Radiosity for dynamic scenes in flatland with the visibility complex

Rachel ORTI, Stéphane RIVIERE, Frédo DURAND and Claude PUECH

iMAGIS[†]/ GRAVIR - IMAG, BP 53, F-38041 Grenoble Cedex 09, France
(E-mail: Rachel.Orti@imag.fr)

Abstract

The radiosity method is particularly suitable for global illumination calculations in static environments. Nonetheless, recent applications of image synthesis such as architectural simulation or lighting design require the ability to modify environments. Previous methods have attempted to deal with dynamic environments (environments where the geometry, the material properties, etc., can change) but still suffer some limitations in the case of moving objects. One of the main problems remaining is the efficient and accurate detection of which form factors must really be recomputed, since their calculation is the most time-consuming part of the radiosity method. To correctly understand and solve this problem, we start with a method in 2D for polygonal scenes using the visibility complex. It is a powerful data structure representing the visibility relationships between objects in the plane. We have developed and implemented an algorithm which uses this structure to efficiently compute the discontinuity mesh and the form factors for static scenes. We also propose an extension to our algorithm to efficiently update only the modified form factors when an object is moving. This approach enhances our understanding and will hopefully lead to efficient solutions in 3D.

Keywords: global illumination, radiosity, discontinuity mesh, form factor, visibility, dynamic environments

1. Introduction

The radiosity method is particularly suitable for visualization of interiors and can thus be used in many applications such as architectural simulation or lighting design. Applications of this type require the capability to modify the scene (move objects, change the material properties, etc.) and also to deal with complex geometry. To be usable, they must use algorithms fast enough to provide a new global solution whenever changes are made to the scene, at interactive if not real-time update rates.

The requirement to rapidly render complex environments has motivated research efforts in visibility processing, since visibility calculations are very important in the rendering process. Specifically in the radiosity method most of the time is spent in the calculation of the form factors (more than 50% of the time in the case of efficient algorithms as shown in [1] for example). Therefore the idea is to build a special data structure that allows for easy determination of the set of potentially visible objects. Most notably, Teller [2,3] has proposed global visibility algorithms that preprocess polygon databases in order to accelerate visibility determination during illumination calculations.

Although the radiosity method was initially applied to static environments [4,5], recently researchers have attempted to deal with dynamic environments [6,7,8,9]. These approaches typically begin with a current radiosity solution and then try, for a given modification of the environment, to *incrementally* compute a new solution from the current one. The difficulty remains in the efficient and accurate identification of the form factors that really need to be recomputed, without considering the others. In order to limit re-computations, several different approaches have been proposed. Chen [7] has limited the computation of the form factors by considering only the

† iMAGIS is a joint project of CNRS, INRIA, Institut National Polytechnique de Grenoble and Université Joseph Fourier.

fraction of the hemi-cube that bounds the extent of the projection of the new object. George *et al.* [8] have used a shadow volume to cull away patches that cannot possibly require a new form factor. These two methods that use progressive refinement radiosity, still perform unnecessary computations and are not interactive in the case of moving objects. Shaw [9] has applied hierarchical radiosity to dynamic environments. She has used a "motion volume" that encloses the range of motion for an object (in the same spirit as [6]) and has considered as affected only those links that intersect this volume. But still too much re-computation is performed. No method has yet been proposed which exactly identifies the form factors that really need to be recomputed.

In [10] we discussed some computational geometry aspects related to the method we present here, but for curved objects. In this paper, we propose a method permitting the efficient computation of the discontinuity mesh and the form factors for 2D polygonal scenes. This method uses the visibility complex (a data structure that represents visibility relationships between objects in the plane). The complex allows us to consider only the mutually visible parts of a pair of objects and thus permits the computation of only the strictly necessary form factors. We have a running implementation which computes the discontinuity mesh and radiosity using the visibility complex for static scenes. We also detail an algorithm for the form factors in dynamic environments and show that the visibility complex, by permitting the efficient identification of the visibility relationships that change, allows us to restrict the re-computation of the form factors to exactly those that change.

The geometric complexity of the visibility relations led us to begin this research with a thorough solution in 2D. In two dimensions analytic solutions permit the validation of the models, and in addition, the scenes are much simpler to understand and allow for better overall comprehension of the phenomena. The structures used (the augmented visibility complex in particular) permit a comprehensive description of all visibility relations required for form-factor calculations. These structures, manageable in 2D, give a better comprehension of the intricate visibility interactions for dynamic scenes, a step necessary for the development of truly efficient algorithms for radiosity in dynamic 3D environments. This approach, i.e. the study of the simpler two-dimensional case to facilitate understanding, leading to a 3D solution, is common in illumination research (e.g., for discontinuity meshing and radiosity calculations [11, 12] or wavelet radiosity approaches [13, 14]).

2. The visibility complex

2.1. Visibility computation

The most time consuming part of a radiosity algorithm is the visibility calculation when computing form factors. For each pair of patches, one must find the part of one patch which is visible from the other. The complexity can be greatly improved if we can consider only pairs of patches that are mutually visible, and if, for those pairs, we can have direct access to their mutually visible parts.

To solve visibility problems in the plane, computational geometry has given particular attention to global data structures that encode visibility information. They are used to efficiently solve global visibility problems or to answer multiple queries concerning local visibility problems. A well known data structure is the visibility graph. Its vertices are, for a polygonal scene, the polygon vertices, and its edges link two mutually visible vertices. But because of its discrete structure, it does not carry enough information for our visibility computation.

To cope with this problem, Pocchiola and Vegter [15] introduced a new data structure, the *visibility complex*, which represents sets of rays going from one object to another.

2.2. The visibility complex

We consider here polygonal scenes in the plane. The elementary objects considered are not the polygons themselves, but the segment lines forming their sides.

