The Close Object Buffer: A Sharp Shadow Modelling Technique for Radiosity Methods

Alexandru Telea

Abstract

Radiosity methods have been using several techniques in order to determine a subdivision of the environment that accurately captures the variations of the radiance field. Most of these techniques use object space approaches to predict the areas of high variation of the radiance. Such methods can be prohibitively expensive to be applied in scenes containing a large number of small 'detail' objects. An adaptive refinement technique addressing the problem of detail shadow detection has been devised and implemented. The presented technique is able to improve the quality of a radiosity image by rendering detail illumination like sharp shadows at an arbitrary precision with a small time and memory consumption.

1 Introduction

Radiosity techniques have become an important part of realistic image synthesis methods. Using a global illumination model, radiosity techniques attempt to provide an estimation for the radiance field in all the points of the tridimensional environment. Since there is an infinity of such points, the environment has to be discretized and the radiance field sampled in the discretization points. This is usually accomplished by a subdivision of the scene's surfaces into small elements. The radiance field is reconstructed by computing its values in each discretization point and interpolating them over the elements' surfaces. Several forms of interpolation can be used, the most common being Gouraud shading which corresponds to linear interpolation. The quality of the images produced by a radiosity renderer therefore strongly depends on the discretization of the environment into elements. Non uniform subdivision comes as a natural alternative: smaller elements will be used in areas of rapid variation of the radiance field and larger ones in areas where this field exhibits a slower variation in order to keep the element count at a minimum for efficiency reasons. The methods that attempt to automatically discretize an environment such that the above constraint is obeyed are generally known as adaptive subdivision or adaptive refinement techniques. Adaptive refinement techniques producing a good non uniform subdivision are generally quite time expensive. This paper presents a new technique for predicting the areas of rapid radiance variation, corresponding to sharp shadows and detail illumination.

The paper starts with an overview of the most used subdivision strategies (Section 2), presenting their advantages and limitations. Section 3 presents an analysis of the situations when sharp shadows are most likely to occur. Section 4 introduces the close objects buffer, a technique for detecting the presence of sharp shadows and performing a mesh refinement for capturing them. Sections 5 and 6 present the algorithms for building and using the close objects buffer and related accuracy and efficiency considerations. Section 7 presents a technique for speeding up close objects buffer computations. Some obtained results and additional conclusions are presented in section 8.

2 Overview of Subdivision Strategies

The quality and speed of a radiosity renderer is essentially influenced by the subdivision strategy used. It is difficult to find a subdivision that accurately captures the radiance field since this field is not known beforehand but is to be computed using the discretization itself. There are essentially two types of strategies for adaptive subdivision, based on the information used for detection of the zones of high radiance gradients.

The first type of strategies examine the gradient of an already computed solution over the current mesh [2]. The areas where a high gradient is found will be refined and, in case of a progressive refinement method, radiant flux will be reshot towards the newly created elements. The refinement process stops when the gradient of the solution over an element is below a desired threshold or the element is small enough for viewing purposes. This method is relatively simple to implement, requires no object space computations. The main disadvantage it has is that it can not guarantee that fine radiance variations will be detected if the initial mesh was too coarse to completely miss these details (the fine shadow of a pencil on a table may be completely missed if the initial mesh's elements are so large that they never intersect this shadow). This is essentially a sampling problem and the best gradient-based subdivision can do is to use an initial mesh fine enough for capturing the desired shadow detail level. This can result in oversampling some areas (the meshing effort is not directed in the areas of interest).

