Chapter 3

Face Cluster Radiosity

3.1. Introduction

The best hierarchical radiosity methods, using volume clustering, permit scenes of moderate complexity (several hundred thousand input polygons) to be simulated in times ranging from ten minutes to a few hours, depending on the level of accuracy chosen for the simulation, and the geometric arrangement of the scene. Unfortunately, current radiosity techniques, even with clustering, use excessive memory and their speeds are not competitive with other, less realistic rendering methods. We would like to be able to apply radiosity methods to the complex scenes common in special effects. Such scenes routinely use objects each employing 100,000 polygons or more. We therefore seek an enhancement to the hierarchical radiosity algorithm that will permit very complex scenes—scenes with millions of input polygons—to be economically simulated on a standard computer.

One of the greatest difficulties with existing radiosity methods is that their memory use is at least linear in the number of input polygons. This is not a problem if the scene is small, but if the input polygons cannot fit in physical memory, the algorithm will thrash and performance will degrade dramatically. To deal with very complex scenes, we need methods which in practice have memory and time cost that is sub-linear in the number of input polygons. This is possible because for many such scenes, the geometric detail represented by the input polygons is much greater than that needed for the radiosity simulation.

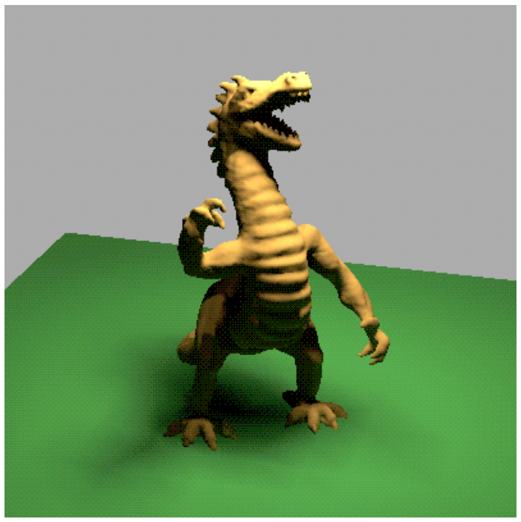
In this chapter I describe the core elements of the face cluster radiosity algorithm, a technique that achieves this goal. This algorithm grew out of research into adapting the radiosity method to use multiresolution models. Later chapters will deal with technical details and some of the assumptions behind the algorithm, but for now, I will focus on its overall structure.

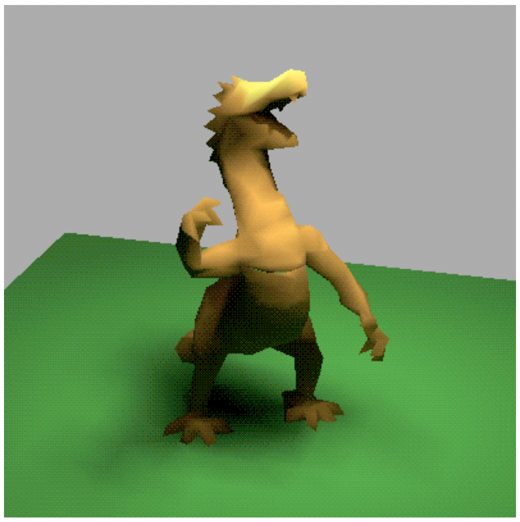
3.1.1. Overview

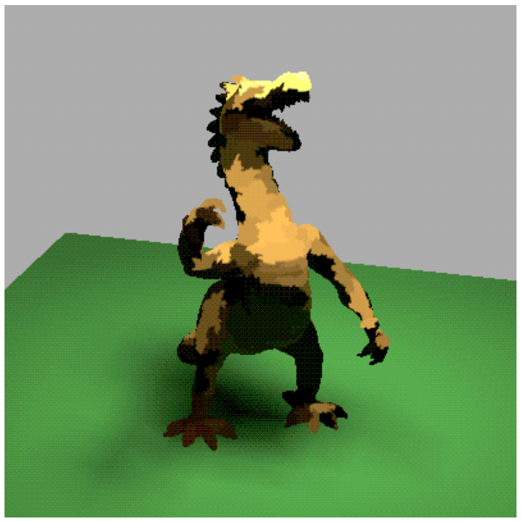
A preview of the technique is shown visually in **Figure 19**. If one of the best existing radiosity algorithms (hierarchical radiosity with volume clustering) is used on a detailed model, a solution takes over ten minutes (**Figure 19**a)[1]. If, on the other hand, the input geometry is simplified by cutting the number of triangles by a factor of 100, and the same algorithm is applied to the simplified model, a solution can be calculated much more quickly (**Figure 19**b, 7 seconds). This is fast, but the accuracy and visual quality are poor. The face cluster radiosity technique allows a solution not much more expensive than this to be calculated and propagated to the fully detailed model, yielding **Figure 19**d. This is much faster than the full solution and almost as accurate.

The three main phases of the algorithm are preprocessing, solution, and postprocessing. Preprocessing converts the scene description into a multiresolution, hierarchical model. The time cost of this is super-linear in the number of input polygons, but preprocessing can be done on an off-line, object-by-object basis, so its memory costs are modest and its time costs can be amortized over multiple solutions. Next, one or more radiosity solutions are found. This is the costliest step, in practice. The solution phase is sub-linear in cost because it accesses only the coarsest levels of detail from the hierarchy that are necessary. Consequently, often large portions of the hierarchy need never be paged in during this phase, with huge physical memory savings. After solution, postprocessing evaluates the radiosity of the finest details of the scene. This requires linear time. The overall cost, being dominated by solution, is thus sub-linear in practice.

1. These times are highly dependent on accuracy settings, platform and implementation. The times quoted are meant to be illustrative, though care was taken to use similar settings for each algorithm.