When dealing with visibility, one deals with oriented maximal free line segments, that is line segments in free space of maximal length. Such segments will be called *rays*. The extremities of a ray lie on the border of the two objects that stop the ray. These two objects O_l and O_r (stopping the ray on its left and on its right) constitute the label (O_l, O_r) of the ray. The visibility complex is the set of equivalence classes of rays according to the following equivalence relation: two rays are equivalent if one can pass continuously from one to another while keeping its label constant.

These equivalence classes are of three types: faces, edges and vertices. The faces are 2D components representing rays going from O_l to O_r without touching any other object (see Figure 1). The edges are 1D components representing rays with label (O_l, O_r) passing through the same polygon vertex. Finally the vertices are 0D components representing rays passing through two polygon vertices, that is edges of the visibility graph. An edge is therefore delimited by two vertices. Edges represent limits of zones of constant visibility. They bound the faces of the complex. A study of the complex also shows that an edge is incident to at most three faces and that a vertex is incident to four edges and to six faces. The visibility complex was introduced in [15]

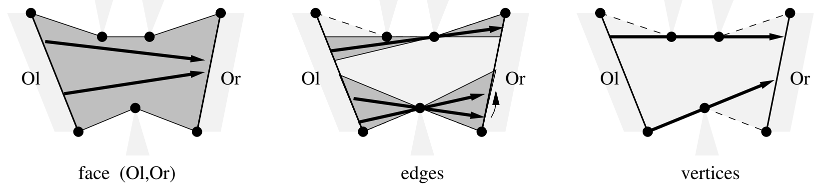


Figure 1. *Elements of the visibility complex.*

for scenes of curved convex objects in the plane, where an optimal incremental algorithm for computing the complex is given. An optimal sweep algorithm is described in [16]. The complex is studied for general polygonal scenes in [17], where an optimal sweep algorithm is given.

2.3. Duality

Sets of lines (or rays) are difficult to handle directly, and visualizing the elements of the visibility complex and their incidence relation is not evident. Dealing with lines (or rays) becomes easier when considering them in a dual space. The duality principle allows the representation of a line in the scene as a point in a dual space. In this paper we consider the duality relation which associates with the line $l : y = ax + b$ the dual point $l^* : (a, b)$. Notice that rays contained in the same line are mapped to the same dual point.

Visibility along a line changes when the line crosses a polygon vertex of the scene. In the dual space, the set of lines passing through a polygon vertex $p : (a, b)$ is the dual line $p^* : y = -ax + b$. The vertices of the polygonal scene induce an arrangement of lines in the dual space, supporting the edges of the visibility complex. Figure 2 illustrates the correspondence of the elements of the complex between the scene and the dual space and also illustrates more clearly the structure of its faces. A face has two extremal vertices (the left one and the right one) separating the edges bounding the faces into two sets: the upper and lower chains of edges. Sets of rays are now turned into sets of points in the dual space and are easier to handle. Figure 3 shows a view

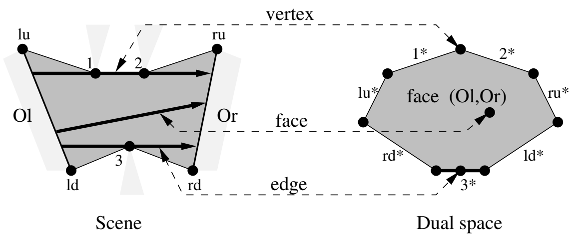


Figure 2. *Correspondence between the scene and the dual space.*

of the complex around one of its vertices. This figure also shows more clearly the incidence relations between elements of the complex.

3. Radiosity in flatland

3.1. The form factor

The form factor F_{ij} between two surface elements A_i and A_j is the fraction of energy leaving A_i reaching A_j [5]:

$$F_{ij} = \frac{\text{Radiative energy reaching } A_j \text{ from } A_i}{\text{Total radiative energy leaving } A_i \text{ in all directions}} . \quad (1)$$

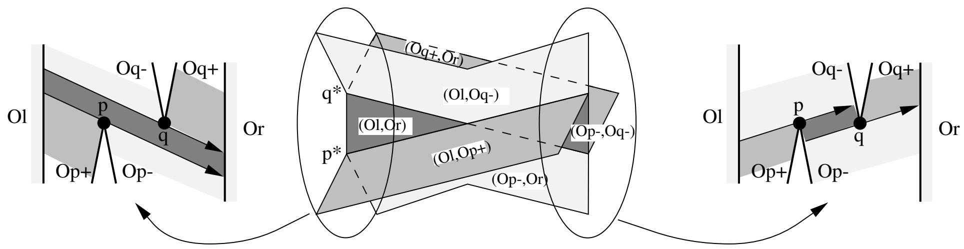
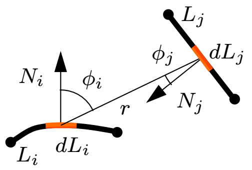


Figure 3. *Visibility complex around a vertex.*

This quantity is expressed in 3D as a double integral over areas, which takes into account the visibility between surfaces. In the 2D case, line segments are the equivalent of surfaces and thus the form factor becomes a double integral over segments. Figure 4 shows its expression for two elements L_i and L_j.



$$F_{ij} = \frac{1}{L_i} \int_{L_i} \int_{L_j} \frac{\cos\phi_i \cos\phi_j}{2r} H_{ij} dL_j dL_i,$$

$$\text{where } H_{ij} = \begin{cases} 1 & \text{if } dL_i \text{ and } dL_j \text{ are mutually visible} \\ 0 & \text{otherwise.} \end{cases}$$

Figure 4. *Formulation of the 2D form factor.*

The form factor is a strictly geometric quantity: it depends only on the shape and relative location of elements in the scene. This property appears in the "string rule" established by Hottel [18] in thermal engineering that we will use to compute the form factors:

String rule: The 2D form factor between two segments C_i and C_j is obtained by computing the length of "strings" drawn between the endpoints of C_i and C_j (see Figure 5). The strings stretched from the endpoints of C_i to the corresponding endpoints of C_j (i.e., a to c and b to d) are called *un-crossed strings*, and those drawn to the opposite endpoints on C_j (i.e., a to d and b to c) are called *crossed strings*.

