Optimal Positioning of Sensors in 3D

Andrea Bottino and Aldo Laurentini

Dipartimento di Automatica e Informatica, Politecnico di Torino,
Corso Duca degli Abruzzi, 24 – 10129 Torino, Italy
{andrea.bottino, aldo.laurentini}@polito.it

Abstract. Locating the minimum number of sensors able to see at the same time the entire surface of an object is an important practical problem. Most work presented in this area is restricted to 2D objects. In this paper we present an optimal 3D sensor location algorithms that can locate sensors into a polyhedral environment that are able to see the features of the objects in their entirety. Limitations due to real sensors can be easily taken into account. The algorithm has been implemented, and examples are also given.

1 Introduction

Sensor planning is an important research area in computer vision. It consists of automatically computing sensor positions or trajectories given a task to perform, the sensor features and a model of the environment. A recent survey [15] refers in particular to tasks as reconstruction and inspection. Several other tasks and techniques were considered in the more seasoned surveys [19] and [12]. Sensor panning problems require considering a number of constraints, first of all the visibility constraint. To this effect, the sensor is usually modeled as a point, the vertex of a frustum if the field of view is taken into account, and referred to as a "viewpoint". A feature of an object is said to be visible from the viewpoint if any segment joining a point of the feature and the viewpoint does not intersects the environment or the object itself and lies inside the frustum. A popular 3D solution is locating the viewpoint onto the view sphere, which implicitly takes into account some constraints. Usually the view sphere is discretized, for instance by locating the possible viewpoints at the vertices of some semi-regular polyhedron, as the geodesic dome [4], [21], [22]. In this paper we will deal with a basic visibility problem, that is finding and locating the minimum number of sensors able to see at the same time all points of the surface of an object. The problem arises for tasks as static inspection and surveillance for several kind of sensors, as TV cameras, range sensors, etc. The sensors are supposed to be rotating or omni directional. The problem also arises in the area of image-based modeling and rendering [7]. A related problem is finding an inspection path optimum according to some metric, since according to a popular approach it is constructed joining static sensor positions which guarantee total object visibility [4], [6], [9], [22].

The major contribution of this paper is to present a 3D algorithm for finding a set of zones where a minimal set of viewpoints, able to observe the entire surface of the object, can be independently located. The algorithm works for multiply connected and unconnected general polyhedra, and can locate viewpoints able to observe the interior

M. Lazo and A. Sanfeliu (Eds.): CIARP 2005, LNCS 3773, pp. 804 – 812, 2005.

or the exterior surfaces of the polyhedra. For finding an optimal solution the view space needs not to be discretized, the only restriction being that each face must be completely observed by at least one viewpoint. This complies with the usual definition of feature of an object surface in term of faces or parts of faces, and with the practical requirement of observing a feature in its entirety. It is also worth observing that, if the faces are subdivided into smaller areas, the solution provided by the algorithm converges towards the optimal solution of the unrestricted problem. In the conclusion we will discuss how the approach can be extended in order to take into account additional constraints besides the visibility constraint.

The paper is organized as follows. In section 2 we will deal with the problem in 2D. Section 3 to 7 are devoted to describing the 3D algorithm. In section 6 we will also present some examples. Concluding remarks are contained in Section 8.

2 The Problem in 2D

Although in general the problem addressed is three-dimensional, in some cases it can be restricted to 2D. This is for instance the case of buildings, which can be modeled as objects obtained by extrusion. Much related work in the area of computational geometry stems from the popular Art Gallery Problem. The problem, stated in 1975 refers to the surveillance, or "cover" of polygonal areas with or without polygonal holes. The famous Art Gallery Theorem stated the upper tight bound $\lfloor n/3 \rfloor$ for the minimum number of "guards" (omni directional sensors) for covering any polygon with n edges, metaphorically the interior of an art gallery. The upper tight bound $\lfloor (n+h)/3 \rfloor$ was subsequently found for polygons with n edges and h holes. Many 2D variations of the problem have been considered, as for instance "rectilinear polygons", that is polygons with edges parallel or perpendicular, guards restricted to particular positions, etc. The original problem, as well as several restricted problems, have been found to be NP-hard [6]. For further detail, the reader is referred to the monograph of O'Rourke [13], and to the surveys [16] and [23]. At present, no finite exact algorithm exists able to locate a minimum unrestricted set of guards in a given polygon. In addition, no approximate algorithm with guaranteed approximation has been found.

