Fast Positioning of Limited-Visibility Guards for the Inspection of 2D Workspaces

Giorgos D. Kazazakis, Antonis A. Argyros

Institute of Computer Science,
Foundation for Research and Technology-Hellas
Heraklion, Crete, Greece
e-mail: {kazaz, argyros}@ics.forth.gr

Abstract

This paper presents a novel method for deciding the locations of "guards" required to visually inspect a given 2D workspace. The decided guard positions can then be used as control points in the path of a mobile robot that autonomously inspects a workspace. It is assumed that each of the guards (or the mobile robot that visits the guard positions in some order) is equipped with a panoramic camera of 360 degrees field of view. However, the camera has limited visibility, in the sense that it can observe with sufficient detail objects that are not further than a predefined visibility range. The method seeks to efficiently produce solutions that contain the smaller possible number of guards. Experimental results demonstrate that the proposed method is computationally efficient and that, although suboptimal, decides a small number of guards.

1 Introduction

This paper deals with the problem of 2D area inspection which belongs to the category of "art gallery problems". The original art gallery problem, originally stated by Klee (see [1]) and studied by Chvatal [2], is to find the smallest number of guards necessary to lookout every point in the interior of an n-wall art gallery room. The gallery room is modelled as a simple, not necessarily convex polygon consisting of n vertices. Covering the art gallery by n_g guards translates to finding n_g points in the interior of the polygon such that every point on the polygon is visible by at least one guard. A point is considered visible by a guard if the straight line connecting them does not intersect the polygon. This problem is also related to the "watchman route" problem [3], where the goal is to compute a path in a given workspace such that every point of the workspace will become visible from at least one point on the path. The above problem definitions deal with unconstrained guard visibility. Interesting variants can be formulated by constraining the visual capabilities of the guards. Example constraints, which arise from practical considerations, include limitations in the visual field of the guards and/or on their visibility range. For example, conventional cameras have a limited field of view and cannot capture features of the environment with sufficient quality, if these features lie far away.

There are several surveys on solutions for the art gallery problems [4,5]. In [6], an overview of what is known about the computational complexity of several art gallery problems is provided. It is has been proven that providing an optimal solution to the inspection problem for simple polygons is an NP-hard problem, either for polygons without holes [7] or for polygons with holes [8]. Therefore, to practically deal with the problem, someone has to opt for sub-optimal solutions. Fisk [9] found a fast guard placement algorithm in linear time, using Chazelles algorithm that triangulates polygons in linear time [10]. However, this solution works for polygons without holes, and requires that the guards have unconstrained visibility. Bjorling-Sachs and Souvaine [11] gave an $O(n^2)$ time complexity algorithm to find the position of the guards in a polygon with holes. Still this solution works for unconstrained guard visibility. Solutions for the constrained visibility variant of the inspection problem are based on randomized approaches, such as the algorithms proposed by Gonzales-Banos, Latombe [12], and Danner and Kavraki [13]. These randomized approaches consider a constraint on the size of the visual field of guards. The two algorithms described in [12] have complexity bounded by $O(nm^2)$ and $O(n_g n log(n_g n))$ respectively, with n being the number of edges of the workspace, m the number of random samples drawn to inspect it, and n_g the number of guards that the algorithm defines for the specific workspace. As expected, n_g and m are related to the complexity of

the workspace.

In our formulation of the problem, we consider guards equipped with panoramic cameras with a field of view of 360 degrees. Such cameras are now readily available in the market and widely used in a number of robotic applications [14, 15]. The employment of such cameras makes unnecessary the consideration of constraints on the field of view of the guards. Even with conventional cameras, such constraints are less important because a rotating camera can sweep the full 360 degrees field of view and produce a panoramic view of the environment at a certain location. However, most of the available panoramic cameras provide images of low resolution because the full visual field is squeezed on an conventional CCD array. Thus, to solve the inspection problem with sufficient quality of observation, it is necessary to impose a constraint on the visibility range of the guards. The actual value of the visibility range depends on the sensor employed and on the characteristics of the target application.

