Author's version

Published in: Lecture Notes in Computer Science, Volume 5856 LNCS, 2009, Pages 707-715

Towards an Iterative Algorithm for the Optimal Boundary Coverage of a 3D Environment

Andrea Bottino

Politecnico di Torino, Dipartimento di Automatica e Informatica,
Corso Duca degli Abruzzi 24, 10129 Torino, ITALY
andrea.bottino@polito.it

Abstract. This paper presents a new optimal algorithm for locating a set of sensors in 3D able to see the boundaries of a polyhedral environment. Our approach is iterative and is based on a lower bound on the sensors' number and on a restriction of the original problem requiring each face to be observed in its entirety by at least one sensor. The lower bound allows evaluating the quality of the solution obtained at each step, and halting the algorithm if the solution is satisfactory. The algorithm asymptotically converges to the optimal solution of the unrestricted problem if the faces are subdivided into smaller parts.

Keywords: 3D sensor positioning, Art Gallery, lower bound.

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 [2] refers in particular to tasks as reconstruction and inspection. Several other tasks and techniques were considered in the more seasoned surveys [3] and [4].

Sensor panning problems require considering a number of constraints, first of all visibility. To this effect, the sensor is usually modeled as a point 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 intersect the environment or the object itself (usually excluding boundary points). Assuming omni-directional or rotating sensors, for tasks such as surveillance, the visibility constraint is modeled by the classic Art Gallery problem, which requires observing or "covering" the interior of a polyhedral environment with a minimum set of sensors. We call this the Interior Covering problem (IC). The problem tackled in this paper is similar, but not identical. It requires observing only the boundaries of P, faces for a polyhedral environment, and it applies, for instance, in problems like inspection or image based rendering. We call this the Face Covering (FC) problem.

FC and IC problems are NP-hard. However, "good" approximation algorithms are sorely needed. In our view, a "good" practical algorithm should not only be computationally feasible, but also provide a set of sensors whose cardinality, on the average, is not far from optimum. In this paper, we present a new FC sensor positioning technique. The algorithm is incremental and converges toward the optimal solution. A key feature of the algorithm is that it computes a lower bound, specific of the polyhedral environment considered, for the minimum number of sensors. It allows evaluating the quality of the solution obtained at each step, and halting the algorithm if the solution is satisfactory. The algorithm refines a starting approximate solution provided by an integer face covering algorithm, (IFC) where each face must be observed entirely by at least one sensor. A set of rules, aimed to reduce the computation, is provided for refining locally the current solution.

Compared to the large amount of literature on the 2D case, relatively few articles on 3D sensor positioning have been published. Furthermore, to our knowledge, currently no method for automatic sensor planning in a 3D polyhedral

environment, providing an evaluation of the coverage and information for improving it towards the optimum has been presented. Tarabanis presents in [6] an algorithm for computing the locus of viewpoints from which faces of polyhedral objects are seen in their entirety. These can be used solve the FC problems, but no indication where to locate a minimal number of sensor for seeing all the features of the object is provided. Rana [7] presents a 2D approach that can be applied to 3D as well, where two heuristics are presented for solving the boundary coverage problem. Again, non indication on optimality of the solution is given. In [5] a graph representation is used to group faces that satisfy all constraints simultaneously and hence are suitable for viewing from a common viewpoint. The set covering problem is solved with a greedy algorithm. The only attempt to define a quality measure for the covering is given in [8], where such measure is used to compute the minimal number of 3D sensor able to cover a polyhedral environment. However, sensor position is restricted to lie on the tessellated boundaries of a reduced area, the walking zone. To avoid the case of faces of the environment not covered entirely by one sensor, all "big faces" are initially split into smaller ones. However, no indications are given to when a face must be subdivided and no certainty of the fact that all sub-faces are visible from the same sensor can be given.