(a) HRVC, 108,000 triangles, 707s


(b) HRVC, 1000 triangles, 7 s


(c) Constant FCR, 108,000 triangles, 7s

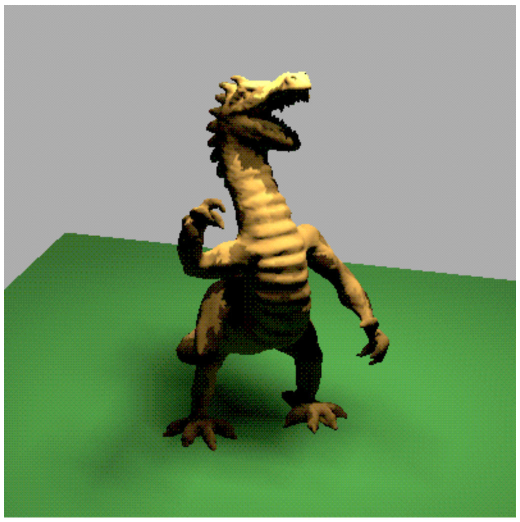

(d) Vector FCR, 108,000 triangles, 8s

Figure 19: Radiosity on a detailed dragon model. Face cluster radiosity (FCR) and hierarchical radiosity with volume clustering (HRVC) algorithms applied to a detailed dragon model.

3.2. Outline

In this section we present the ideas which lead to the development of the face cluster radiosity algorithm, and a rough sketch of the algorithm itself. We close with a more thorough analysis of the complexity of the new method and its relationship to previous work on hierarchical radiosity algorithms.

3.2.1. Motivation

The current state of the art in radiosity methods, hierarchical radiosity with volume clustering [Smit94], has a complexity that depends, among other things, at least linearly on the number of polygons in the input scene. The work presented in this chapter has a number of motivations, but the primary motivation can be summarised very simply:

> If we increase the number of input polygons in our scene by tessellating existing polygons[1], it should have no impact on the speed, memory use, or results of our radiosity simulation.

That is, we would like the performance of our radiosity algorithm to be *sub-linear* in the number of input polygons. This should hold if we simply tessellate polygons in the scene, but we would also like it to be largely true if we add small-scale detail to surfaces, perhaps by perturbing the tessellations slightly, that is unlikely to affect the broader radiosity computation. In other words, high-resolution detail in a scene should have minimal impact on the radiosity solution time.

This motivation arose from experiments with radiosity simulations on scenes that contained large scanned models; large enough to make the dependence on input polygons of the volume clustering system a problem. The observation that most of the polygons in such models are for high resolution detail, and don't affect radiosity computations much, led me to investigate the use of model simplification in radiosity. Model simplification is a process whereby the polygon count of an existing model is reduced, while trying to keep the same approximate shape as the original model [Heck] (see **Figure 20**). In particular, it lead to the use of multiresolution models [Hopp97].

A multiresolution model allows the resolution of an existing polygonal-mesh model to be lowered by different amounts at different points on that model. It is possible to have one region of the model be heavily simplified, and another

1. Tessellating means replacing an existing polygon with several smaller polygons that cover the same area.

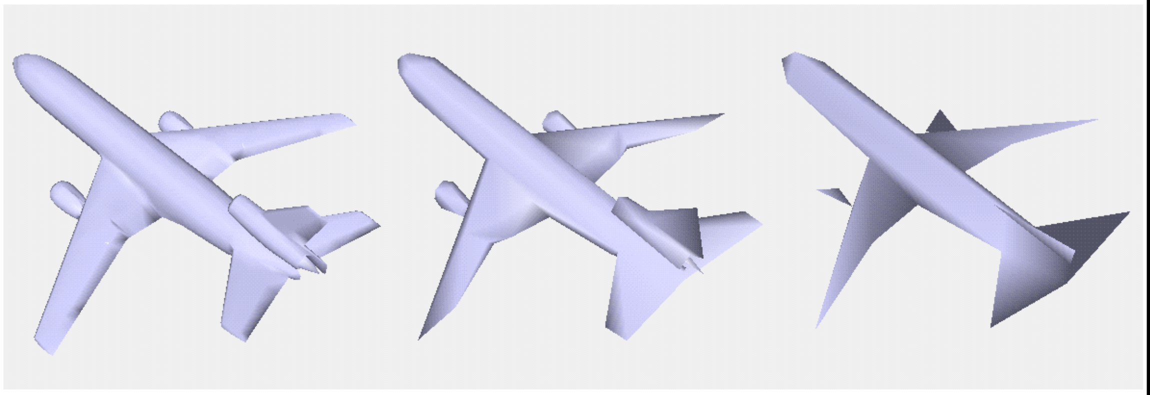


Figure 20: Progressively simplified models of a DC10 aircraft.

region simplified almost not at all, with the intermediate regions having corresponding gradations of resolution. Such models are generated by applying an iterative simplification process to the original model, and generating a corresponding tree of simplifications, with the original model's polygons at the leaves of the tree. Any cut through this tree gives a version of the model with a particular resolution.

Intuitively, we can envisage using such a model to remove the unneeded high-resolution detail of models before running our radiosity algorithm. Instead of running our radiosity simulation on a detailed model (as in **Figure 19**a), we would run it on a simplified model (**Figure 19**b), and then somehow apply the results back to the original model.

A better solution would be to combine both the multiresolution model and radiosity algorithms. By using the multiresolution hierarchies of the models directly in the element hierarchy of the radiosity solver, we could adjust the model resolution "on the fly" to match that needed by the radiosity algorithm. Apart from being more flexible, this would ensure that no manual selection of simplification level was necessary. This is vital, because it is difficult to guess a priori where the models can be simplified, as these decisions are best made using global information. To do a good job, we would need the results of the radiosity simulation we are trying to run.

Traditionally, below the level of the input polygons we add refinements to those polygons to capture any illumination detail that is at a higher resolution than those polygons, as shown in **Figure 21**. If we envisage a radiosity hierarchy where the root of the hierarchy represents the entire scene, and the input polygons lie at some particular level in the hierarchy, refinements can be represented by nodes below that line in the hierarchy.