There are many applications that could benefit from an efficient solution of the above formulation of the inspection problem. These include inspection and surveillance tasks performed by a moving robot equipped with a panoramic camera. The solution of the inspection problem can derive a minimal number of points that the robot should visit to guarantee full inspection of the workspace. The fast computation of inspection points is of particular importance in time-critical situations. Other interesting applications are the automatic computation of architectural walk-throughs, virtual reality exploration systems, 3D reconstruction of a workspace etc.

The proposed method uses an algorithm [16] that decomposes the initial, simple polygon with holes that models the workspace into a number of convex polygons. Then, a divide and conquer strategy is applied, to successively divide each of the resulting convex polygons into smaller polygons that can be locally inspected with only one guard. The algorithm can handle both versions of the inspection problem (i.e. inspecting the workspace borders or the whole area of the workspace) in a unified manner. However, the computational time required by the algorithm in the first case, is substantially smaller.

The rest of the paper is organized as follows. The proposed method is described in detail in section 2. The computational complexity of the method is analysed in section 3. Experimental results are presented in section 4. Finally, section 5 summarizes the paper.

2 Method description

The proposed algorithm operates on a 2D, simple, non-convex polygon with holes that models the workspace to be inspected. The holes of the polygon correspond to obstacles or internal structures in the environment. The algorithm solves the inspection problem under constrained visibility for the interior borders of every such polygon. If an exterior inspection is desired, then a rectangle enclosing the workspace defines a new virtual border, and the original polygon appears as a hole of the new workspace. The internal inspection of the new polygon will yield the desired result.

The algorithm first decomposes the original non-convex polygon into a collection of convex polygons [16]. This decomposition is advantageous because it simplifies the definition of guards. More specifically, deciding whether a panoramic guard with limited visibility range can inspect a convex polygon is reduced to measuring the distance of the guard to the vertices of the polygon and thus, the employment of more complex sweep algorithms is avoided.

In order to cope with the visibility range constraint, we apply a successive division of every convex polygon into convex sub-polygons, until each of them can be inspected locally by one guard. In the remainder of this section, this algorithmic procedure is described in more detail.

2.1 Polygon division

In order to determine the set of guards that are sufficient for the inspection of a convex polygon given limited visibility range β, the following division procedure is employed. First, a potential guard (observation point - OP) inside the polygon is computed, using a procedure that is described later in this section. Then, the polygon vertex (MDV) with the maximum distance α from the selected OP is determined. Point OP is considered as a guard if it is capable of inspecting the polygon, that is, if and only if α is smaller compared to the visibility range β. In the opposite case, the line L that is perpendicular to the line segment that connects OP and the MDV and passes through OP is computed. This line, divides the convex polygon into two convex sub-polygons. Then, the same procedure is recursively applied to each of the derived convex sub-polygons. The key aspects of this algorithmic procedure is the selection of observation points (OPs) as well as the method for decomposing a convex polygon into convex sub-polygons, which are further discussed in the sequel.

Selection of observation points (OPs). The optimum OP is the one that minimizes the maximum distance from the polygon's vertices. Equivalently,

the optimum OP is the center of the minimum-radius circle that contains the polygon. Unfortunately, the optimum OP may be a point on the edges of the polygon which is obviously a poor location for a guard. Moreover, the selection of OPs should be a computationally cheap process because it is encountered very often. An alternative OP is the point defined by the mean of the coordinates of the polygon vertices (MC). The disadvantage of MC is that it differs substantially from the optimum OP, in cases of polygons with non-uniformly distributed vertices. To alleviate this problem, instead of using MC, we use the point WS defines as:

$$WS(x,y) = \sum_{i=1}^{A} \|E_i\| M_i(x,y) / \sum_{i=1}^{A} \|E_i\|, \quad (1)$$

where $M_i(x,y)$ is the coordinates of the midpoint of the ith edge E_i of the polygon ($1 \leq i \leq A$), and $\|E_i\|$ is the length of edge E_i. The idea behind this selection is to bias OP towards long polygon edges. Extensive experiments have shown that OPs defined as in eq.(1) perform much better than MC (see also Fig. 1).
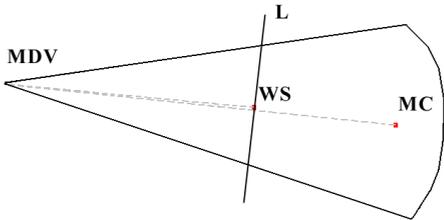