2 Outline of the algorithm

The algorithm aims at finding an optimal boundary cover of an environment P that is assumed to consist of polyhedra (with or without holes). Both internal and external coverage of the environment are managed. We stress that our work is focused on the optimality of the solutions provided by the algorithm. The approach is incremental, and it starts from an initial solution which is refined step by step. The initial step is given by a useful reduction of the covering problem, the Integer Face Covering (IFC), where each face must be covered in its entirety by at least one sensor. This (restricted) problem has an optimal solution, provided by the Integer Face Covering Algorithm (IFCA). In order to develop an effective incremental algorithm, it is also necessary to have a technique for evaluating at each step the quality of the current approximate solution, and an algorithm able to refine locally the solution, in order to reduce the computational burden and leading towards the optimum. A key component for the first step is the evaluation of a lower bound LB(P) on the number of sensors that is specific to the polyhedron P considered. Its value can be compared with the current solution and, if the result is not satisfactory, this can be refined by *dividing* some of the faces of P into smaller areas and applying again IFCA. For this task, the $INDIVA_{3D}$ algorithm allows finding the faces of P that must not be split (that is, the "*indivisible*" faces) since they are entirely observed by at least one guard of all optimal solutions.

The outline of the incremental algorithm is as follows:

- ***Step 1***. Compute a lower bound LB(P), specific to the polyhedron P, for the cardinality of the minimum set of guards using the algorithm LBA_{3D}.
- ***Step 2***. Compute an integer face cover of cardinality IFCC using the algorithm IFCA
- ***Step 3***. Compare LB(P) and IFCC. If they are equal, or the relative maximum error (IFCC-LB(P))/LB(P) is less than a predefined threshold, ***STOP***. Otherwise:
- ***Step 4***. Apply algorithm $INDIVA_{3D}$ for finding indivisible faces. If all faces are indivisible, ***STOP***, since IFC is optimal. Otherwise, split the remaining faces and compute a new lower bound.
- Compare the new lower bound and the current IFCC. If they are equal, ***STOP***. Otherwise go to ***Step 2***.

For a practical implementation, the algorithm can be halted if several consecutive steps have not changed the cardinality of the current solution. Clearly, the algorithm converges toward an optimal solution in an undefined number of steps. In the following paragraphs, we will detail the basic components of the algorithm.

2.1 Integer Face Covering Algorithm (IFCA)

Integer face covering (IFC) requires each face to be entirely covered by at least one guard. First, let us observe one fact. Let the *Integer Visibility Region* I(*f*) of a face *f* be the region of the viewing space whose points see entirely *f*. An IFC cover requires a sensor to be placed in the I(*f*) of every face of P. However, while a non empty I(*f*) exists for every convex face, this is not true in the case of concave faces. This can be seen from the example in Fig. 1(a), where,

considering internal covering of P, I(f_1) and I(f_2) are empty. Therefore, in order to guarantee the IFC problem has a solution, we require that any concave face is initially split into convex parts, as in Fig. 1 (b).

Given this initial constraint, a simple example showing the difference between FC and IFC is shown in Fig. 2, where three sensors are necessary for the integer covering of the faces of polyhedron P (a), while only two FC sensors are necessary (b).

Regarding complexity, a detailed analysis is omitted for the sake of conciseness, but is similar to the one presented in [1]. The relevant point is that IFC is NP-complete and finite algorithms are possible [9]. An algorithm of this kind, working for any polyhedral environment (external coverage of multiple polygons, internal coverage of polygons with or without holes) is described and implemented in [9]. Here we will present only the main lines of this algorithm, which are necessary for fully understanding its incremental extension. The steps of the IFC algorithm are the following:

IFCA

Step 1. Compute a partition Π of the viewing space into regions *Zi* such that:

- The same set $\boldsymbol{F_i}=(f_p, f_q,\ldots, f_t)$ of faces is entirely visible from each point of Z_i, $\forall i$
- The regions Z_i are maximal regions, that is $\boldsymbol{F_i} \not\subset \boldsymbol{F_j}$ where Z_j is any region bordering Z_i

Step 2. Select the dominant regions and the essential regions. A region Z_i is dominant if there is no other region Z_j such that $\boldsymbol{F_i} \subset \boldsymbol{F_j}$. An essential zone is a dominant zone that covers a face not covered by any other dominant zone.

Step 3. Select an optimal (or minimal) solution. A minimal solution consists of a set containing all the essential and some dominant regions $\mathbf{S_j} = (Z_{j1}, Z_{j2},..., Z_{jk})$ such that it covers all faces with the minimum number of members.