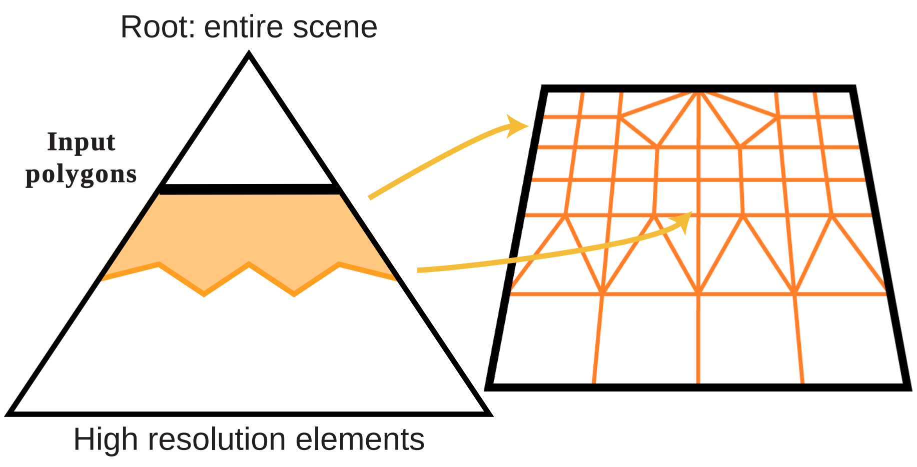


Figure 21: Refinement of polygonal elements in the detail hierarchy

In the same way, **Figure 22** shows how in turn we can use simplification to reduce the resolution of the scene above the input polygons; conceptually, these simplifications make up the nodes above the input polygons in the hierarchy. This also has the benefit of improving our representation of the scene at such levels. Volume clusters, which are traditionally used to represent versions of the scene coarser than the input polygons, are designed to approximate a cloud of unconnected polygons. For the connected, largely smooth surfaces that occur in many models, surface simplification provides a better and more natural approximation.

The particular multiresolution model format we use in our algorithm, as discussed in the next section, is called a face cluster hierarchy. **Figure 23** outlines how these face clusters fit into a traditional hierarchical radiosity with volume clustering algorithm. As indicated, they help bridge the gap between the surface

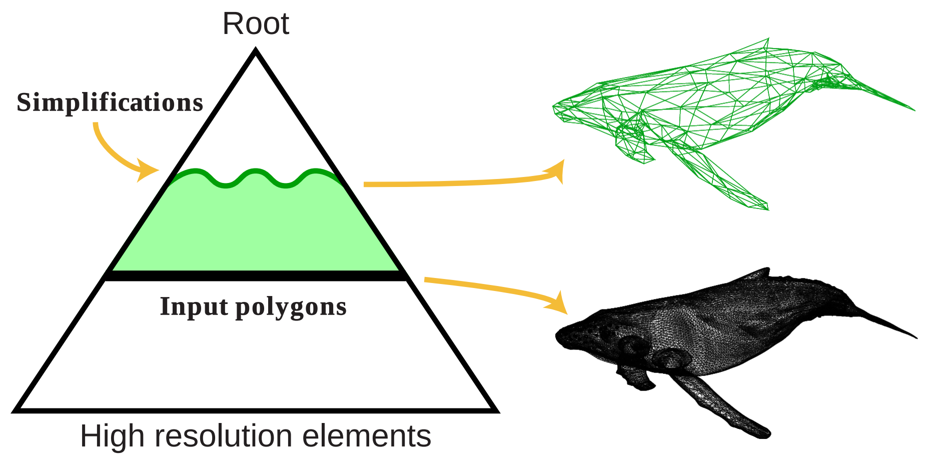


Figure 22: Simplification of polygonal elements in the detail hierarchy

representation of polygons, which are contiguous and have a single orientation, and the representation of volume clusters, which is best at modelling a collection of scattered, disconnected surfaces. Face cluster nodes are intended to model a contiguous, connected chunk of the surface, which is assumed to be largely co-planar, but to have varying surface normals.

With both a vanilla hierarchy, **Figure 23** (a), and with volume clustering (b), the solver must descend to at least the level of the k input polygons. This level gets ever further away from the root (as $\log k$ increases) as the detail of the scene is increased. That is, as we arbitrarily tessellate the scene, the solver must do more work. Once face clusters are introduced in (c), this is no longer the case; the solver must only descend as far as the leaves of the solution hierarchy, which remain at a constant level regardless of the tessellation of the underlying models. This has the further advantage that we need not store radiosity coefficients for all input polygons — only for those face cluster nodes within the solution hierarchy. In a simulation where a detailed model can be safely approximated for the purposes of radiosity computation by a much simpler model, this provides tremendous memory savings.