The form factor between the two segments C_i and C_j (with lengths L_i and L_j respectively) can be expressed as:

$$F_{ij} = \frac{\sum \text{length } \textit{crossed strings} - \sum \text{length } \textit{un-crossed strings}}{2L_i}. \tag{2}$$

When some parts of C_i and C_j are not mutually visible, the strings are simply "stretched" around the obstruction (see the "string" between a and c in Figure 5) and formula 2 is still valid.

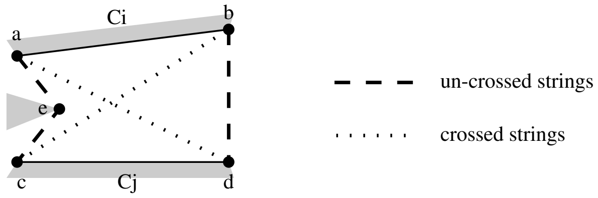


Figure 5. *Strings for two portions of curves C_i and C_j.*

When considering the visibility complex, the form factor defined by a weighted sum of curve lengths can be re-expressed as a weighted sum depending on vertices bounding a face of the complex:

Expression in a dual space: The line segments that compose the strings corresponding to a pair (C_i, C_j) of objects are edges of the visibility graph. These edges correspond to vertices in the visibility complex which bound the corresponding face with label (C_i, C_j). More precisely, the two extremal vertices correspond to the crossed strings and the other vertices of the two chains of edges of the face correspond to the two un-crossed strings (see Figure 6a). If we compute the length $d(v)$ of the corresponding line segment in the scene for each vertex v of the complex, then the "string rule"(2) becomes:

$$F_{ij} = \frac{\sum_{v\ bounding\ face} d'(v)}{2L_i}, \text{ where } d'(v) = \begin{cases} d(v) & \text{if } v \text{ is an extremal vertex} \\ -d(v) & \text{otherwise} \end{cases} \quad (3)$$

If the opposite extremities of C_i and C_j are hidden by an obstacle, some segments of the strings do not

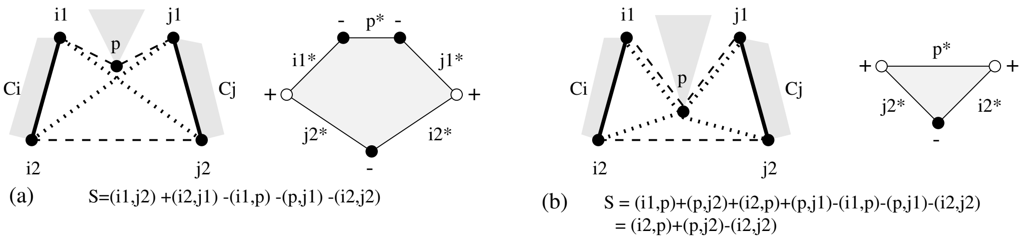


Figure 6. *String rule in the visibility complex.*

correspond to vertices bounding the face. But these segments are part of the crossed and the un-crossed strings, so they disappear from the computation and the rule is still valid (see Figure 6b).

3.2. Discontinuity meshing and radiosity sampling

In order to apply the radiosity method to a polygonal scene, each edge of a polygon must be divided into small elements (or samples) for which the form factors must be computed. The accuracy of the radiosity solution depends on the discretization of the environment. Changes in visibility in the scenes produce discontinuities of illumination and indirect illumination. Such discontinuities are important to consider when computing the samples since an element typically has constant radiosity. An appropriate method is to position the elements according to a discontinuity mesh whose boundaries are placed on the radiosity discontinuities caused by occlusions [11]. However, this meshing strategy traditionally requires many geometric calculations which make it expensive to use. The visibility complex is very useful in this case since it allows to compute the discontinuity mesh very easily. In what follows we describe an augmented structure to directly provide the form factors for the samples obtained.

Let us consider a face (O_l, O_r) in the complex. If we consider O_l as being lit by O_r, then the objects associated with the edges that bound the face introduce discontinuities in the illumination of O_l. More precisely, the points of discontinuities (l_i) are the intersections between O_l and the (extended) edges of the visibility graph that bound the face. In the complex, the face is subdivided by dual lines that pass through the dual point of O_l and the vertices bounding the face. Figure 7(a) shows an example where a sub-face and the corresponding set of rays in the scene are highlighted. They correspond to rays that can see the lower extremity of O_r but that are blocked by the vertex 2 when looking at the upper extremity of O_r. The same reasoning applies when considering O_r as being lit by O_l. Obstacles induce illumination discontinuities (r_i) on O_r, and a corresponding subdivision appears in the corresponding face in the complex. Figure 7(b) illustrates the set of rays leaving O_r that can see both extremities of O_l.

If we divide both objects O_l and O_r into elements according to their illumination discontinuities (l_i) and (r_i), we notice that the form factors between these elements can be expressed by considering the sub-faces of the complex induced by both subdivisions. Figure 7(c) shows the set of rays going from patch $[l_1, l_2]$ to patch $[r_1, r_3]$ in the scene and the corresponding sub-face in the dual space.