The most efficient strategies do not rely on the existing mesh when trying to predict refinement areas (are said to be *mesh-independent* or working in *object space*). Geometric information as the relative positions of light sources and illuminated surfaces is used in order to predict the areas of rapid radiance variation. Such a method is the *discontinuity meshing:* shadow boundaries (areas where a sharp variation of the radiance field is visible) are detected and the elements intersected by these boundaries are subdivided until they do not contain any *discontinuity* in the radiance value (i.e. they are not intersected by a shadow boundary) or they are considered small enough [Heckbert 1992], [1], [**?**]. Shadow boundaries due to primary light sources can be detected using a preprocessing step that shoots shadow rays from these light sources to determine where shadows will be the most likely to occur [4], [Campbell and Fussell 1990]. The disadvantage of discontinuity meshing is that full-fledged shadow casting algorithms working in object space can be very slow (especially if they are used for casting shadows from the secondary light sources as well and if the scene contains a large number of objects) and that keeping track of complex shadow boundaries crossing elements can be a very delicate process. A more serious liability is that shadow boundaries detected is a preprocessing step take into account only the primary light sources and therefore might not represent real shadow boundaries in the end, since reflection of light between objects usually creates a complex shadow pattern. Refinement based on preprocessing shadow computations might be therefore directed in areas where it is not really needed.

In conclusion a good environment meshing must capture the shading details as accurately as allowed by the maximum number of elements permitted by the user. This information can *not* be provided by the radiance solution computed so far but has to be obtained by other means. Shadow casting techniques involving the primary (and secondary) light sources may partially provide this information at rather high costs.

3 Detection of Sharp Shadows

As it was previously described, adaptive refinement strategies can not (and should not be used to) provide the refinement level required for capturing shading detail missed by the initial meshing resolution. This section presents a method for detecting the existence of such shading detail and for adaptively refining the mesh in the areas of interest.

There are two main reasons that cause visible illumination detail to appear over an area illuminated by a light source:

- the points on that area are unequally illuminated by an unoccluded light source. This happens if the lightsource and the illuminated area are facing each other at sharp angles and the distance between each other has a sufficiently large variation over their points. The variation of illumination over the receiver will be rather smooth and continuous. These are the so called 'smooth shadows'.
- the points of the illuminated area have different visibility terms with respect to the light source. This happens if the light source is occluded for some points of the receiver and not occluded for other ones. Since occlusion is zero or one, the variation of the illumination over the receiver may exhibit sharp, irregular transitions between neighbour points. These are the so called 'sharp shadows'.

Smooth shadows (smooth radiance variations more exactly) are therefore treated properly by adaptive refinement techniques using gradient criteria: areas exhibiting too sharp radiance variations are subdivided and the radiance is recomputed for the new elements.

Sharp shadows are mainly generated by partial occlusion of light sources. There is a high probability for a sharp shadow to occur over an element receiving radiance if:

- the receiver gets an important amount of the light source's flux. This implies that the light source *directly* illuminates the receiver (i.e. the angles between a source-receiver light ray and the source's and receiver's normals are small) and the light source is close to the receiver.
- the light source has a small area. The smaller the area, the sharper the shadow is. At the limit, a pointlike light source will create a shadow with no penumbra.
- there is an object between the source and the receiver, partially occluding the receiver. If this object is *closer* to the receiver, then it is very

probable that the shadow it will cast will be sharper. Therefore, the distance between the occluder and the receiver influences the sharpness of the shadow.

- the receiver's total reflectance (the sum of receiver's reflectivities for all used color bands $\rho_R + \rho_G + \rho_B$) is large enough such that its radiant exitance is sufficiently high for a sharp difference in illumination between the shadowed and the non shadowed areas to be visible (shadows are not well visible on a receiver with low reflectivity).

4 The Close Objects Buffer

The *close objects buffer* (or the COB for short) is a radiosity software tool for progressive refinement methods that attempts to solve the problem of detection of areas where a sharp shadow can occur. As we have seen, there is a high probability of sharp shadow appearance when a receiver (an element) is partially occluded from a light emitter by an object which is very close to it and when the emitter casts a strong illumination over the receiver.