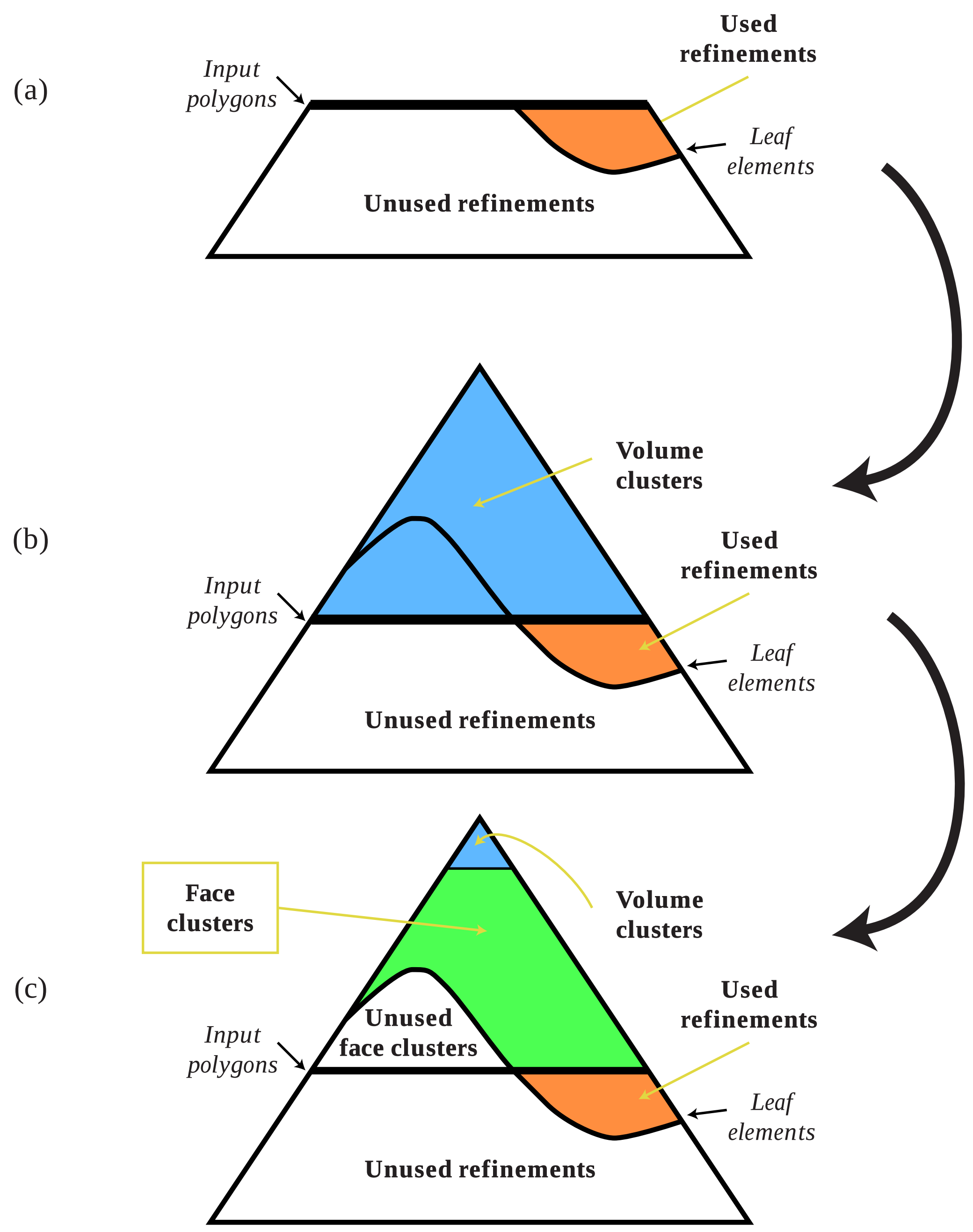


Figure 23: Adding face clusters to the hierarchy. The original polygon-only solution hierarchy (a) can be augmented with in turn (b) volume clusters and (c) face clusters.

3.2.2. Hierarchy Analysis

Figure 24 is a schematic comparison of variants of hierarchical radiosity on a scene with two large polygons A and B in close proximity, and eight small polygons C-J, more distant. Simple hierarchical radiosity (a) yields a forest of quadtrees. Polygons A and B are subdivided and some of their children are linked. The large number of links between the small input polygons C-J makes the algorithm inefficient. **Figure 24**b shows hierarchical radiosity with volume clustering. If cluster Q is sufficiently small and distant from cluster P then a single link between them suffices. Since a cluster can illuminate itself, a self link on Q is necessary as well. The algorithm is still slow because it is necessary to push light down to polygons C-J.

Face Cluster Radiosity. We propose that volume clusters be replaced by multiresolution models for all groups of input polygons that represent a surface. The use of such models allows pushing of light to the leaves to be avoided, and they often provide a better fit to the original surfaces than volume clusters. The particular multiresolution representation that we use, face clustering, groups adjacent faces that have similar normals, and thus can approximate largely planar surfaces well.

With this scheme, the data structures for elements coarser than and finer than the input polygons are more similar, since both face clustering and quadtree refinement yield a contiguous piece of surface with a normal. Hierarchical radiosity is now free to subdivide below the level of the input polygons and to “unsubdivide” above this level. This reduces the hitherto inordinate role of the input polygons, permitting hierarchical radiosity to represent light transport at more natural levels of detail, and to operate more efficiently in complex scenes.

The use of multiresolution models improves the accuracy of our representations and permits our algorithm to avoid touching the lowest portions of the hierarchy during iterations. In **Figure 24**c we see how a tree of simplified models is built above the input polygons. The tree nodes below T and above C-J are now face clusters, not volume clusters, while the highest levels of the tree, above connected objects, are volume clusters. Note that the subtree below T can be paged out during simulation, saving time and memory.

3.2.3. Previous Work

The use of simplified models in radiosity has previously been proposed. Rushmeier et al. demonstrated the feasibility of the concept, but their method employed manually-constructed simplified models, making it impractical for