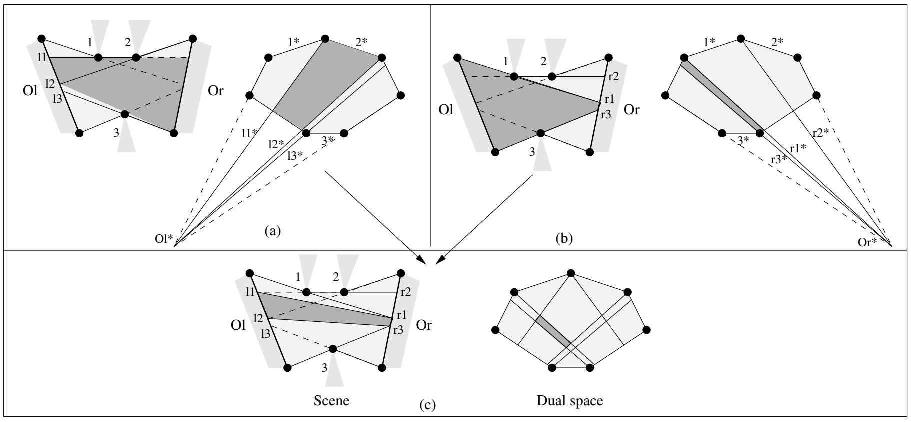


Figure 7. *Discontinuity mesh for a face*

4. Implementation - Results

A program has been implemented which first computes the visibility complex for a 2D polygonal scene. It then uses this structure to compute the discontinuity mesh and the radiosity solution of this scene. It also provides different types of visualization: the scene with the radiosity value for each element, the radiosity matrix and a 3D visualization of the visibility complex.

In this section we give some details about the different algorithms we have implemented.

4.1. Computing the discontinuity mesh

Heckbert studied radiosity in flatland for polygonal scenes [11, 12]. In [12], he computed D^1 discontinuities (discontinuities occurring at critical points where there is a change in visibility) with a straightforward ray tracing algorithm running in $O(n^3)$ time (where n is the number of edges of polygons). As he states, the running time can be improved to $O(n^2 \log n)$ by using a radial sweep-line perspective visibility algorithm.

In our program, the visibility complex is constructed using the code of the output-sensitive algorithm of [17]. This algorithm runs in optimal $O(m+n \log n)$ time (where m is the number of vertices of the visibility complex, that is the size of the visibility graph, which is $\Omega(n)$ and $O(n^2)$), and is efficient in practice too (see [17]). Once the visibility complex is built, the discontinuity mesh can be computed by considering the vertices of the complex, in $O(m)$ time. Figure 8 shows the algorithm we use to compute the discontinuities. Care has been taken to implement all parts of the algorithm using the appropriate data structures, permitting truly optimal running time. The different types of discontinuities named (1), (2), (3), and (4) in the algorithm are illustrated in Figure 9. The discontinuity of type (5) is not illustrated since it is a symmetrical case of type (4). In Figure 9, the objects O_l and O_r associated with the face correspond to the edges of polygons with endpoints (A_1, A_2) and (B_1, B_2) respectively.

4.2. Computing the form factors between elements

The form factor between two elements is null if these elements are not mutually visible. In practice, such situations are frequent and should be taken into account in order to avoid unnecessary computations. Heckbert [12] has noticed in his tests that matrix density α (α is the fraction of nonzero elements in the radiosity matrix) typically ranged between 10% and 40%.

The visibility complex is very useful in this case, since it allows one to consider only pairs of mutually visible objects and then to accurately examine only the mutually visible parts of these objects. To compute the necessary form factors between two objects O_l and O_r, we just have to consider all the faces of the complex labelled (O_l, O_r).

We define a *zone of interference* associated with a given edge of a face of the complex, as the zone in the scene which is between the two objects associated with the face and which is bounded by the two straight lines

Computation of the discontinuities

For each face f of label (O_l, O_r) **do**
  For each edge e of f not associated with O_l or O_r **do**
    For each vertex V which is an extremity of e **do**
      If V is also the end of an edge associated with O_l **(1)**
        Compute discontinuity on O_r
      Else If V is also the end of an edge associated with O_r **(2)**
        Compute discontinuity on O_l
      Else Compute corresponding discontinuities on O_l and O_r **(3)**
    End For
  End For
  If incident edges to the first vertex of f are both associated with O_l (resp. O_r) **(4)**
    Compute discontinuity on O_r (resp. O_l)
  If incident edges to the last vertex of f are both associated with O_l (resp. O_r) **(5)**
    Compute discontinuity on O_r (resp. O_l)
End For

Figure 8. *Discontinuity meshing algorithm.*

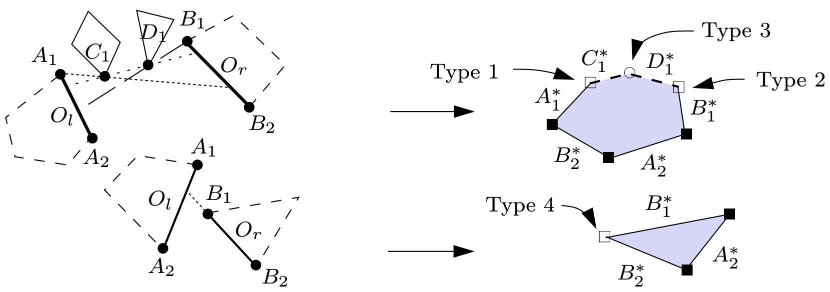


Figure 9. *Different types of lighting discontinuities.*

(in the scene) associated with the left and the right vertices of this edge. This zone corresponds to the region (in the scene) between the left and the right object of the face where the point associated with the current edge may obstruct the visibility. For example, in Figure 11, the region colored in light gray in the first schema corresponds to the *zone of interference* of the edge c^*.

During the computation, we maintain for a given face f the two current potential *zones of interference* ZI_{up} of $list_{up}$ (the list of upper edges of the face) and ZI_{down} of $list_{down}$ (the list of lower edges of the face). The points associated with the zones of interference of $list_{up}$ and $list_{down}$ are respectively called p_{up} and p_{down}.

Knowing the current zones of interference of $list_{up}$ and $list_{down}$ and the position of a sample on the left object relative to these zones, we can easily determine the part seen by this sample on the right object. For a given sample s_l with endpoints (p_{i0}, p_{i1}) on the left object O_l, the visible part on the right object is the interval between the points $p_{j_{up}}$ and $p_{j_{down}}$ (see Figure 11 for an example):

- $p_{j_{up}}$ is the intersection of line (p_{i1}, p_{up}) with O_r if it exists, and otherwise the nearest endpoint of O_r.
- $p_{j_{down}}$ is the intersection of line (p_{i0}, p_{down}) with O_r if it exists, and otherwise the nearest endpoint of O_r.