Figure 1: *Definition of OPs: MC vs WS (as defined in eq.(1)). A smaller visibility range is required for WS to inspect the polygon, compared to MC. This is due to the fact that in this polygon, vertices have a non-uniform spatial distribution.*

The choice of polygon division line. The choice of the line that divides a given polygon (that cannot be inspected locally) into two sub-polygons, is also an important factor for the success of the algorithm. The goal of polygon division is to come-up with sub-polygons that can locally be inspected from their observation points. In order to achieve this goal, it is required that polygon division produces sub-polygons where the distance α (the distance of OP from MDV) is decreased as much as possible in successive polygon generations. As already stated earlier in this section, the selected divisor line L is the line that is perpendicular to the line defined by OP (defined as WS) and MDV and passes through OP (see also Fig. 1). This definition of line L aims at minimizing the distance between the OP and the

MDV in the resulting polygons. As an example, in the case of a very elongated convex polygon, this decomposition will reduce this distance by a factor of almost two. At the same time, this choice assures the termination of the algorithm since the maximum distance between OPs and MDVs in all generations of polygon divisions is guaranteed to be a monotonically decreasing function.

2.2 Internal/external polygons

The decomposition of the initial non-convex polygonal workspace into convex polygons and the recursive division of convex polygons into convex sub-polygons, produces edges and vertices that do not necessarily belong to the initial edges of the workspace. We differentiate between *external* and *internal* vertices, edges and polygons. External vertices are vertices that belong to the edges of the workspace borders. External edges are the edges or parts of the edges of the workspace borders. External polygons are polygons that contain at least one external vertex or at least one external edge. Internal vertices are vertices in the inner part of the workspace but not on its edges. An internal edge is defined by two internal vertices. Finally, internal polygons are those defined by internal edges, only. By definition, the borders and vertices of the workspace are external ones. The sub-polygons defined during polygon division can be either external or internal. Note that the intersection of any line with an internal edge gives rise to an internal vertex and the intersection of any line with an external edge gives rise to an external vertex. By exploiting these observations, it can easily be determined whether the sub-polygons that are produced by a division are internal or external. Moreover, at each level of recursion, only external polygons are considered for further subdivision. This is because the edges of internal polygons are virtual ones and do not correspond to parts of the borders of the workspace that need to be inspected. This process speeds up the computations substantially, because large free-space areas will be quickly excluded from the search space of possible guards. Note, however, that for the version of the problem that requires inspection of the full area of the workspace, both internal and external polygons should be inspected.

Figure 2 presents an example of polygon division. Thick black lines show the contour of the polygon, while the rest of the lines are the results of polygon division. The numbers close to the division lines, correspond to the generations of polygon divisions. In this example, polygon division resulted in six external polygons, two internal ones and six guards. The horizontal line at the bottom right of the figure illustrates the visibility range of each guard.
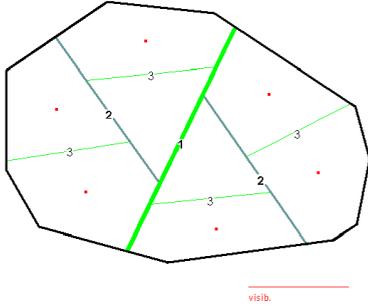
Figure 2: *Example of polygon division (inspection of the borders of the workspace).*

3 Complexity analysis

For decomposing the initial non-convex-polygon with holes to a set of convex polygons, we use an implementation of a fast randomized triangulation algorithm with complexity $O(Nlog^*N + clogN)$, where N is the number of polygon vertices of the original workspace and c is the number of connected components representing the polygon [16]. Then, we use an algorithm to remove unnecessary triangle lines to form convex polygonal areas, in time O(N). In the case of a workspace that does not contain holes, Chazelles [10], [17] triangulation algorithm can be used, with $O(N)$ complexity. The following analysis concerns the computational complexity of a convex polygon division (see section 2.1). The goal of this analysis is to measure the total number of polygon vertices that the algorithm needs to process until it converges to the final solution. This is a direct indication of the computational complexity of the method, since all further operations are linear to the number of vertices encountered.