For each element in the environment, a COB will be used to store references to all the objects that are above and close to the visible surface of the element. Practically, this will be an item buffer storing references to the polygons that are close to the element's visible surface: The COB of an element can be in

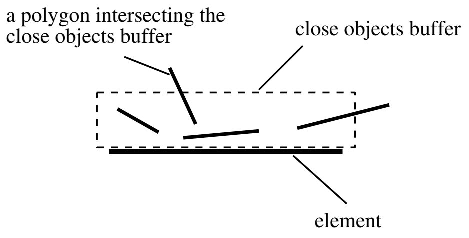

Figure 1: An element and its close objects buffer

three states: uninitialized (no buffer-object intersections have been performed yet), empty (the buffer contains no objects) and not empty (the COB contains at least an object) Initially, all elements have an uninitialized buffer. When a light emitting element is about to shoot radiosity to a receiving element, the emitter is firstly checked to see if it potentially could cast a sharp shadow over the receiver. A set of criteria including light source's power, distance to the receiver, mutual orientation of receiver and emitter are used to estimate the interaction strength. If the criteria succeed, then the receiver's COB is checked: if it is uninitialized, then the buffer is built now. If the COB is empty, then there aren't any objects that might cast sharp shadows over the element, so the emitted radiosity is received by the element. If the buffer is not empty, then there are objects partially occluding light coming from the source and potentially casting sharp shadows. The element is then directly subdivided (without shooting at it anymore) and the objects held into its buffer are distributed in the buffers of the newly created elements. The close objects buffer proves itself efficient in two situations:

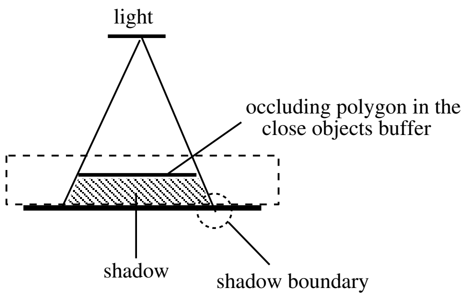

Figure 2: Shadows created by occluding polygons in an element's close objects buffer

- small objects are located on or nearby the surface of large objects (like pencils or cups on a table top). It is very likely that the initial meshing of the large object (the table top) will be coarse enough such that it won't be able to capture the small but sharp shadows cast by the objects placed on its surface (like the pencils and the cups). The close objects buffer of the table's elements will detect the presence of the close objects and start the adaptive sudivision that will ultimately capture the detail shadows.

- arbitrary objects that have common edges (like two walls sharing an edge or the inside faces of an open box). If the initial meshing of such objects is coarse, it is very likely that the subtle shadows appearing sometimes near an edge shared by two faces will be missed. Again the small objects buffer will detect such situations and initiate a local mesh refinement over the area close to such edges.

The COB is unique per element: it stores all the polygons being in the proximity of its surface independently on the light sources. The reason for this is that for objects very close to a surface it is assumed that they will cast a *sharp* shadow if illuminated by *any* light source above that surface. This is true in most cases since the distance receiver-occluding object is much smaller than the distance source-occluding object (the buffer's height is very small compared to the usual inter-object distance). The cases when this assertion doesn't hold will be treated separately.

The COB is not built by default, in a preprocessing stage (as the one used for some shadow-detection algorithms) but rather on demand, when there is a high probability for an emitter to cast a sharp shadow over a receiver in the presence of occluding objects close ot the receiver. Many elements will never be in such a situation so their COBs will remain uninitialized (therefore the memory usage increase is negligible for them - an empty buffer is 5 bytes per element in the current implementation). Building the COB on demand will also save computing time.

5 Building the Close Objects Buffer

The buffer will contain, as previously described, references to polygons being in the proximity of an element's visible surface. Geometrically speaking, the buffer is a prism having the base a bit larger than the element and a given height δ computed as a small fraction of the buffer's base edge. In the simplest case, one can use a simple rectangular box placed on the element's surface (similar to the hemicube used in form factor determinations)(see figure 3).

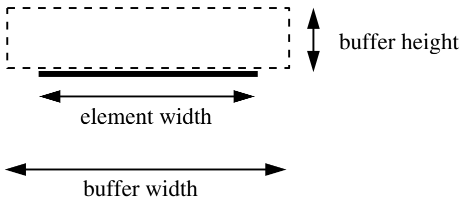

Figure 3: Sizes of the close objects buffer

The box has been made a bit larger than the element itself in order to trap also the polygons located in the vicinity of the element. This is especially useful for elements near an edge shared by two polygons. These are typical cases when a shadow strip occurs and they are directly detected using the close objects buffer (see figure 4).