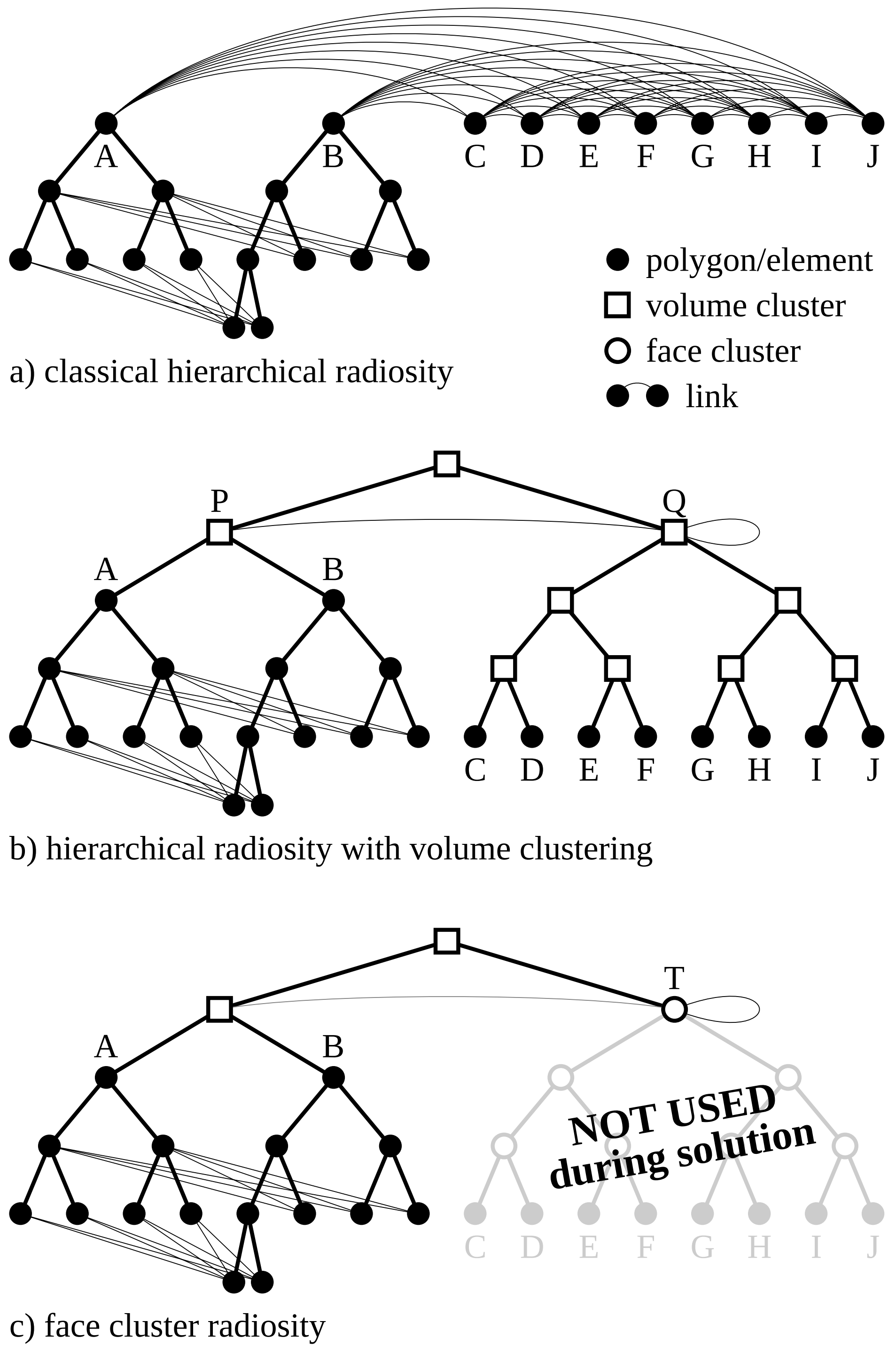


Figure 24: Three approaches to hierarchical radiosity

complex scenes [Rush93]. Greger et al. showed how a radiosity simulation of a simple scene could be applied to a more detailed version of that same scene by using an *irradiance volume* [Greg98]. Both these methods require the user to judge the level of pre-simulation simplification; simplification and simulation are two separate steps, whereas in our algorithm, the level of simplification is driven by the radiosity simulation. That is, the simulation picks the level of simplification appropriate to each transfer of radiosity between solution elements, and multiple levels of simplification exist during the solution.

3.3. Building Surface Hierarchies

3.3.1. Multiresolution Surface Hierarchies

Recent work from the area of surface simplification provided a starting point for our research into radiosity using multiresolution models. Iterative edge contraction, one of the currently popular simplification techniques, can be used to construct a hierarchy of progressively larger vertex neighbourhoods on the surface. Each edge contraction collapses the two vertices at either end of the edge to a single, new vertex. The hierarchy is created by treating the endpoints as the children of this new vertex. These vertex hierarchies have been used primarily for view-dependent refinement of models for real-time rendering [Hopp97, Xia96, Lueb97].

In the original version of the algorithm presented in this dissertation, I used vertex hierarchies directly, as implied by the preceeding section. This sometimes proved problematic, because vertex nodes don't have a well defined area and normal. (These properties are easier to establish for face hierarchies than for vertex hierarchies.)

To address this problem, the initial algorithm treated vertices in the simplification hierarchy as faces on the *dual graph* of the model. That is, conceptually the polygonal mesh was assumed to define the Voronoi diagram of the actual surface used by the radiosity simulation. Both the area and the normal for each vertex were constructed from its surrounding faces; the dual was never explicitly formed.

This approach still had problems, not least of which was that the radiosity solution was being performed on what amounted to a smoothed version of the original model, which made the treatment of creases and sharp edges on the model problematic. Also, without explicitly constructing the dual graph, a step which would have been prohibitive in cost, the vertex nodes did not have a well-defined extent. This made calculating form-factors and good visibility estimates difficult.

At this point the following critical insight presented itself: a more robust approach to the problem would be to use the original faces of the model in the radiosity solution, and instead have the simplification algorithm work on the dual graph of the model.

To meet the need for such an algorithm, Garland developed *face cluster hierarchies* [Garl99], a dual form of the quadric-based simplification algorithm of Garland and Heckbert [Garl97]. (It is also closely related to the “superfaces” simplification algorithm of Kalvin and Taylor [Kalv96].) Rather than iteratively merging pairs of vertices to directly simplify the mesh, the algorithm iteratively merges groups of topologically connected faces, which we refer to as face clusters, thereby partitioning the original mesh (see **Figure 25**). This process does not change the original geometry of the surface in any way; it merely groups surface polygons into progressively larger clusters. These merge operations can be used to create a multiresolution hierarchy in the same manner as edge collapses; an example is shown in **Figure 26**. The resulting *face cluster hierarchy* has the property that any cut across the hierarchy gives a particular partitioning or clustering of the original model. **Figure 27** shows an example hierarchy, and the corresponding clusters on the model for several different cuts.

In Garland and Heckbert’s original paper [Garl97], quadric functions of position, $\mathbf{v}^T\mathbf{A}\mathbf{v} + 2\mathbf{b}^T\mathbf{v} + c$, are used to represent the set of planes associated with a vertex node, and can be used to find the best-fit point to those planes. When we are working in the dual space, we instead use quadric functions of the face normal, $\mathbf{n}^T\mathbf{A}\mathbf{n} + 2\mathbf{b}^T\mathbf{n} + c$, to represent the set of vertices in a cluster node, and to find the best-fit plane to those points. Because of this, it can be shown that by applying the quadric error approach to the dual problem, face clusters can be made to preserve planarity where possible, in the same way that vertex simplification tries to preserve shape.

3.3.2. The Face Clustering Algorithm

Much of the material described in the next three sub sections is summarised from Garland’s work on surface simplification. Further details can be found in chapter 8 of his dissertation, “Quadric-based Polygonal Surface Simplification” [Garl99]. Further discussion and refinement of the face cluster algorithm can be found in **Chapter 5**.