Convex polygon division is applied to each convex polygon that is derived by convex decomposition. When dividing a polygon, the division line has two intersections with the polygon's edges. Let's suppose that the initial polygon has N vertices and that the two derived sub-polygons (second generation) have N_1 and N_2 vertices, respectively. Then, $N_1 + N_2 = N + 4$, because in the worst (but also most often) case, two new vertices will be introduced in each of the new sub-polygons. In each generation m of divisions, the polygons are doubled (worst case scenario). Thus the number of vertices at a generation m is $N + \sum_{i=2}^{m} 2^i = N + 2^{m+1} - 4$. It can be shown that at the end (i.e. after k generations) the total number of vertices considered by the algorithm will be:

$$k(N-4) + 2^{k+2} - 4. \tag{2}$$

Let P be the initial convex polygon we want to di-

vide and WS the observation point for P, as defined in eq.(1). We also define $1 < \gamma \leq 2$ to be the decrease rate of α between two successive generations of sub-polygons. That is, $\alpha_m = 1/\gamma\alpha_{m-1}$, or, after k generations $\alpha_k = 1/\gamma^k * \alpha_1$. Assuming that γ is constant in all generations, it turns out that

$$k = \log_\gamma \frac{\alpha}{\beta}. \tag{3}$$

In fact, γ is not constant in all generations, however this coarse approximation is useful for deriving some coarse estimation of the overall complexity of the algorithm. By substituting eq.(3) in eq.(2), we come up with a total number of vertices of

$$(\log_\gamma \frac{\alpha}{\beta})(N-4) + 2^{(\log_\gamma \frac{\alpha}{\beta})+2} - 4. \tag{4}$$

Equation (4) captures the intuition behind the performance of the algorithm. The larger the dimensions of the polygons with respect to the visibility range of the sensor, the larger the quantity $\frac{\alpha}{\beta}$ and the larger the number of the polygons and polygon vertices that should be examined by the algorithm. Moreover, the performance of the algorithm increases as the decrease rate γ of α between generations, increases. Experimental results have demonstrated that for reasonably shaped workspaces, γ maintains an average value of the 1.4, and gets higher values, in cases of elongated convex polygons.

Note, however, that eq.(4) refers to the total number of both internal and external polygons defined by the polygon division algorithm. It is very important that, depending on the shape of the workspace, a large majority of these vertices need not be examined because they belong to internal polygons and are removed from consideration due to the algorithmic step described in section 2.2.

4 Experimental Results

The proposed inspection method has been extensively tested. To facilitate testing, a simulator has been developed. In this simulator the user defines (a) the target workspace by drawing a polygon with holes and (b) the desired visibility range. Then, the proposed algorithm is employed to compute the guards that are necessary for inspecting the defined workspace. Figures 3 and 4 show the results of the proposed method when applied to different workspaces under different visibility constraints. The borders of the workspaces are shown with thick black lines, while thinner lines are the results of polygon divisions. The locations of guards appear as points in these figures. Large empty spaces correspond to regions of internal polygons that the method automatically rejected from further consideration. The visibility range is shown as a horizontal
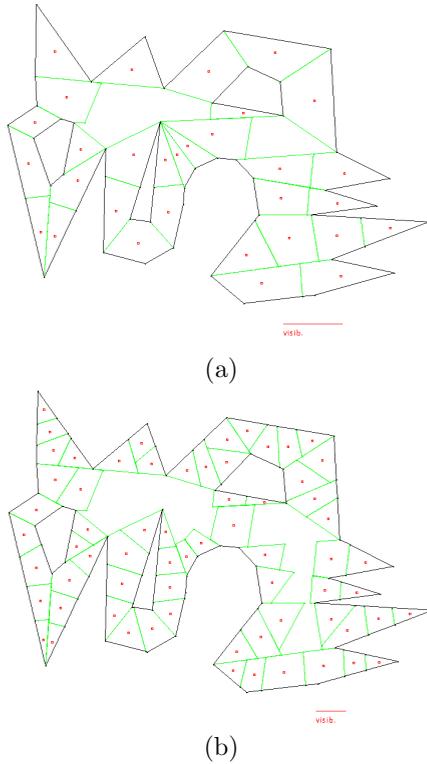
(a)