Let us observe that our problem is similar, but not equal, to the classic Art Gallery problem, since we are interested in observing only the *boundary* of the object. Then, our 2D problem can be called the internal or external edge covering problem. A detailed analysis of the edge covering problem compared with the classic Art Gallery problem is reported in [10]. Among other results, it is shown that in general a minimal set of guards for the Art Gallery problem is not minimal for the interior edge covering, and that also the edge covering problem is NP-hard. However, edge covering admits a restriction which makes practical sense and allows to construct a finite algorithm which supplies a minimum set of viewpoints able to cover internal or external polygonal boundaries. The restriction is that each edge must be observed entirely by at least one guard, and it allows finding one or more sets of regions where a minimal set of viewpoints can be independently located. In addition, the algorithm asymptotically converges to the optimal solution of the unrestricted problem if the edges are subdivided into shorter segments. Finally, the algorithm can easily take into account

several constraints. Here we briefly present the essentials of a 2D sensor-positioning algorithm presented in [3]. The steps of the algorithm are as follows.

1. Divide the view space into a partition Π of maximal regions such that the same set of edges is completely visible from all points of a region.
2. Find the dominant zones (a zone Z of Π is dominant if no other zone Z* exists which covers the edges of Z plus some other)
3. Select the minimum number of dominant zones able to cover all the faces.

The idea of partition Π has been also proposed in restricted cases for robot self location in [17], [18]. Step 1), and 2) of the algorithm can be performed in polynomial time (see [10] for the details). Step 3) is an instance of the well-known set covering problem, which in the worst case is exponential. However, covering the edges using the dominant zone only usually substantially reduces the computation complexity. In addition, in many cases several dominant zones are also *essentials*, that is they cover some edges not covered by the other dominant zone, and must be selected. Observe that there could be minimal solutions also including non-dominant zones. However, replacing a non dominant regions with a dominant region covering the same edges plus some others provides multiple coverage of some edges, which is preferable for instance in the case of sensor failure. Some overlapping of the views is also useful for image registration.

3 The 3D Algorithm

The general idea of the 2D algorithm can be extended in 3D:

Step 1- Compute a partition Π of the viewing space into regions Z_i such that:

- The same set $\mathbf{F}_i = (F_p, F_q, ...F_t)$ of faces is completely visible from all points of $Z_i \; \forall$ i
- The Z_i are maximum regions, i.e. $\mathbf{F}_i \neq \mathbf{F}_j$ for contiguous regions.

The list of the visible faces must be associated to each region of Π.

Step 2- Select the *dominant regions*. A region Z_i is defined to be dominant if there is no other region Z_j of the partition such that $\mathbf{F}_i \subset \mathbf{F}_j$.

Step 3- Select an optimal (or minimum) solution. A minimum solution consists of a set of dominant regions $\mathbf{S}_j = (Z_{j1}, Z_{j2},...Z_{jk}....)$ which covers $\mathbf{F}=\cup$Fi with the minimum number of members.

In the following paragraph we will detail the steps of the algorithm. The environment is assumed to consist of simple polygons. Partition Π is built by means of a particular set of surfaces, called the *active visibility surfaces*. Each resulting region will be associated with the list of the faces that are completely visible from that zone. This set can be built traversing the partition graph from an initial region whose set of visible faces is known. Observe that interior inspection is similar, with a polygon enclosing the workspace and defining the outer border.