The main difference of the current approach with the algorithm in [9] is how Π is built. Here a two phase process is necessary. In the first phase, a more detailed partition Π', whose regions are also used by LBA_{3D} and $INDIVA_{3D}$, is constructed. In the second phase, Π' is refined to obtain Π.

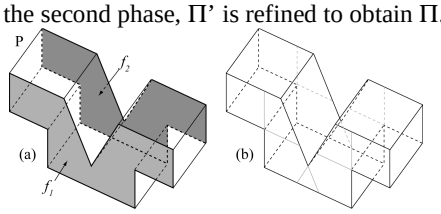

Fig. 1. In the case of internal covering, I(f_1) and I(f_2) are empty (a); a convex decomposition of the two faces (b)

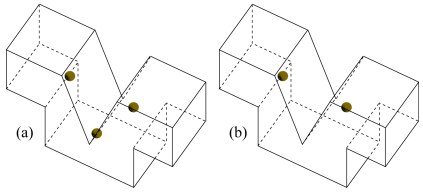

Fig. 2. Three IFC sensors are required (a) while only two FC sensors are necessary (b)

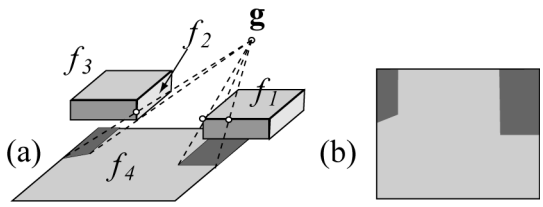

Fig. 3. An example of aspect of point **g** (a). The projection of the four occluding features (edges) of P on *f* (b)

Before defining Π', let us define the *aspect* **A**(**g**) of a point **g**:

$\mathbf{A}(\mathbf{g}) = ((f_h, n_h), (f_k, n_k), \ldots (f_q, n_q))$

where f_h, f_k, … f_q are the faces fully or partially visible from **g**, and n_h, n_k, … n_q are the number of occlusions for each face. The number of occlusions is, briefly, the number of edges of P that are entirely or partially projected on *f* from **g**. The aspect defines if a face *f*

is partially visible ($n_f \neq 0$), totally visible ($n_f = 0$) or not visible (f not in the aspect) from **g**. The word aspect has been used in agreement with the literature on aspect graphs. The interested reader is referred to the survey paper [10]. An example of aspect is shown in Fig. 3.

Partition Π' is defined as the partition that divides the interior/exterior of P into regions Z'_i such that

- All points of Z'_i have the same aspect $\mathbf{A_i}$
- Z'_i are maximum regions, i.e., $\mathbf{A_i} \neq \mathbf{A_j}$ for contiguous regions.

The construction of Π'can be performed using a set of *active patches*, belonging to *active surfaces*. The active patches are the boundaries between points whose aspects are different. The active patches are a subset of four kinds of active surfaces related to a face f of P:

- ***Type I***: the plane supporting f
- ***Type II***: surfaces originating from a vertex of f and tangent to an edge of P (VE surfaces), or from an edge of f and tangent to a vertex of P (EV surfaces)
- ***Type III***: EEE surfaces (ruled surfaces), tangent to an edge e of f and two other edges of P, or tangent to three edges of P and intersecting f
- ***Type IV:*** planar surfaces tangent to two parallel edges of P and intersecting f

According to the geometry of these surfaces, to each active surface can be associated one or more active patches, and to each patch a particular *visual event*. A visual event is a rule for changing the aspect of an imaginary viewpoint crossing the active patch, and it is synthesized by a *3D cross operator* having a positive and a negative direction. The *positive visual event* is the change of aspect of a point crossing the active patch along the positive direction; a similar definition holds for the negative visual event. Therefore, after constructing the partition Π' using the active patches, the aspect of each region can be constructed with a visiting algorithm, starting from a region whose aspect has been computed. The complete catalogue of active surfaces and active patches is shown in Fig. 4. The changes in the aspect due to the different 3D cross operators are listed in Table 1. A further analysis of the active patches might be necessary since, crossing an active patch T, f can be partially or totally hidden by other parts of the polyhedron P not related to the feature originating T. A detailed analysis of the different cases will not be performed here, for the sake of brevity.