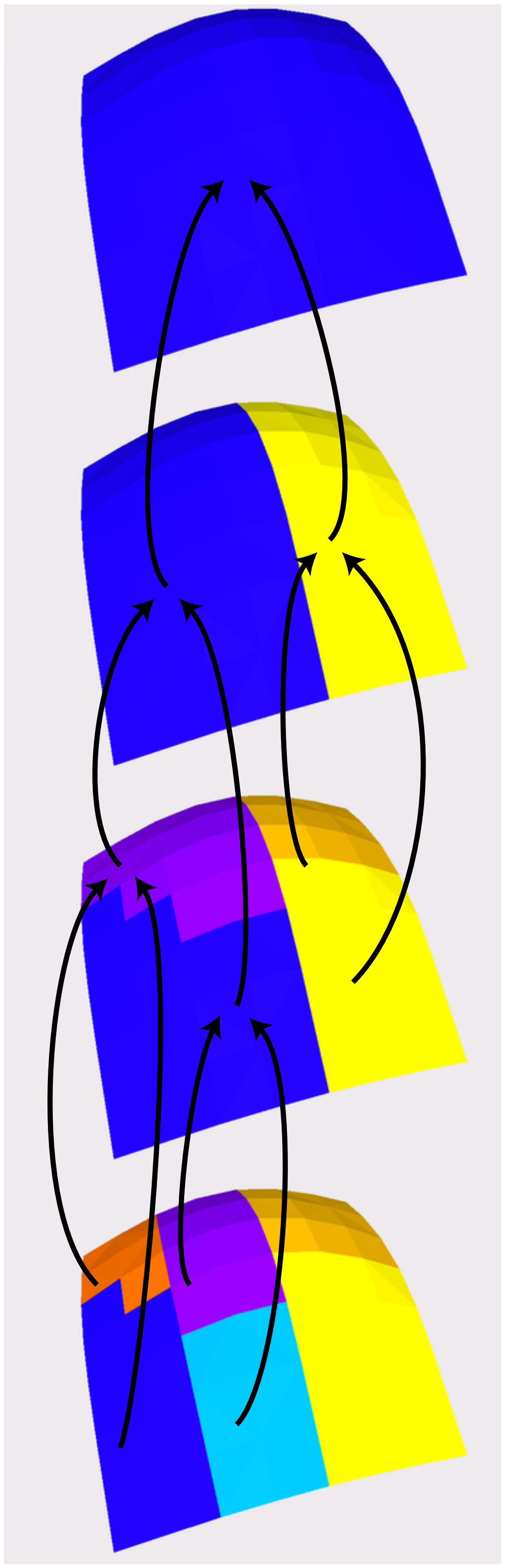


Figure 25: Face clusters. Each face cluster is created by merging two other adjacent face clusters. The several levels of the hierarchy are shown here for a small, curved mesh. At bottom, there are six surface regions, which are iteratively merged pairwise, until we are left with only a single cluster covering the entire surface, at top. This is the root of the cluster hierarchy.

Note how the cluster regions largely follow the curvature of the surface.

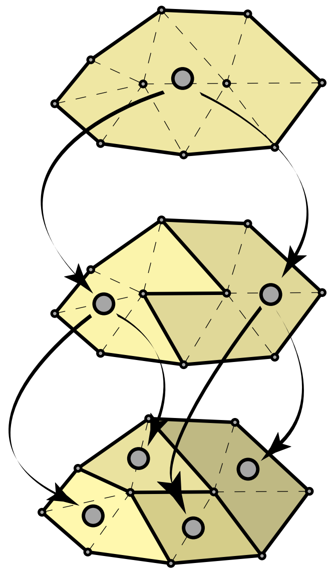


Figure 26: Building a Face Cluster Hierarchy.

Our aim is to automatically construct a hierarchy of surface regions for every connected surface component. Each region must consist of a connected set of faces. We begin by forming the *dual graph* of the input mesh. Every face of the surface is mapped to a node in the dual graph, and two dual nodes are connected by an edge if the corresponding faces are adjacent on the surface. For the moment, we will assume that the surface is a manifold with boundary; in other words, every surface edge has at most 2 adjacent faces. We will also assume that all input polygons have been triangulated.

In the dual graph, an edge collapse corresponds to merging two face clusters together. **Figure 28** illustrates a simple example. On the left is a mesh where each dual node corresponds to a single face; in other words, each face is its own cluster. After collapsing a single dual edge, the two darkened triangles have been merged into a single cluster.

To construct a complete hierarchy, we use a simple greedy procedure to iteratively collapse dual edges. We assign a "cost" to each dual edge and iteratively collapse the edge of least cost. After each collapse, we update the costs of the surrounding edges. At each iteration, we have constructed a partition of the surface into disjoint sets of connected faces. This is in contrast to simplification,