3.1 Partition Π

A *visibility surface* (VS) relative to a face divides the space into areas where the surface is partially or totally hidden. A VS is an half-open planar surface starting at one

of the edges or at a vertex of a face, lying in the half space opposite to the inner side of the face. Also, each potential VS has a positive side, which is the side closer to the originating face. The angle between the normal of this surface and the normal of the originating face is in the range $[0,\pi]$. Examples can be seen in **Fig. 1**, where the arrows mark the positive side of the VSs. Each VS can have an *active* and an *inactive* part. Only the active VSs are the effective boundaries of the visibility region of the corresponding surface. A VS is active when:

1. the angle with the normal of the originating face is 0 and the surface is not entering the object in the proximity of the originating vertex or edge (VS of type I)
2. it is tangent to another part of the object (or to another object) and in the neighborhood of this part, the inner side of the object lies on the negative side of the potential VS (that is, the VE surfaces defined in [8]). Those surfaces are defined by a vertex of the object and an edge of the face (VS of type II) or by an edge of the object and a vertex of the face (VS of type III). A surface of the first type is an unbounded triangle starting at the vertex of the object; a surface of the second type is an unbounded trapezoid with the object edge as one of its sides. In both cases, the active part of the VS starts at the intersection point (**Fig. 2**).

We can associate to each active VS an operator ^, where `^j` means that the surface is the boundary between a region where face `j` is hidden and a region where the face `j` visible, and `j` is the face the VS relates to. The operator has also a direction, which points to the area where the face is visible (see **Fig. 2**). In the following we will use a result found in [21], that is: if the face is convex (and simply connected), its visibility region is connected. This property is useful in order to avoid more complex situation and allows pruning radically the initial set of potential VSs of a face. Therefore any concave face will be split into convex parts.

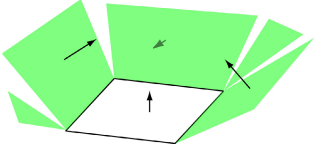

Fig. 1. Example of VSs

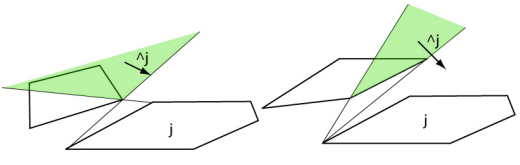

Fig. 2. VS of type II (left) and type III (right)

Finding the active part of a VS

For each initial VS, we must identify the part which is indeed active. In order to construct the active VSs of type I, we must find the regions of the plane P of a face F from where its 2D silhouette is completely visible. Forcing F to be convex, its 2D silhouette from a viewpoint V corresponds to the list of edges of F totally visible from the viewpoint. The active VS of type I can be constructed in the following way:

1. find on P the polygons corresponding to the intersection of the objects with P; let S, the initial active VS, be the union of all the regions in P where the 2D silhouette of F is completely visible;

2. for each edge, define as positive the side of the edge pointing to the internal part of the face; for each edge of the face closed by another face, or by another object intersecting the plane of the face, let H be the half plane in P bounded by the line containing the edge and corresponding to the positive side of the edge. Then S = S∩H (see **Fig. 3**).

Consider **Fig. 3** where a face F and its active VS of type I are shown; edges e_1 and e_2 are closed by other objects, H_1 and H_2 are the half planes bounded by e_1 and e_2.

Fig. 3. Active part of the VS of types I

The initial active VS on P can be evaluated using a modified version of the 2D region labeling algorithm of [3].

The active part of a VS of type II can be found determining the parts of the initial unbounded triangular surface where the originating edge is entirely visible. The algorithm is similar to the one used to find the active part of a VS of type I, that is:

1. let P be the initial unbounded VS of type II
2. find on P the polygons corresponding to the intersection of the objects with P; if one of the face is coplanar with P, it must not be considered if its normal is parallel to the positive direction of P
3. let S, the active VS, be the union of all the regions in P where the edge of F is visible

An example can be seen in **Fig. 4**.