The algorithm of Figure 10 describes how we determine the form factors $ff(s_l, s_r)$ for all pairs of samples (s_l, s_r) of a given face f of the visibility complex.

To better understand the method, the execution of the algorithm is shown on a small example where we compute the form factors for samples on two edges O_l and O_r of polygons (see Figure 11). For each step we show the current edges in $list_{up}$ and $list_{down}$ (represented in dashed and dotted points respectively) and the corresponding *zones of interference* in the scene. The current zone of interference of $list_{up}$ is represented in light gray and the one of $list_{down}$ in dark gray. The part of the left edge O_l currently considered is indicated

Computation of the form factors of the face $f(O_l, O_r)$

Initialization

ZI_{up} = zone of interference associated with the first obstructing point in $list_{up}$
ZI_{down} = zone of interference associated with the first obstructing point in $list_{down}$
s_l = first sample on O_l in the face f
Compute points $p_{j_{up}}$ and $p_{j_{down}}$ according to the current value of s_l
s_r = first sample with a part in $[p_{j_{up}}, p_{j_{down}}]$

Iteration

```
For each sample s_l ∈ ZI_up
    For each sample s_r with a part in [p_j_up, p_j_down]
        If s_l ∉ ZI_down
            If s_r ∉ ZI_up
                Compute ff(s_l, s_r) with total visibility
            Else Compute ff(s_l, s_r) with potentially obstructing point p_up
        Else If s_r ∈ ZI_down
                If s_r ∉ ZI_up
                    Compute ff(s_l, s_r) with potentially obstructing point p_down
                Else Compute ff(s_l, s_r) with potentially obstructing points p_up and p_down
              Else If s_r ∉ ZI_up
                    Compute ff(s_l, s_r) with total visibility
                 Else Compute ff(s_l, s_r) with potentially obstructing point p_up
    End For
    Update points p_j_up and p_j_down according to the current value of sample s_l
End For
Next edge of list_up (if it exists)
ZI_up = zone of interference associated with this edge
If beginning of s_l = lower limit of ZI_down
    Next edge of list_down (if it exists)
    ZI_down = zone of interference associated with this edge
```

Iteration stops when the computations have been completed for the last sample s_l of the face f

Figure 10. *Algorithm for form factors computation.*

with a line drawn along O_l. We assume here that this part corresponds to a sample s_l. We then show on the edge O_r the different regions for which there is a specific type of form factor (i.e. "with total visibility", "with potentially obstructing point p_{up}", etc., according to the algorithm in Figure 10).

The different types of form factors are computed using the method described in previous sections. Typically for a given type, "with potentially obstructing point p_{up}" for example, we can identify different configurations for which there is a specific formulation of the form factor. These configurations are related to the different regions where the point p_{up} can be located. Figure 12 shows the different possible regions (colored in light gray) containing the point p_{up} and the face corresponding to each configuration. For each case it also shows the expression of the sum of curves lengths defined in the "string rule".

4.3. Results

Test runs have been performed. We show in color Plate 1 a simple sample scene made of a room with three objects inside. The source is the white square and the other square is moved. The entire solution is recomputed after each change since the dynamic update (see section 5) has not yet been implemented. The scene is represented in a pseudo-3D view with the radiosity values projected onto both sides of the walls. This increases the quantity of information seen from a given view point. The corresponding visibility complex is represented with the edges in blue. The external faces are represented in transparent gray, and the faces relative to the moving square are in red; they correspond to the modified form factors. The other faces are not shown because they do not change (and also for clarity's sake). We can see here that the modifications of the visibility complex are local, around the dual lines of the vertices of the moving square. The complex has 106 vertices. There are 71 samples in the

