.): Handbook of Discrete and Computational Geometry. CRC Press, Boca Raton (2004)
9. González-Baños, H.H., Hsu, D., Latombe, J.-C.: Motion planning: recent developments. In: Ge, S.S., Lewis, F.L. (eds.) Autonomous Mobile Robots: Sensing, Control, Decision-Making and Applications, Chapter 10. CRC (2006)
10. Wang, P.: View planning with combined view and travel cost. PhD thesis, Simon Fraser University (2007)
11. González-Baños, H.H., Latombe, J.-C.: Planning robot motions for range-image acquisition and automatic 3d model construction. In: AAAI Fall Symposium (1998)
12. Hörster, E., Lienhart, R.: On the optimal placement of multiple visual sensors. In: VSSN '06: Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks, pp. 111–120, New York, NY. ACM (2006)
13. Erdem, U.M., Sclaroff, S.: Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. Comput. Vis. Image Underst. **103**(3), 156–169 (2006)
14. Osais, Y.E., St-Hilaire, M., Yu, F.R.: Directional sensor placement with optimal sensing range, field of view and orientation. Mob. Netw. Appl. **15**(2), 216–225 (2008)
15. Kazazakis, G.D., Argyros, A.A.: Fast positioning of limited visibility guards for the inspection of 2d workspaces. In: Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS 2002), Lausanne (2002)
16. Scott, W.R., Roth, G., Rivest, J.-F.: View planning for automated three-dimensional object reconstruction and inspection. ACM Comput. Surv. **35**(1), 64–96 (2003)
17. Blaer, P.S., Allen, P.K.: Data acquisition and view planning for 3-d modeling tasks. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007), pp. 417–422, 29 October–2 November 2007
18. Nüchter, A., Surmann, H., Hertzberg, J.: Planning robot motion for 3d digitalization of indoor environments. In: Proceedings of the 11th International Conference on Advanced Robotics (ICAR), pp. 222–227 (2003)
19. González-Baños, H.H., Latombe, J.-C.: Navigation strategies for exploring indoor environments. Int. J. Rob. Res. **21**(10–11), 829–848 (2002)
20. González-Banos, H.H.: A randomized art-gallery algorithm for sensor placement. In: SCG '01: Proceedings of the Seventeenth Annual Symposium on Computational Geometry, pp. 232–240. ACM, New York (2001)
21. Lavalle, S.M.: Planning Algorithms. Cambridge University Press (2006)
22. Overmars, M.H., Welzl, E.: New methods for computing visibility graphs. In: SCG '88: Proceedings of the Fourth Annual Symposium on Computational Geometry, pp. 164–171. ACM, New York (1988)
23. Seidel, R.: A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. Comput. Geom. Theory Appl. **1**(1), 51–64 (1991)
24. Latecki, L.J., Lakämper, R.: Convexity rule for shape decomposition based on discrete contour evolution. Comput. Vis. Image Underst. **73**(3), 441–454 (1999)
25. Wolter, D., Richter, K.-F.: Schematized aspect maps for robot guidance. In: Proceedings of the ECAI Workshop Cognitive Robotics (CogRob) (2004)
26. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: CONCORDE TSP Solver. http://www.tsp.gatech.edu/concorde.html (2003). Accessed 23 July 2010
27. Applegate, D., Cook, W., Rohe, A.: Chained lin-kernighan for large traveling salesman problems. INFORMS J. Comput. **15**(1), 82–92 (2003)
28. Chen, Z., Birchfield, S.T.: Qualitative vision-based path following. IEEE Transactions on Robotics **25**(3), 749–754 (2009)
29. Sohn, H.J., Kim, B.K.: Vecslam: an efficient vector-based slam algorithm for indoor environments. J. Intell. Robot. Syst. **56**(3), 301–318 (2009)

30. CGAL—Computational Geometry Algorithms Library. http://www.cgal.org (2004). Accessed 23 July 2010
31. JTS Topology Suite. http://www.vividsolutions.com/jts/jtshome.htm. Version 1.5 (2004). Accessed 23 July 2010
32. Diablo Caffe JDK 1.6.0-7. http://www.freebsdfoundation.org/downloads/java.shtml (2009). Accessed 23 July 2010
33. Wein, R., van den Berg, J.P., Halperin, D.: The visibility–voronoi complex and its applications. In: SCG '05: Proceedings of the Twenty-First Annual Symposium on Computational Geometry, pp. 63–72. ACM, New York (2005)
34. Huang, W.H., Beevers, K.R.: Complete Topological Mapping with Sparse Sensing. Technical Report 6, Rensselaer Polytechnic Institute Department of Computer Science (2005)