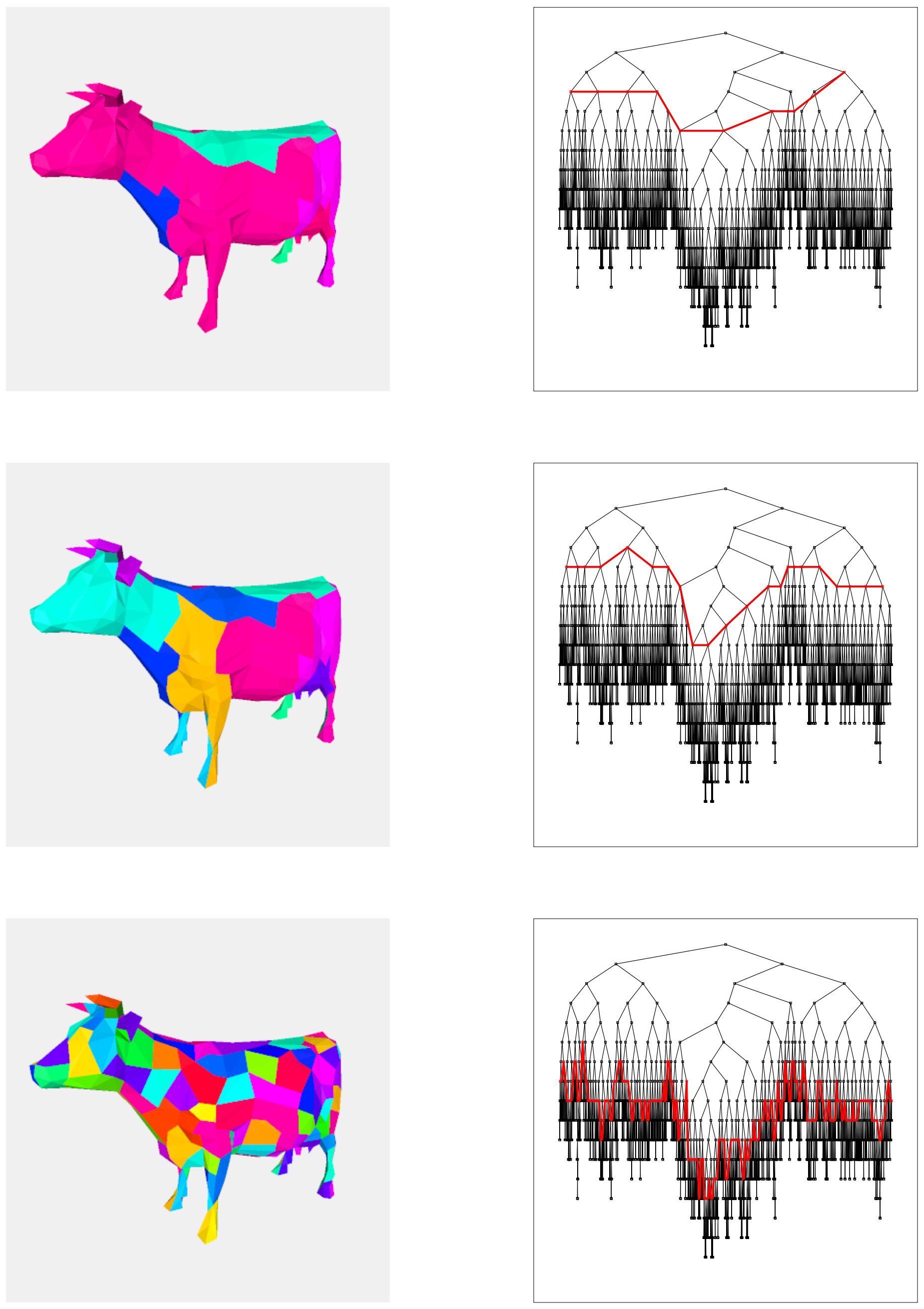


Figure 27: An example face cluster hierarchy. Left: several different clusterings of a cow model. Each cluster has a different colour, and the number of clusters increases from top to bottom. Right: the corresponding cut through the cow's face cluster hierarchy is shown in red.

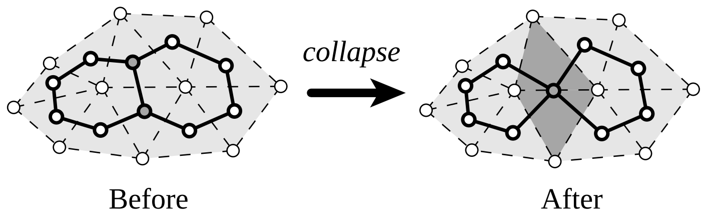


Figure 28: One edge of the dual graph (solid lines) is collapsed. The faces corresponding to its endpoints (darkened) are merged into a single face cluster.

where at each iteration we would have constructed an approximate surface. Face clustering an object with f faces costs $O(f \log f)$ time and $O(f)$ memory.

3.3.3. Dual Edge Cost Metric

Each dual edge corresponds to a set of faces, namely the faces associated with its endpoints, and a set of points $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ determined by the vertices of these faces. We begin by fitting a least squares optimal plane through this set of points using the standard technique of principal component analysis (PCA) [Joll86]. We construct the covariance matrix

$$\mathrm{CV} = \sum_i (\mathbf{v}_i - \bar{\mathbf{v}})(\mathbf{v}_i - \bar{\mathbf{v}})^{\mathrm{T}} \quad \text{where} \quad \bar{\mathbf{v}} = \sum_i \mathbf{v}_i / k \tag{11}$$

The three eigenvectors of the matrix CV determine a local frame with $\bar{\mathbf{v}}$ as the origin. The eigenvector corresponding to the smallest eigenvalue is the normal of the optimal plane, and the remaining eigenvectors define a frame for parameterizing this plane. Given this optimal plane, we assign a cost

$$E = E_{fit} + E_{dir} + E_{shape} \tag{12}$$

to every dual edge. In addition, we use this plane to compute an oriented bounding box for the cluster.

The first, and most important, error term measures the planarity of the cluster, and is defined to be the average squared distance of all the points in the cluster to the optimal plane

$$E_{fit} = \sum_i (\mathbf{n}^{\mathrm{T}}\mathbf{v}_i + d)^2 / k \qquad (13)$$

Following the quadric error metric of Garland and Heckbert [Garl98], we can compactly represent this sum using quadrics as

$$E_{fit} = \sum_i P_i(\mathbf{n}, d)/k = \left(\sum_i P_i\right)(\mathbf{n}, d)/k \qquad (14)$$

where

$$P_i = (\mathbf{A}_i, \mathbf{b}_i, c_i) = (\mathbf{v}_i\mathbf{v}_i^{\mathrm{T}}, \mathbf{v}_i, 1) \qquad (15)$$

$$P(\mathbf{n}, d) = \mathbf{n}^{\mathrm{T}}\mathbf{A}\mathbf{n} + 2\mathbf{b}^{\mathrm{T}}(d\mathbf{n}) + cd^2 \qquad (16)$$

Each dual node has an associated fit quadric P which requires ten coefficients to represent the symmetric 3x3 matrix $\mathbf{A}$, the 3-vectors $\mathbf{b}$, and the scalar c. It is important to note that the covariance matrix can be extracted directly from this quadric: $\mathrm{CV} = \mathbf{A} - (\mathbf{b}\mathbf{b}^{\mathrm{T}})/c$.

Minimizing the planarity term E_{fit} will naturally tend to merge clusters which are collectively nearly planar. However, a surface may locally fold back on itself; it will seem nearly planar, but the normal of the plane will not be a good fit for the surface normals in the region. Therefore, we add an additional error term which measures the area-averaged deviation of the plane normal $\mathbf{n}$ from the surface normals

$$E_{dir} = \left(\sum_i A_i(1 - \mathbf{n}^{\mathrm{T}}\mathbf{n}_i)^2\right) / \sum_i A_i \qquad (17)$$

where A_i and $\mathbf{n}_i$ are the area and unit normal, respectively, of a face i in the cluster. Again, we can represent this error term using a quadric by rewriting $(1 - \mathbf{n}^{\mathrm{T}}\mathbf{n}_i)^2$ as $R_i(n) = \mathbf{n}^{\mathrm{T}}\mathbf{D}_i\mathbf{n} + 2\mathbf{e}_i^{\mathrm{T}}\mathbf{n} + f_i$ where

$$R_i = (\mathbf{D}_i, \mathbf{e}_i, f_i) = (\mathbf{n}_i\mathbf{n}_i^{\mathrm{T}}, -\mathbf{n}_i, 1) \qquad (18)$$

Every dual node has two associated quadrics, a fit quadric and an orientation quadric. Whenever we merge two nodes, we add their corresponding quadrics together to form the quadrics for the resulting node.