Finally, the active part of a VS of type III can be found determining the parts of the initial unbounded trapezoidal surface where the originating vertex of F is visible. The algorithm is similar to the previous one, letting P at point 1 be the initial unbounded VS of type III. An example can be seen in **Fig. 5**.

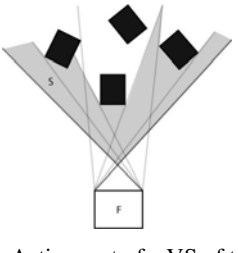

Fig. 4. Active part of a VS of type II

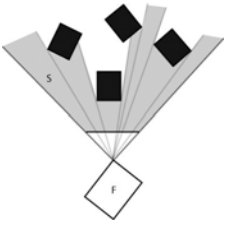

Fig. 5. Active part of a VS of type III

Additional rules for potential VS to be active
Some other conditions must be checked to identify an active VS.

For a potential VS of type II or III, its orientation must be such that the plane containing the surface intersects F only in the originating vertex or on the edges joining the vertex for a surface of type II, on the edge itself for a surface of type III. See for instance **Fig. 6**. The surface S relative to vertex V is lying on the plane P, whose intersection with the face f_1 is the line L. Therefore the surface S is not a potential VS.

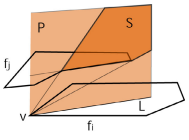

Fig. 6. Surface S is inactive

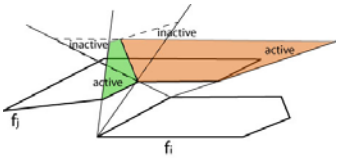

Fig. 7. Only part of these surfaces is active

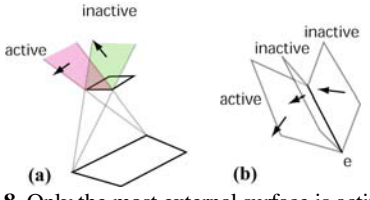

Fig. 8. Only the most external surface is active

Second, when the active parts of two VS relative to elements of the same face intersect somewhere, they both stop at their intersection (see **Fig. 7**). The part of the VS that falls on the negative side of another VS becomes inactive. Finally, consider a set of VS of type III insisting on the same edge e. If one of the VS is found to be on the negative side of another VS, then it is inactive. See for instance **Fig. 8**(a) and (b), where only the outermost surface is active. In (b), e is the edge common to all the VSs. The same rule applies to VSs of type II insisting on the same vertex when previous rules do not apply (that is, when the VSs are not intersecting).

3.2 The Algorithm

Given the definition of VS and operator ^, we can outline a region labeling algorithm:

1. find all the active VSs and the associate operator ^
2. intersect all the active VSs and subdivide the space into regions
3. select one region and compute the set of visibile faces V(f1,…,fn) for that zone
4. visit all the regions and compute their set of visible faces with the following rules:
   a. when crossing a boundary between R1 and R2 in the direction of the operator ^, the operand (j) is added to the set of visible faces of R2
   b. when crossing a boundary between R1 and R2 in the opposite direction of the operator ^, the operand (j) is removed from the set of visible faces of R2

An example of how the algorithm works can be seen in the following pictures. The original object is the L-shaped solid of **Fig. 9** (left). The expression ^(e,f) as a short form for ^e and ^f.

Now, let's imagine to place our viewpoint in the half-plane defined by face 2. The object, the VS surfaces and the corresponding regions as seen from this viewpoint are shown in **Fig. 9** (right). The picture depict also the operators ^ and their sign for the boundaries outgoing the plane of face 2 (the information about other boundaries have been omitted for clarity). Let's choose as starting region the central region of the figure, where the only visible face is 2. If we visit the partition moving southward, we cross a boundary declaring ^1 in the direction of crossing. The operand (1) will become visible, and in the second region 1 and 2 will be visible. Now moving to the right, we cross a boundary declaring ^3 in the direction of crossing. Therefore 3 will become visible in the arrival region. Let's make one more step upward. In the current region 1, 2 and 3 are visible. We cross the boundary in the opposite direction of ^1, therefore 1 will be hidden in the arrival region. By visiting all the regions following the rules specified in step 4 of the algorithm, the final result can be seen in **Fig. 10**.