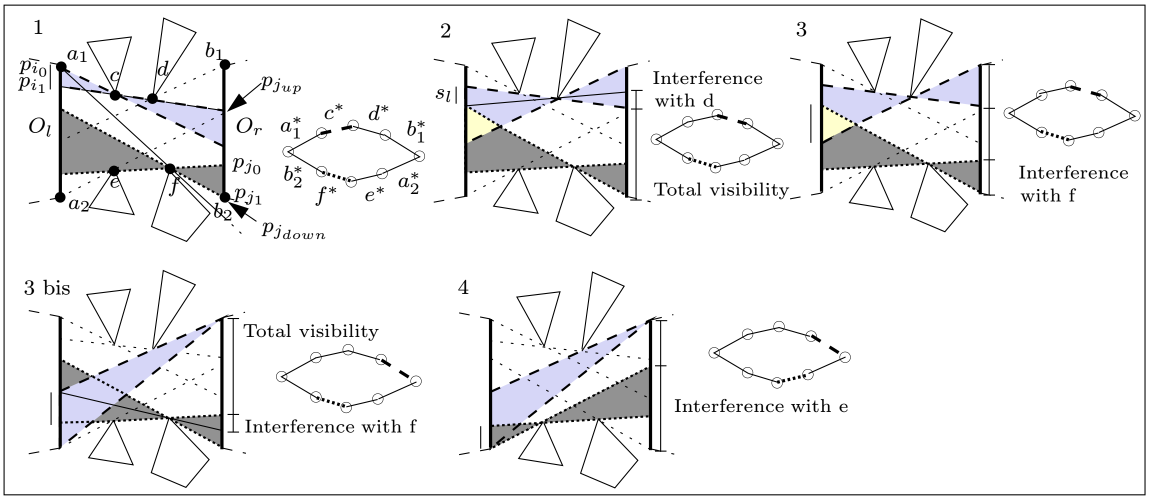


Figure 11. *Zones of interference during form factor computation.*

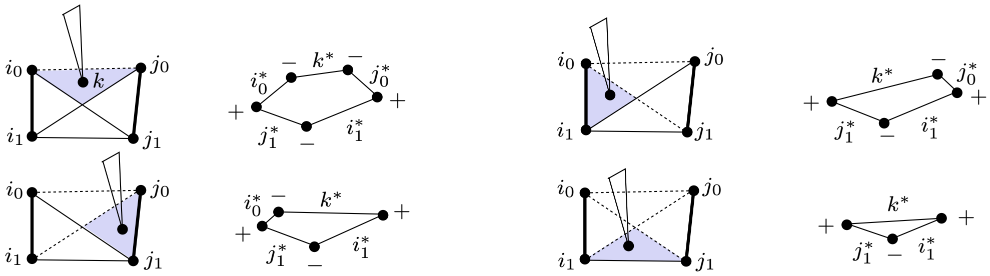


Figure 12. *Different upper obstructions for a pair of elements.*

scene and 2093 nonzero form factors. The total calculation took less than 0.2s on a SGI Indigo2 workstation with a non-optimized program. Another scene with 229 samples was calculated in about 1s.

5. Radiosity in dynamic environments

In this section, we study dynamic environments, that is scenes where objects are moving. To efficiently update the radiosity computation we must consider only form factors affected by the moving objects. In fact when an object O is moving in the scene, we must take two things into account. First, even if there is no change in visibility (the visibility complex is topologically unchanged), form factors concerning elements on this object must be numerically recomputed. Secondly, visibility can change between objects in the scene. In this case, the complex must be updated: some faces are destroyed (and the corresponding form factors become null) and others are created (requiring the recomputation of the corresponding form factors).

5.1. Re-computing the form factors in case of unchanged topology

To efficiently update the illumination when the complex remains unchanged, we must recompute the form factor F_{ij} only if i or j lies on the moving object O or if the visibility between i and j is obstructed by O. Using the complex, this means visiting only the faces with O in their label as well as the faces having in their boundary an edge which is part of the dual lines of O. All these faces are called faces *related to O*. By sweeping the complex along the dual lines of the object O, one can visit these faces *in time proportional to their number*. The dual lines of each object are directly accessible, thus eliminating unnecessary searches.

In order to recompute only the strictly necessary form factors, we apply for each face *related to* O, the algorithm described in Figure 13. Figure 14 shows an example of an object that has moved without a change in the topology of the visibility complex. We can see that even if the face associated with the edges of polygons a and b has changed, its topology remains unchanged. In order to recompute at a time $t+\Delta t$ the strictly necessary form factors associated with this face, we just have to consider the zone of interference associated with the point C_1 (region colored in dark gray in the figure).

Update of the form factors with the faces related to the moving object

Let f be a face *related to* a moving object O
For each face f *related to* O **do**
  If O is in the label of f
    Recompute $ff(s_l, s_r)$ for all pairs of mutually visible samples of f
    $\longrightarrow$ *See Figure 10 for more details about the computation of the form factors for a given face*
  Else If O appears in the upper list of edges of f
    Find edge e associated with O in $list_{up}$
    ZI_{up} = zone of interference associated with e
    ZI_{down} = zone of interference associated with the first obstructing point in $list_{down}$ such that:
      $ZI_{down} \cap ZI_{up}$ on O_l **or** the limits of ZI_{down} on O_l are under the limits of ZI_{up} on O_l
    Recompute $ff(s_l, s_r)$ for s_l and s_r such that:
      $s_l \in ZI_{up}$ and s_r has a part in $[p_{j_{up}}, p_{j_{down}}]$ with:
      $p_{j_{up}} = O_r \cap (p_{i_1}, p)$, where p is the point associated with e
      $p_{j_{down}} = O_r \cap (p_{i_0}, p)$
    $\longrightarrow$ *We proceed as in Figure 10 apart from the fact that we do not update* ZI_{up}
  Else // O appears in the lower list of edges of f
    Find edge e associated with O in $list_{down}$
    ZI_{down} = zone of interference associated with e
    ZI_{up} = zone of interference associated with the first obstructing point in $list_{up}$ such that:
      $ZI_{up} \cap ZI_{down}$ on O_l
    Recompute $ff(s_l, s_r)$ for s_l and s_r such that:
      $s_l \in ZI_{down}$ and s_r has a part in $[p_{j_{up}}, p_{j_{down}}]$ with:
      $p_{j_{up}} = O_r \cap (p_{i_1}, p)$, where p is the point associated with e
      $p_{j_{down}} = O_r \cap (p_{i_0}, p)$
    $\longrightarrow$ *We proceed as in Figure 10 apart from the fact that we swap*
      ZI_{up} *and* ZI_{down} *in the algorithm, and that we do not update* ZI_{down}
End For

Figure 13. *Algorithm for moving object (unchanged topology).*

More re-computations are necessary for updating the discontinuities while an object is moving. When a face is changed, the discontinuity mesh on its two associated objects is also partly modified. Therefore, some of the samples on the left and the right objects are also changed. We must then recompute the form factors between these samples and all the samples they see in all the faces associated with the left object (in the case of a sample of the left object) and in all the faces associated with the right object (in the case of a sample of the right object).

When visibility changes occur, local modifications must be done as shown in the next section.