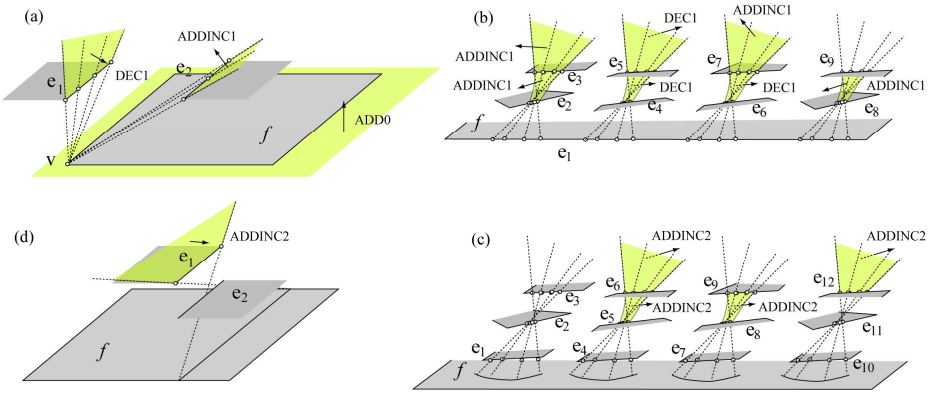

Fig. 4. The catalogue of active surfaces related to a face *f*. a) type I and type II surfaces. b) and c) type III. d) type IV. For each surface, the active patches are highlighted, together with their associated cross operator

3D Cross Operator	Positive visual event	Negative visual event
ADD0(f_i**)**	Add (f_i,*0*) to aspect	Delete (f_i, *0*) from aspect
DEC1(f_i**)**	$n_i = n_i - 1$	$n_i = n_i + 1$
ADDINC*k*(f_i**) *k*=[1,2]**	if(f_i in aspect) → $n_i = n_i + k$ else → Add (f_i, *k*)	if($n_i == k$) → Delete (f_i, *k*) else → $n_i = n_i - k$

Table 1. Cross operators and corresponding positive and negative visual events

2.2 Lower Bound Algorithm (LBA3D)

LBA_{3D} computes a lower bound of the number of sensors, specific to P, for the unrestricted sensor positioning problem. Both LBA_{3D} and $INDIVA_{3D}$ algorithms make use of the concept of weak and integer visibility regions of a face. They can be defined as follows:

- the ***Weak visibility region*** W(f_i) of a face f_i is the 3D region whose points see at least a point of f_i; points seeing only boundaries of f_i do not belong to W(f_i)
- the ***Integer visibility region*** I(f_i) is the 3D region whose points see entirely f_i

An example of weak and integer visibility regions for a face *f* of a simple polyhedron can be seen in Fig. 5.

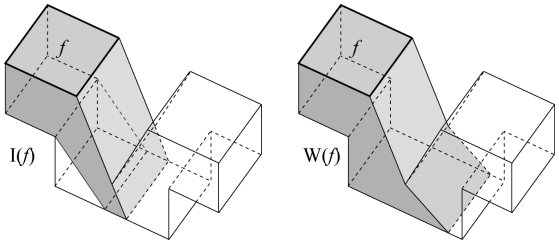

Fig. 5. Integer and weak visibility regions of face *f*

Both regions can be easily obtained as a byproduct of IFCA. In fact, given the aspect of each region of Π', W(f_i) and I(f_i) are simply obtained by merging the regions whose aspect

contains f_i, in the former case, or where the number of occlusions of f_i is zero, in the latter case. If there are p zones in the partition Π, and n faces in P, computing the visibility regions for all faces is O(pn).

Weak visibility regions allow us to determine a lower bound for the number of sensors needed. It is easy to see that:

Statement 1: *The cardinality of the maximal subset of disjoint (not intersecting) weak visibility regions $W(f_i)$ of P is a lower bound LB(P) for the minimal number of sensors.*

In fact, since each weak visibility region must contain at least one sensor, no arrangement of face covering sensors can have fewer sensors than LB(P).