The algorithm has been implemented. An example of the sensor positioning can be seen in **Fig. 11**, where the white spheres represents the position of the two sensors placed.

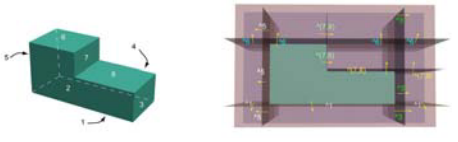

Fig. 9. Object (left) and boundaries of partition Π (right)

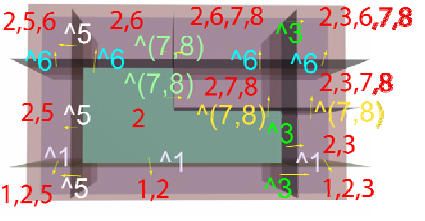

Fig. 10. Labeled regions

4 Complexity

To find the active VSs, given n faces, we have $O(n^2)$ VE surfaces. Checking if a surface intersects the polyhedron at one the edges can be done in constant time. For each surface, checking the extra conditions and finding the active surfaces requires intersecting each surface with any other and sorting the intersections. Overall $O(n^2)$ Vss can be obtained in $O(n^3 \log n)$ time. A classic algorithm can create the partition Π in $O(n^6)$ time. We should stress that this is the asymptotic complexity, while the difference is substantial in real world scenes. For instance, in the example of **Fig. 11**, the faces of the objects are 12, the active VSs after pruning 16, and the regions of the

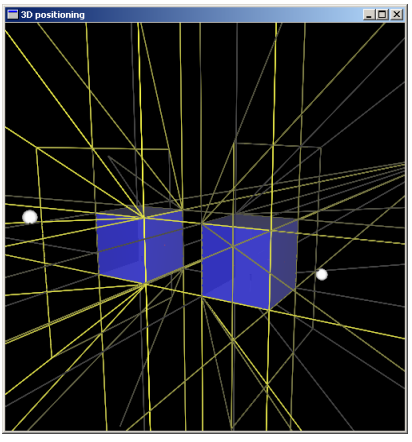

Fig. 11. Objects (blue), boundary lines of partition Π (yellow), and sensor position (white spheres)

partition are 75. Computing the visible surfaces of the starting region takes $O(n^2)$ time [14]. The time required for traversing the partition is $O(p)$ where p is the number of vertices of the partition (regions and edges also are $O(p)$) [2]. To find d dominant zones, we must compare the sets of visible faces of each region. This process can be shortened if we observe that a necessary condition for a region to be dominant is that all the positive crossing directions of the boundaries of the region lead to the interior of the region (except for the faces of the objects). Given c candidate found with this rule, d dominant regions can be found in $O(nc^2)$ time [10]. Step 3 requires, in the worst case, exponential time. However, an interesting alternative could be using a greedy heuristic, which selects the region covering the largest number of uncovered faces each time, requiring polynomial time.

5 Conclusions

In this paper a method for positioning a minimum number of sensors into a 3D polyhedral environment has been presented for some sample cases. With this approach is also simple to take into account additional constraints besides the visibility constraint by adding other rules to the process of generation of the Vss, since for each face f the constraints define a region C(f) of the space where the viewpoint can be located. The approach has been implemented and results have been presented. Future work will be focused on extending the algorithm in order to consider the general case of face covering, and not only its integer face covering restriction. The idea it is to develop an iterative algorithm for the general problem. This requires finding a lower bound for the number of sensors needed. Then we can evaluate the integer solution and check if they match. Otherwise, subdividing some of the initial surfaces and reapplying the integer algorithm can refine the solution. Rules for finding indivisible edges must be studied as well.