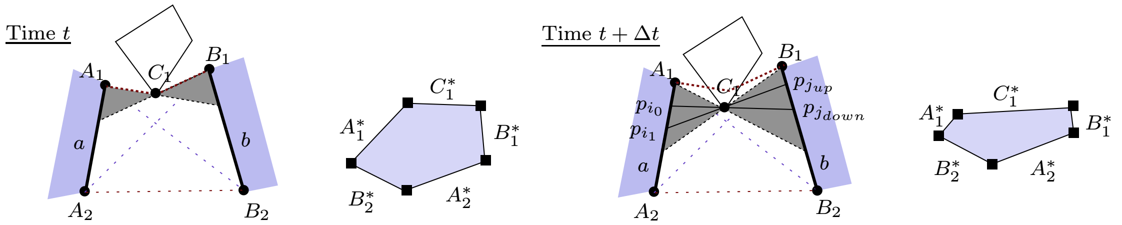


Figure 14. *Object moving without topological change in the visibility complex.*

5.2. Case of topological changes in visibility

When an object moves, it can induce visibility changes in the scene. It can hide two objects from one another, or it can make two previously hidden objects see each other. All visibility changes can be reduced to the elementary visibility change between three polygon vertices (and its opposite) shown in Figure 15: two polygon vertices l and r previously hidden by p become visible (and inversely). In the complex a new vertex and a new face are created while some edges are removed from some faces and inserted in other faces.

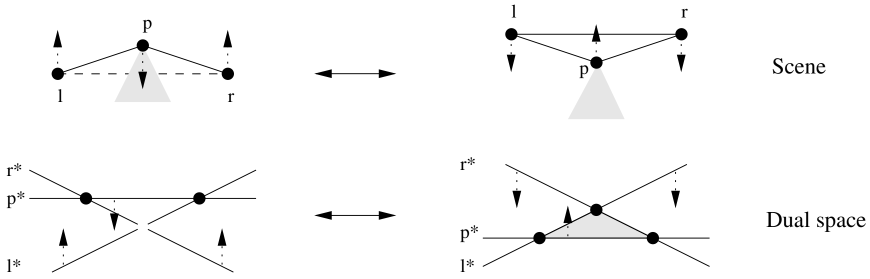


Figure 15. *Elementary visibility change.*

Nonetheless, the changes remain local and affect elements of the complex incident to the dual lines of the vertices of the moving polygon. Moreover, the complex can be updated *in constant time* once the involved triplet of vertices (l, p, r) is found: there is a constant number of cases involved (see Figure 16 which shows the modifications for one of these cases) and modifications are done in constant time for each case.

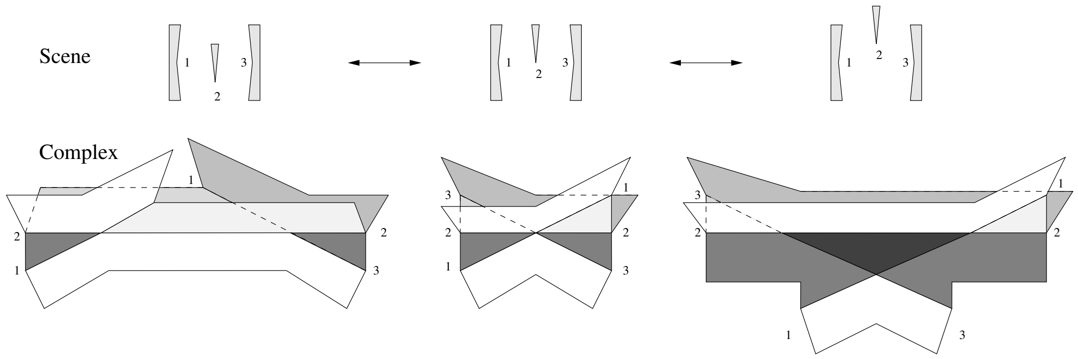


Figure 16. *An example of visibility change.*

When an object is moving, one must compute the topological changes in order. If the trajectory of the moving points is known, a priority queue is used to compute these changes in $O(\log m_O)$ time at each change (after an $O(m_O \log m_O)$ initialization step), where m_O is the size of the visibility polygon of the moving object O. When updating form factors, apart from the re-computation of those related to the moving objects (see section 5.1), we must set to zero form factors corresponding to the destroyed faces and compute new form factors corresponding to the created faces.

6. Conclusion

We have implemented a program which uses the visibility complex for the computation of the radiosity solution of 2D polygonal scenes. The visibility complex is very useful in this case, since it directly provides the discontinuity mesh and avoids unnecessary computation by considering only mutually visible parts between objects. It also provides an efficient method to compute the form factors. The current implementation only handles static environments, but we have studied the use of the visibility complex in the case of dynamic environments and described an efficient algorithm to update the form factors in the case of a moving object. The visibility complex allows us to identify and then to update only the strictly necessary form factors and should permit a rapid update of illumination in dynamic scenes.

We are currently working in this direction. The visibility complex seems to be adapted to the development of a test-bed environment for radiosity in dynamic environments. This will allow a better understanding of how the radiosity solution is affected in dynamic scenes (notably the efficient identification of exactly what must be recomputed), both in two and three dimensions. Such a study is a necessary and valuable step in the development of efficient algorithms well adapted to dynamic environments. A study of extensions to 3D of some of the work presented here is under way. Preliminary results have already been obtained.

Acknowledgements

The authors wish to thank George Drettakis for his constant support and his helpful insights and suggestions.