(b)

Figure 3: *First example of inspection of a given workspace with two different visibility ranges.*
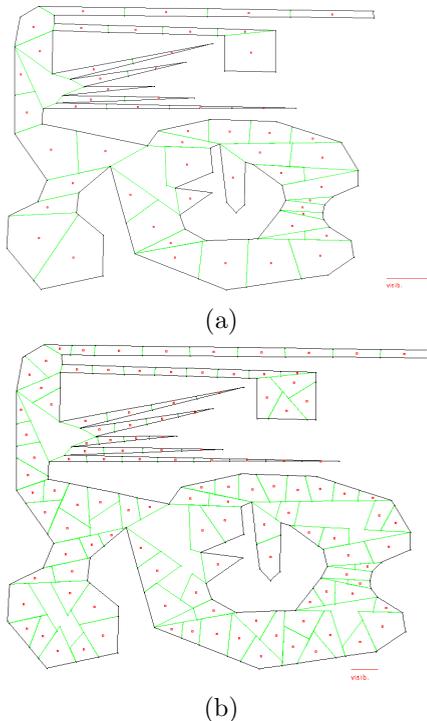

(a)

(b)

Figure 4: *Second example of inspection of a given workspace with two different visibility ranges.*

line at the bottom of each figure. The algorithm appears to perform very satisfactorily, since the number of guards is kept at acceptable levels, and they are placed at locations that are suitable for inspection by a mobile robot. A great advantage of the algorithm is its ability to completely inspect the workspace, even when there are very narrow corridors or isolated spots. A disadvantage of the algorithm appears in the treatment of neighboring convex polygons, with dimensions close to the visibility range. In such cases, the algorithm may assign unnecessary guards. This is because, at least in the current version of the method, polygon division is applied to all different convex sub-polygons, and defines guards that are sufficient to inspect them, without seeking for global improvements between neighboring convex polygons. This effect is apparent in the bottom-right part of Fig. 4(a), (b) (at the narrow passage of the workspace), where more guards are placed compared to what would actually be needed.

In a second set of experiments, the performance of WS as an observation point was tested against the performance of MC. A total of 23 workspaces were tested. The proposed method was employed with the algorithm of section 2.2 disabled, so as to inspect the surface of the workspace in each run. When WS was employed, the method came-up with a total of 2528 guards. In the case of MC the method resulted in 3073 guards. Thus, MC resulted in approximately 21% more guards compared to WS in these experiments. Moreover, for each individual experiment, MC was from 11.6% to 28.8% worse than WS. The above figures show that the selection of observation point is really important to the performance of the algorithm and that WS outperforms MC.

A third series of experiments was conducted to test the number of guards that the proposed method derives as a function of the user defined visibility range. A large convex workspace was employed with a bounding rectangle of dimensions 1081 x 776. Then the proposed inspection method was employed, with varying visibility ranges. The number of guards placed by the algorithm for the case of surface and border inspection are shown in Fig. 5. It can be seen that in both versions of the problem, the number of necessary guards falls substantially as the visibility range of each of the guards increases.

Regarding computational performance, the execution times in the 1st and 2nd series of experiments were less than 10 msecs on a Pentium III processor at 750 MHz running under Linux. In the case of the 3rd series of experiments, and for visibility range equal to 10 (worst case) the execution time for surface inspection was 230 msecs and the execution time for borders inspection was 20 msecs.
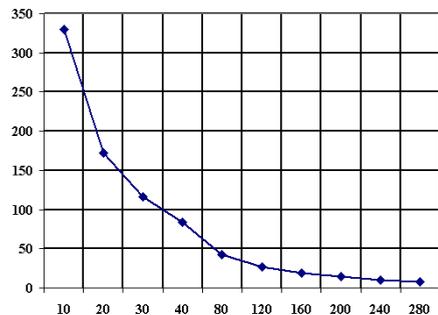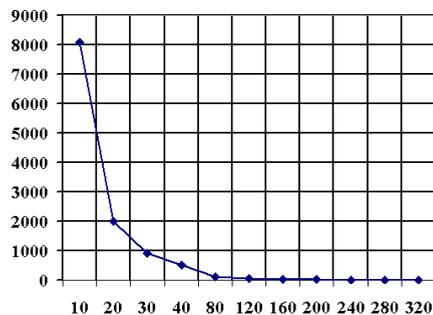
Figure 5: *Number of guards as a function of visibility range, (a) for inspecting the workspace borders, (b) for inspecting the entire workspace surface.*