References

[1] S. Abrams, P.K. Allen, and K.A. Tarabanis, "Computing camera viewpoints in a robot work-cell," in Proc. 1996 IEEE Int. Conf. Robotics and Autom., pp.1972-1979

[2] Baase S, Computer algorithms. Addison-Wesley, New York, 1988

[3] A. Bottino, A. Laurentini "Optimal positioning of sensors in 2D". Proc. CIARP 2004, Puebla (Mexico)

[4] S.Y. Chen, and Y.F. Li," Automatic sensor placement for model-based robot vision," to appear in IEEE Trans. On Systems, Man ,and Cybernetics

[5] C. K. Cowan and P.D. Kovesi, "Automatic sensor placement from vision task requirements," ," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.10, no.3, May 1988, pp.407-416

[6] T. Danner and L.E. Kavraki, "Randomized planning for short inspection paths," Proc. 2000 IEEE Int. Conf. Robotics and Autom., pp.971-976, April 2000

[7] S. Fleishman, D. Cohen-Or, and D. Lishinski," Automatic camera placement for image-based modelling," Computer Graphic Forum, June 2000, 101-110

[8] Gigus, J. Canny, R. Seidel, "Efficiently computing and representing the aspect graphs of polyhedral objects", IEEE Trans. PAMI, Vol. 13, no. 6, pp. 542-551, June 1991

[9] G. D. Kazakakis, and A.A. Argyros, "Fast positioning of limited visibility guards for inspection of 2D workspaces," Proc. 2002 IEEE/RSJ Intl. Conf. On Intelligent Robots and Systems, pp.2843-2848, October 2002

[10] A. Laurentini, "Guarding the walls of an art gallery," The Visual Computer, vol.15, pp.265-278, 1999

[11] Nemhauser G. and Wolsey L., Integer and Combinatorial Optimization, John Wiley& Sons, 1988

[12] T.S. Newman and A.K. Jain, "A survey of automated visual inspection," Comput. Vis. Image Understand. , vol.61, no.2, pp.231-262, 1995

[13] J. O'Rourke, Art gallery theorems and algorithms, Oxford University Press, New York,1987

[14] Goodman JE, O'Rourke J, Discrete and Computational Geometry, Chapman and Hall, New York, 2004

[15] Scott W.R, Roth G. "View Planning for Automated Three-Dimensional Object Reconstruction and Inspection". ACM Computing Surveys, 2003, Vol. 35(1), pp. 64–96

[16] T. Shermer, "Recent results in art galleries," IEEE Proc. Vol. 80, pp.1384-1399, 1992

[17] K.T. Simsarian, , T.J. Olson, and N. Nandhakumar, "View-invariant regions and mobile robot self-localization," IEEE Trans. Robot. and Automat., vol. 12, no. 5 , pp. 810 –816, 1996

[18] R.Talluri and J.K.Aggarwal,"Mobile robot self-location using model-image feature correspondence," IEEE Trans. On Robotics and Automation, Vol.12, no.1, February 1996, pp.63-77

[19] K.A. Tarabanis, P.K. Allen, and R. Y. Tsai, "A survey of sensor planning in computer vision," ," IEEE Trans. Robot. and Automat., vol. 11, no. 1 , pp. 86 –104, 1995

[20] K.A. Tarabanis, P.K. Allen, R. Y. Tsai, "The MVP sensor planning system for robotic vision tasks," IEEE Trans. Robot. and Automat., vol. 11, no. 1 , pp. 72 –85, 1995

[21] K. Tarabanis, R.Y. Tsai, A. Kaul, "Computing occlusion-free viewpoint", IEEE Trans. PAMI, Vol. 18, no. 3, pp. 279-292, march 1996

[22] E. Trucco, M. Umasuthan, A.M. Fallace, and V. Roberto," Model –based planning of optimal sensor placements for inspection," IEEE Trans. Robot. and Automat., vol. 13, no. 2 , pp. 182 –194, 1997

[23] J. Urrutia, "Art Gallery and Illumination Problems" Handbook on Computational Geometry, Elsevier Science Publishers, J.R. Sack and J. Urrutia eds. pp. 973-1026, 2000