Computing the lower bound requires solving the *maximum independent set* problem for a graph G where each node represents the weak visibility region of a face of P and each face of G connects nodes corresponding to intersecting visibility regions. The problem is equivalent to the *maximum clique problem* for the *complement graph* G' (the graph obtained by joining those pairs of vertices that are not adjacent in G). It is well known that these are again NP-complete problems, but *exact* branch-and-bound algorithms for these problems have been presented and extensively tested ([11], [12], [13]), showing more than acceptable performances for graphs with hundreds of nodes. Then, computing LB(P) is computationally feasible for practical cases.

2.3 INDIVisible faces Algorithm (INDIVA3D)

If optimal sets of sensors exist such that a face is entirely observed by at least one sensor of each set, then, in order to approach these optimal solutions, that face does not need to be split. Such a face is called *indivisible*. The rules for finding the indivisible faces of P are as follows:

Rule1. If $W(f_i) = I(f_i)$, f_i is indivisible.

Rule2. If $W(f_i) \subseteq I(f_j)$, f_j is indivisible.

Both rules follow from the fact that, for any solution, *at least one sensor of any minimal set* must be located in each weak visibility region. If the weak region is equal to the integer region of the face, rule 1, then for any solution the sensor placed in the weak region observes the face in its entirety. If the weak region is included in the integer region of another face, rule 2, then the sensor placed in the weak region observes the second region in its entirety. It follows that for every solution (and in particular for every optimal solution) these faces are observed in their entirety by at least one sensor and, therefore, they do not need to be divided.

A simple example will show how to apply these rules, and that they are powerful tools for simplifying the problem. Let us consider the polyhedron shown in Fig. 6(a) with the subdivision of its concave faces. The integer and weak visibility regions of face f_1 are coincident, and therefore for rule 1 the face is indivisible. The integer visibility region of f_2 is equal to the weak visibility region of f_1, and therefore is indivisible for rule 2. It can be easily seen that all the faces of P are indivisible, and then the unrestricted minimal set of guards is that provided by IFCA. The same result could have been obtained by computing the IFC solution, whose cardinality is equal to the lower bound LB(P).

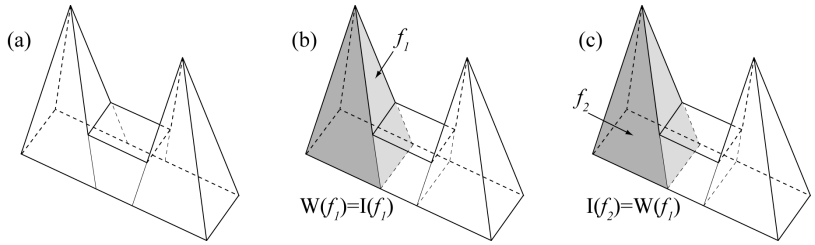

Fig. 6. In this case, the optimum FC and IFC covers are equal

Divisible faces must be partitioned, for instance, by splitting in two all the edges of the face, and connecting the central point of each edge with the face center.

2.4 Examples

A first example of how the algorithm works can be seen in Fig. 7. In (a) the polyhedron P is shown. LB(P) is 2, and in (b) the two not-intersecting weak polygons of faces f_1 and f_2 are shown, together with the initial solution of IFCA, whose cardinality is three. Applying rules 1-2, face f_3 is found to be the only divisible face and (c) shows its decomposition. Applying again IFCA, we obtain a solution with only two sensors (d), whose cardinality is equal to LB(P) and therefore is optimal.

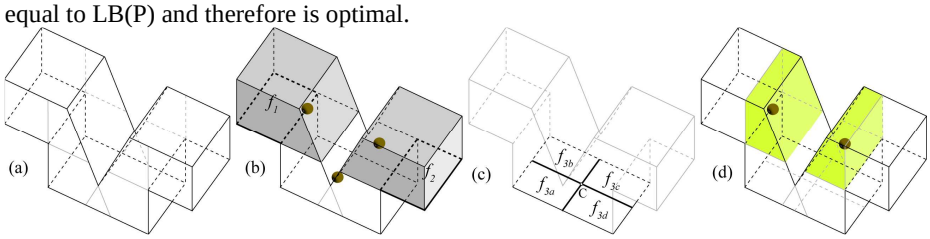

Fig. 7. Polyhedron P (a), $W(f_1)$ and $W(f_2)$, determining LB(P) = 2, and the initial IFC covering of cardinality three (b), subdivision of f_3 (c), the solution of the second iteration of IFCA and the regions where sensors can be located (d)