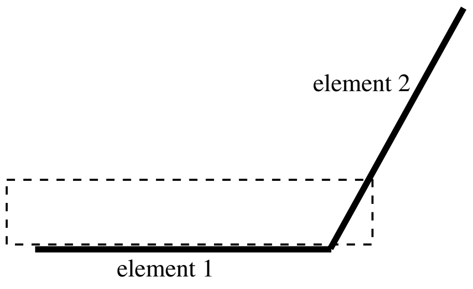

Figure 4: Two elements sharing an edge: element 1 is in element 2's close objects buffer and conversely

As described previously, the buffer holds all polygons that potentially cast sharp shadows over the receiver. A simple approach would be to store in the buffer all polygons intersecting the element's box. However not *all* the polygons intersecting this box will cast sharp shadows over the element, so a more elaborate strategy uses use several criteria to determine the likelihood of having a sharp shadow.

A first important tests is concerned with the *total occlusion criterion*: a polygon that covers the whole box (i.e. intersects all the box's edges that are normal to the element's surface) will practically occlude the whole element rather than cast a sharp shadow on it. In this special situation the element beneath can be skipped during the shooting process: since there's practically no chance for a light source to reach it, there will be no shooting towards this element anymore. This can save an important amount of time: consider, for example, the case when a table is covered with several sheets of paper. Most of the elements of the table are invisible from a light source placed above the table since they are completely occluded by the paper sheets. There will be just one attempt to shoot at them and then they will be found to be totally occluded. (see figure 5).

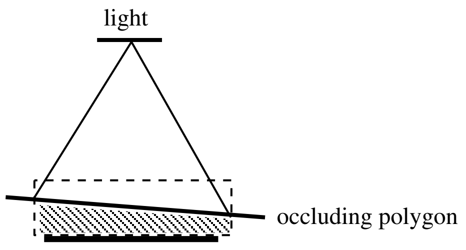

Figure 5: Totally occluded element

The total occlusion criterion may be invalid in two cases:

- if the light source illuminates *tangentially* the element, then the element might not be totally occluded from the light even though its close objects buffer is totally occluded as seen from above. This situation is handled by default since a tangential illumination that might creep under the occluding polygon will probably influence very little the element's final radiant exitance, so it can be safely ignored (figure 6).

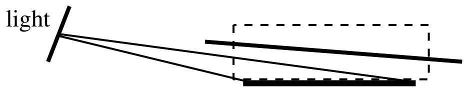

Figure 6: Tangential illumination of a *totally occluded* element

- if there exists a *primary* light source intersecting the close objects buffer of an element that is totally occluded, this lightsource might be placed *below* the occluding polygon. Therefore this light source will not be occluded when illuminating the element so we can't skip shooting at this element (figure 7).

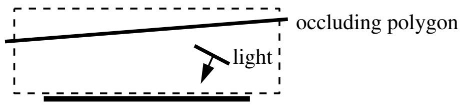

Figure 7: Light illuminating under a *totally occluded* element

- if the buffer is totally occluded and there's no primary light source inside as described above, then we practically don't store anything in the element's buffer but just mark it as 'totally occluded'. This saves memory and time and also is consistent with the fact that a totally occluded element will never need to be further subdivided. Another good alternative would be to remove this element out of the polygon's elements list so that the renderer will never try to shoot at it in the future, therefore saving a certain amount of time.

The element can be marked as totally occluded only in the case there doesn't exist a light source in its close objects buffer or in the case such a source does exist but it is oriented in such a direction that it doesn't illuminate *directly* the element. The reason for this is that a such light source illuminates indirectly the polygon, hence it has a smaller chance of casting a sharp shadow than a directly illuminating source.

The above tests are very simple and can be done when the close objects buffer is built for an element with virtually no time penalty.