References

1. N. Holzschuch, F. Sillion, and G. Drettakis. An efficient progressive refinement strategy for hierarchical radiosity. In *Fifth Eurographics Workshop on Rendering, Darmstadt, Germany*, pages 343–357, June 1994.
2. S. J. Teller. *Visibility Computations in Densely Occluded Polyhedral Environments*. PhD thesis, UC Berkeley, 1992.
3. S. J. Teller and P. M. Hanrahan. Global visibility algorithms for illumination computations. In *Computer Graphics (SIGGRAPH'93 Proceedings)*, pages 239–246, August 1993.
4. M. F. Cohen and J. R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional Inc, 1993.
5. F. X. Sillion and C. Puech. *Radiosity and Global Illumination*. Morgan Kaufmann Publishers, Inc., 1994.
6. Daniel R. Baum, John R. Wallace, Michael F. Cohen, and Donald P. Greenberg. The back-buffer algorithm: An extension of the radiosity method to dynamic environments. *The Visual Computer*, 2(5):298–306, September 1986.
7. Shenchang Eric Chen. Incremental radiosity: An extension of progressive radiosity to an interactive image synthesis system. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH'90 Proceedings)*, volume 24, pages 135–144, August 1990.
8. David W. George, Francois X. Sillion, and Donald P. Greenberg. Radiosity redistribution for dynamic environments. *IEEE Computer Graphics and Applications*, 10(4):26–34, July 1990.
9. E. S. Shaw. Hierarchical radiosity for dynamic environments. Master's thesis, Cornell University, August 1994.
10. R. Orti, F. Durand, S. Rivière, and C. Puech. Using the visibility complex for radiosity computation. In *Proc. ACM Workshop on Applied Computational Geometry, Philadelphia*, May 1996. To appear.
11. P. S. Heckbert. *Simulating Global Illumination Using Adaptive Meshing*. PhD thesis, UC Berkeley, June 1991.
12. P. S. Heckbert. Radiosity in flatland. In *Computer Graphics forum (EUROGRAPHICS'92 Proceedings), 11:3*, pages 181–192, September 1992.
13. P. Schroeder, S. Gortler, M. Cohen, and Pat Hanrahan. Wavelet projections for radiosity. In *Proc. 4th Eurographics Workshop on Rendering, Paris, France*, pages 105–114, June 1993.
14. S. J. Gortler, P. Schroder, M. F. Cohen, and P. Hanrahan. Wavelet radiosity. In *Computer Graphics (SIGGRAPH'93 Proceedings)*, pages 221–230, August 1993.
15. M. Pocchiola and G. Vegter. The visibility complex. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 328–337, 1993.
16. M. Pocchiola and G. Vegter. Computing the visibility graph via pseudo-triangulation. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 248–257, 1995.
17. S. Rivière. Topologically sweeping the visibility complex of polygonal scenes. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages C36–C37, 1995.
18. H. C. Hottel. Radiant heat transmission. In W. H. McAdams, editor, *Heat Transmission*, chapter 4. McGraw-Hill, New-York, 3rd edition, 1954.

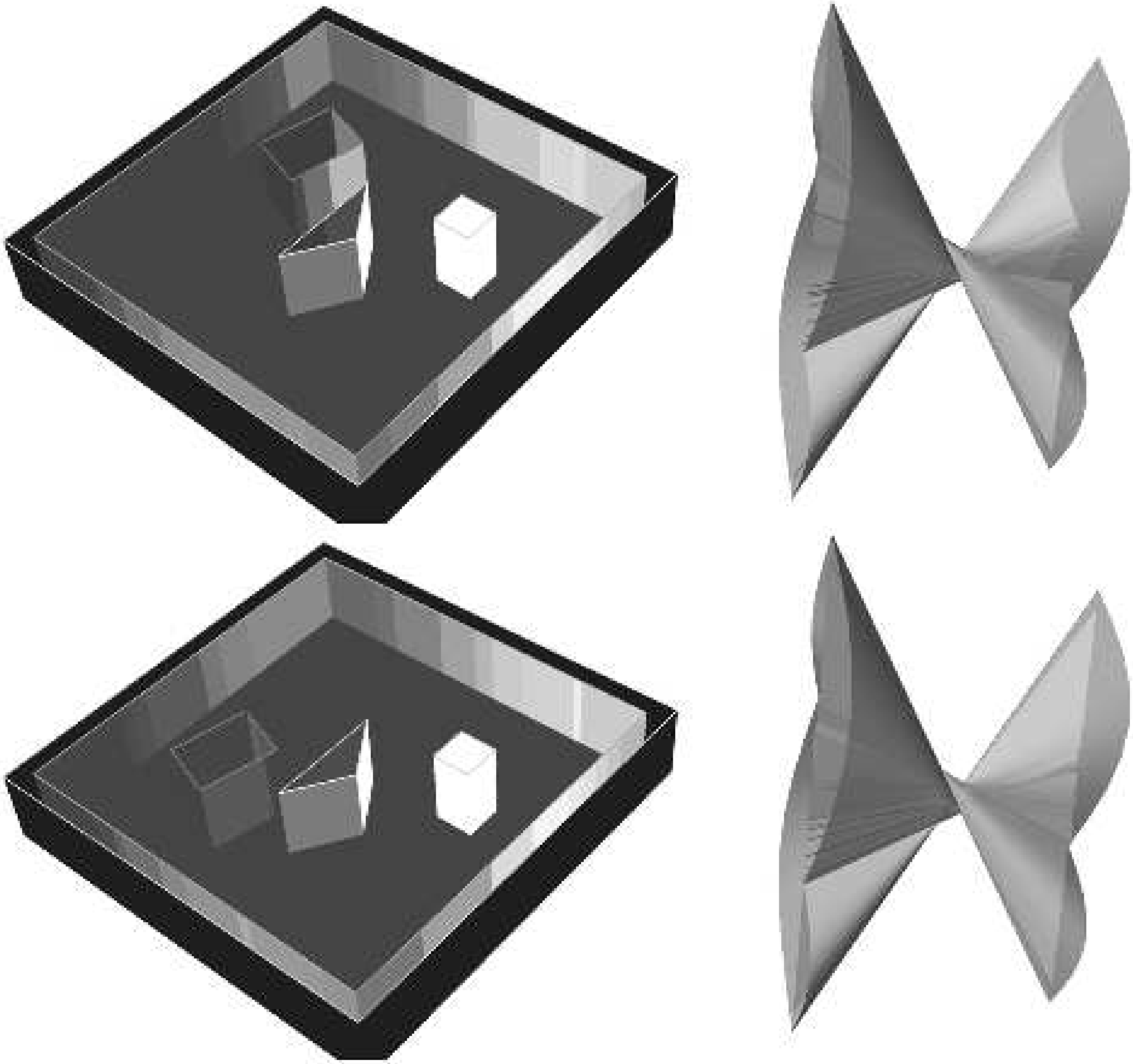


Plate 1: A simple example with three objects inside a room