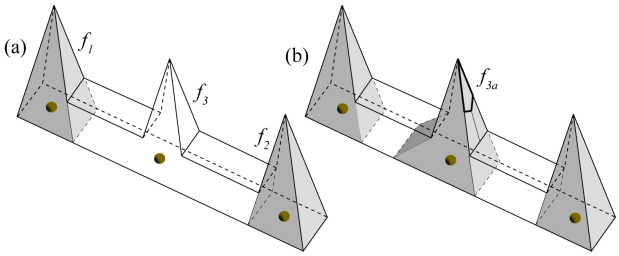

Fig. 8. Dividing faces of the initial polyhedron in (a) the lower bound can increase (b)

Observe that the lower bound is evaluated at every iteration. In fact, it might increase, and then improve, after splitting some faces of the polygon. An example can be seen in Fig. 8. In (a) the comb polyhedron is shown, together with the initial lower bound, whose value is 2, the non-intersecting weak regions of faces f_1 and f_2, and an initial IFCA solution, whose cardinality is three. Face f_3 is found to be divisible, and is split. Now, consider face f_{3a} drawn in bold in (b). Its weak polygon, also shown in the picture, does not intersect $W(f_1)$ and $W(f_2)$ and the value of the lower bound increases to three. The new

lower bound is equal to the cardinality of the previous IFCA solution that is, therefore, optimal.

3 Conclusions

This paper presents an incremental algorithm for positioning sensors in a 3D environment that are capable of seeing in their entirety the external or internal boundaries of a polyhedral environment. The approach is iterative and it is based on a lower bound on the number of sensor that allows to evaluate the closeness to optimality of the solution and to define rules for trying to improve the current solution. This is, in our knowledge, the first work in literature that attempts to tackle this problem in 3D. Future work will be focused on the full implementation of the algorithm, which is a rather complex task, especially for the generation and intersection of the active patches, and to extend it to take into account several additional constraints besides the visibility one.

References

[1] Bottino, A., Laurentini, A.: A Nearly Optimal Sensor Placement Algorithm for Boundary Coverage. Pattern Recognition, 41(11), 3343–3355 (2008)

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

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

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

[5] Roberts, D.R., Marshall, A.D.: Viewpoint Selection for Complete Surface Coverage of Three Dimensional Objects. Proc.of the Britsh Machine Vision Conference (1998)

[6] Tarabanis, K.; Tsai, R.Y.; Kaul, A.: Computing occlusion-free viewpoints. IEEE Trans. Pattern Analysis and Machine Intelligence, vol.18(3), pp.279-292 (1996)

[7] Rana, S.: Two approximate solutions to the Art Gallery Problem. Proc. of International Conference on Computer Graphics and Interactive Techniques (2004)

[8] Fleishman, S., Cohen-Or, D., Lischinski, D.: Automatic Camera Placement for Image-Based Modeling. Proc. 7th Pacific Conf. on CG and Applications. pp.12-20 (1999)

[9] Bottino A., Laurentini A.: Optimal positioning of sensors in 3D. Proc. Proceedings of 10th Iberoamerican Congress on Pattern Recognition, Havana, Cuba, November 15-18 (2005)

[10] Schiffenbauer R.: A Survey of Aspect Graphs. Polytechnic University, Brooklyn, Technical Report TR-CIS-2001-01 (2001)

[11] Woods, D.R.: An algorithm for finding a maximum clique in a graph. Operations Research Letters,Vol.21, pp. 211-217 (1997)

[12] Carraghan, D.R., Paradalos, P.: An exact algorithm for the maximum clique problem. Operations Research Letters, 9:375-382 (1990)

[13] Oestergard, P.: A fast algorithm for the maximum clique problem. Discrete Applied Mathematics,Vol.120, pp.197-207 (2002)

[14] Fleishman, S., Cohen-Or, D., Lischinski, D.: Automatic Camera Placement for Image-Based Modeling. Proc. Of 7th Pacific conf. on Computer Graphics and Applications (1999)