In the case the total occlusion criterion fails but a polygon still intersects the small objects buffer of an element, this polygon will be regarded as a potential shadow caster and will be added to the element's buffer. There are a few cases when such polygons do not really cast a sharp shadow over the element but they are still added to the buffer. A subdivision process will start even though it will ultimately not be necessary. Since these cases are statistically very few (the time penalty being paid due to the unnecessary subdivision being consequently very small) we can refrain from devising special tests to reject them.

Summarizing the above:

- the small object buffer is built only on demand, when there is a high probability of a sharp shadow to occur on an element.
- the buffer can be modelled with a rectangular box similar to the hemicube placed on the top of an element, with the height proportional with the element's edge and the width slightly larger than the element's. A polygon is said to be in the small objects buffer if it intersects this bounding box.
- a special function of the buffer is to detect cases of total occlusion of elements from all light sources. Such elements are completely excluded from the flux gathering process, resulting in a speed increase.
- at an element's subdivision, the small object buffers of the new elements can be easily computed out of the original buffer.

We can say that the construction of the buffer attempts to detect the possibility of a sharp shadow from the *receiver's point of view.*

6 Using the Close Objects Buffer

The close objects buffer is used within the progressive refinement process, when an element is selected to receive radiant flux. There are several steps performed during this process:

STEP 1: The buffer is first checked to see if it is totally occluded or not. A positive answer prevents shooting to a totally occluded element from *any* light source, thus saving several expensive form-factor and occlusion test computations. Note that this result can not be obtained with a usual preprocessing shadow detection method which only detects shadows cast by the primary light sources.

STEP 2: Depending on the estimated interaction between the shooter and the element, the buffer will be used or not. Recalling the necessary conditions for having a visible sharp shadow on the receiver, we can evaluate:

$$\Delta M_i = (\rho_{R_i} + \rho_{G_i} + \rho_{B_i}) \frac{\cos\theta_i \cos\theta_j}{\pi r^2} A_j \tag{1}$$

where i is the receiver element and j the shooter. ΔM_i will be an approximative estimate of the increase in radiant exitance of element i due to the shooter. This increase can be compared to the actual radiant exitance of the element M_i to determine if it is large enough to produce a visible shadow. Alternatively the fraction of shooter j's flux reaching element i can be evaluated: if it exceeds a certain percentage of shooter's total shot flux, then there's a high probability that a shadow over element i will indeed be sharp. This is justified by the fact that the shooters are picked in decreasing order of their unshot fluxes in the progressive refinement method, hence a large fraction of a shooter's flux is indeed capable of creating a sharp shadow.

Together with this criterion, the element's size is checked as well. If the element is too small, we ignore the buffer and shoot to it as usually. It is important to remark that the stop threshold for the subdivision initiated by the previous criterion is independent on the stop threshold used for the gradient-based refinement: the small objects buffer subdivision attempts to capture *sharp* shadows over rather small areas, so the minimum element size can be sensibly smaller than the one allowed for the gradient-based subdivision. This allows using a rather large element size stop threshold for the gradient criterion and a much smaller one for sharp shading detail detected by the close objects buffer.

STEP 3: If the criterion has decided that the interaction is strong, the buffer is checked to see if it has been built or not. If it has been built and it is not empty, then there are close objects to this element, therefore potential sharp shadows. The element is simply put aside to be subdivided at the end of the current iteration. If the buffer has been built but it is empty, then there aren't close objects to this element so the shooting part takes place. If the buffer hasn't been built, then this is done at this moment and step 3 is reexecuted.

Summarizing the above:

- a criterion estimating the strength of the interaction between shooter and receiver is used to determine if there's a high probability of having a sharp shadow. If not, the element will receive radiosity as usually.
- a non-empty buffer will trigger element subdivision. This subdivision is different from the one initiated by the gradient-based refinements and attempts to accurately capture the sharp shadow boundaries over the element.
- the close objects buffer subdivision and the gradient-based subdivision are independent processes; although both try to refine the mesh, they are based on different criteria, have different stop thresholds and practically perform their best in different areas of a scene.