5   Summary

In this paper, a method for positioning limited visibility guards for the inspection of 2D workspaces has been presented. The derived set of guard locations can be used as control points defining the inspection path of a mobile robot. The number of guards proposed by the algorithm is suboptimal, however the algorithm can efficiently solve the problem for large workspaces, which is important in time-critical applications.

The proposed algorithm first decomposes the workspace to a set of convex polygons and then successively divides them into smaller sub-polygons, until each of them can be inspected by a guard. The time complexity of the algorithm is related to aspects of the shape of the workspace and to the visibility range of the selected panoramic sensor. The proposed method guarantees the complete inspection of the workspace, independently of its configuration and complexity. Another important issue is that the same general algorithmic procedure can be used for solving the inspection problem either for the borders or for the whole surface of the workspace. To solve the second variant, the only requirement is the deactivation of the algorithmic step of section 2.2.

References

[1] R. Honsberger, "Mathematical Gems II", Mathematical Association of America, 1976.

[2] V. Chvatal, "A Combinatorial Theorem in Plane Geometry", Journal of Combinatorial Theory (B) 18 (1975), pp 39-41.

[3] W. Chin and S. Ntafos, "Optimum Watchman Routes", Inf. Processing Letters, 28:39–44, 1988.

[4] J. O'Rourke, "Art Gallery Theorems and Algorithms", Oxford University Press, New York 1987.

[5] T. Shermer, "Recent Results in Art Galleries", Proc. of the IEEE, 1992.

[6] B. Nilsson, "Guarding Art Galleries - Methods for Mobile Guards", doctoral thesis, Department of Computer Science, Lund University, 1994.

[7] J. C. Culberson and R. A. Reckhow, "Covering Polygons is Hard", Proc. 29th Symposium on Foundations of Computer Science, 1988.

[8] J. O'Rourke and K. J. Supowit, "Some NP-hard Polygon Decomposition Problems", IEEE Transactions on Information Theory, Vol IT-29, No.2, 1983.

[9] S. Fisk, "A Short Proof of Chvatal's Watchman Theorem", Journal of Combinatorial Theory, Series B 24 (1978) pg 374.

[10] B. Chazelle, "Triangulating a Simple Polygon in Linear Time", Discrete Comp. Geom. 6 (1991), no. 5, 485-524.

[11] I. Bjorling-Sachs, and D. Souvaine, "An Efficient Algorithm for Guard Placement in Polygons with Holes", Discrete Comp. Geom. 13, pp 77-109 (1995).

[12] H. Gonzalez-Banos and J.-C. Latombe, "Planning Robot Motions for Range-image Acquisition and Automatic 3D Model Construction", In Proc. AAAI Fall Symposium Series, 1998.

[13] T. Danner and L.E. Kavraki, "Randomized Planning for Short Inspection Paths", IEEE Intl. Conf. on Robotics and Automation, San Francisco, 2000.

[14] A.A. Argyros, C. Bekris, S. Orphanoudakis, "Robot Homing based on Corner Tracking in a Sequence of Panoramic Images", CVPR '2001, 11-13 December 2001, Hawaii, USA.

[15] D. Tsakiris, A.A. Argyros, "Corridor Following by Nonholonomic Mobile Robots Equipped with Panoramic Cameras", in the IEEE Med. Conf. on Control and Automation 2000, invited session in "Sensor-based Control for Robotic Systems", Patras, Greece, July 2000

[16] R. Seidel, "A Simple and Fast Incremental Randomised Algorithm for Computing Trapezoidal Decompositions and for Triangulating Polygons", CS Division, Univ. of California Berkeley, 1991.

[17] Joseph O'Rourke, "Computational Geometry in C", Cambridge University Press 1995.