7 A Two-Level Close Object Buffer

The management of a close objects buffer is simple but it has to be done for each element in the scene, each time that it has to receive radiant flux from a shooter. If the number of elements is large this process can be slow: building a buffer involves, in a brute-force approach, intersecting *all* polygons in the scene with it. Although these intersections have been significantly speeded up by using an octree built for form factor ray tracing purposes (the polygons potentially intersecting an element's close objects buffer are a subset of the polygons contained in the octree cells over which that element spans), the method presented in the following can reduce the computational time even further.

There exists a coherence in the set of close objects for the elements of a polygon which can be expressed by the fact that neighbouring elements tend to have a similar set of close polygons - that is, a similar close objects buffer. In particular there usually exist large areas over the polygons of a scene for which all the elements have an empty close objects buffer.

We can take advantage of this coherence by introducing a COB for each polygon in the scene. This COB will have the same semantics as an element's buffer: it stores all the polygons close to the visible surface of that polygon. Polygons' buffers are built on demand, the first time we shoot at such a polygon, exactly like for the elements' buffers. Such a two-level hierarchy of close objects buffers can speed up the whole process significantly, especially for scenes containing a large number of polygons.

Building an element's buffer is now very fast: if the element's polygon has an empty or totally occluded buffer then *all* its elements will have empty or totally occluded buffers respectively. If the element's polygon's COB is not empty then its elements build their buffers by testing only the polygons in their polygon's COB. This is much faster than testing all polygons in the scene and can be faster than using an octree.

Using the close buffer is now speeded up as well: if a polygon has an empty or totally occluded buffer then we can skip testing all its elements' buffers at once. The buffers of a polygon's elements will start to be used only if the polygon's buffer is not empty and not totally occluded.

This two-level scheme can be extended to include a larger number of levels, similarly to the hierarchical radiosity method presented by [3].

8 Conclusions

Adaptive gradient-based mesh refinement can provide only a smoothing of the radiosity solution over a given environment but typically fails to detect sharp detail shadows. Discontinuity meshing based on shadow casting gives better results but at a much higher expense and is quite complicated to be implemented. A strategy integrating directly in a progressive-refinement radiosity method has been presented, featuring the possibility of detection of detail shadows cast by occluding objects placed in the immediate vicinity of receivers. The presented method is virtually independent on the initial mesh's resolution (works entirely in object space), requires small amounts of memory and imposes a very small time penalty on a normal radiosity method. It doesn't need any preprocessing step, is applied only on demand and it can integrate and cooperate very well with other adaptive refinement techniques. The method also detects totally occluded elements, eliminating them from the radiance computations and speeding up the radiosity process. Furthermore, the user can easily control the maximum subdivision level, being able to choose the one matching the available time and memory resources. A two-level hierarchical variant of the close objects buffer offering an important speed improvement for large scenes has been presented.

The method has been practically incorporated in a progressive refinement radiosity renderer using gradient-based adaptive subdivision. The noticed time penalties were less than 2% of the total rendering time. The first iterations of the progressive refinement were perceivably slower since most of the elements' close objects buffers were not yet built. After a few iterations, most buffers get computed so the process practically reaches its normal speed. Tuning the buffer's height parameter can strongly affect the level of refinement: a higher buffer will presumably contain more polygons so the chance its element gets subdivided due to a non empty buffer will increase accordingly.

The improvements in the rendered images are quite visible. Detail shadows and illumination are now detected and captured appropriately around small objects placed on large surfaces and around edges shared by two large surfaces.

References

[1] M. F. Cohen and J. R. Wallace. *Radiosity and Realistic Image Synthesis.* Academic Press Professional, Cambridge MA, 1993.

[2] M. F. Cohen, J. R. Wallace, Chen S. E., and D. P. Greenberg. *A Progressive Refinement Approach to Fast Radiosity Image Generation.* SIGGRAPH, 1988.

[3] P. M. Hanrahan, D. Salzman, and L. Auperle. *A Rapid Hierarchical Radiosity Algorithm.* Computer Graphics (SIGGRAPH '91 Proceedings vol 25 no 4), 1991.

[4] T. Nishita and E. Nakamae. *Continuous-Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflection.* SIGGRAPH, 1985.