3.3.4. Compact Shape Bias

If planarity is our only clustering criterion, then the combination of the error terms E_{fit} and E_{dir} performs well. However, for our radiosity application, we would also like the regions to have a compact shape; a compact region is one which is as nearly circular as possible. Following Kalvin and Taylor [Kalv96], we define the compactness γ of a region to be the ratio of its squared perimeter to its area. Larger values of γ correspond to less compact (more irregular) regions.

For a given dual edge, the clusters associated with its endpoints will have compactness γ_1 and γ_2. If γ is the compactness of the resulting merged cluster, we define the shape penalty as:

$$E_{shape} = 1 - \max(\gamma_1, \gamma_2)/2\gamma \qquad (19)$$

This penalizes clusters where the compactness worsens and rewards clusters where the compactness improves. **Figure 29** illustrates the effect of the shape bias. With the shape bias disabled (a), the clusters tend to follow the features of the model but may have non-compact shapes. The long strips stretching down the forehead and the nose are a good example. In contrast, when using the shape bias (b), the clusters still reflect the form of the surface, but they have a much more compact shape.

3.3.5. A Face Cluster Node

Each node in our face cluster hierarchy is a representation of an approximately coplanar region on the mesh. It acts as a container for a set of connected faces (see **Figure 30**). Once we have constructed the hierarchy, we must address the question of what representative surface properties to store for each node within it, in addition to the pointers to the two child face clusters that partition it. A balance must be struck between the amount of useful information we store, and the amount of memory or disk space the hierarchy will occupy.

As discussed, for each node *i* in the face cluster hierarchy, we calculate an oriented bounding box for the faces it contains using principal component analy-

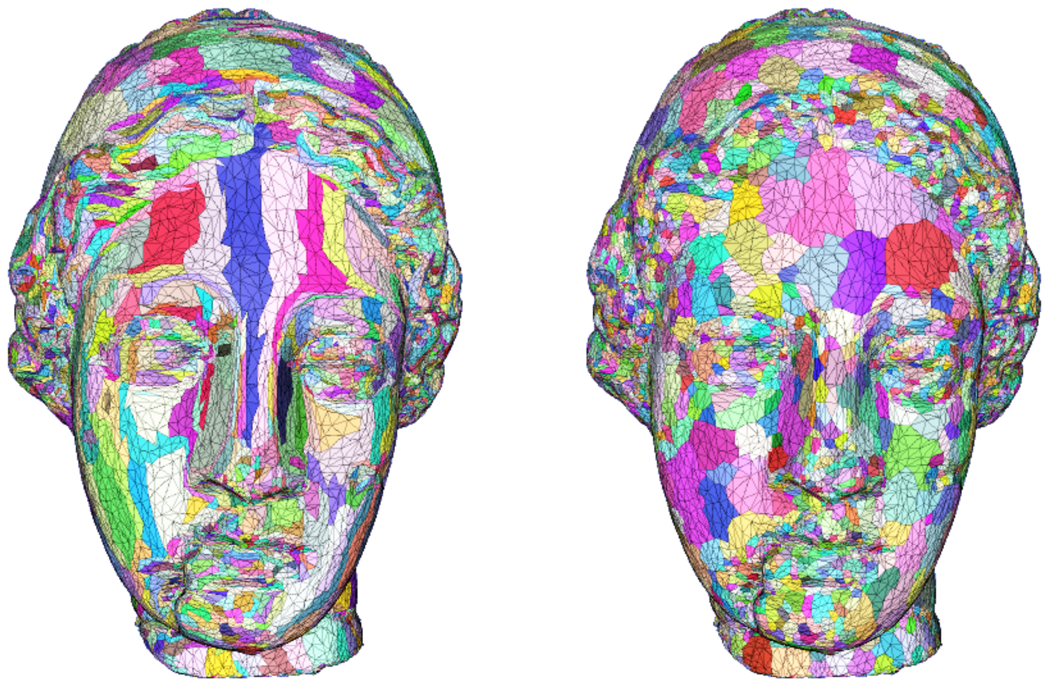


Figure 29: Partitions of the venus head, each with 8000 clusters

sis [Joll86]. For each node, we store this, in addition to the sum of the area-weighted normals of those faces:

$$\mathbf{S} = \sum_j A_j \hat{\mathbf{n}}_j \tag{20}$$

for each face j belonging to the cluster. As we will show in **Section 3.4** and in more detail in **Chapter 4**, this is a useful approximation of the reflective qualities of the faces within the node.

3.4. The Face Cluster Radiosity Algorithm

In this section we describe our algorithm for simulating radiosity on multiresolution models that use face clustering. We show how the standard radiosity equa-

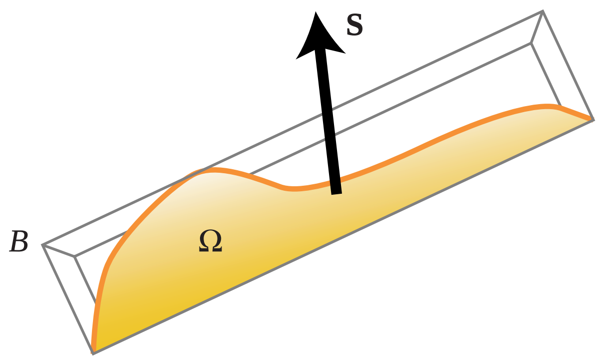


Figure 30: A face cluster. For our purposes, a face cluster is defined by a group of connected faces Ω, with an enclosing oriented bounding box B, and an area-weighted normal, $\mathbf{S}$.

tions can be modified to better suit the use of these hierarchies, starting with the standard formulas governing diffuse interreflection [Cohe93].

3.4.1. Standard Radiosity Derivation

The radiosity b_i of surface *i* is the outgoing power per unit area due to emission or reflection. It equals emittance e_i plus reflectance ρ_i times the sum of contributions from other surfaces *j*,

$$b_i = e_i + \rho_i \sum_j F_{ij} b_j . \tag{21}$$

Here F_{ij} is the form factor, the fraction of light leaving surface *j* that arrives at surface *i*,

$$F_{ij} = \frac{1}{A_i} \int\int \frac{\cos\theta_i \cos\theta_j}{\pi r_{ij}^2} v_{ij} dA_i dA_j , \tag{22}$$

and A_i is the area of element *i*, θ_i and θ_j are the polar angles, r_{ij} is the distance between points on elements *i* and *j*, and v_{ij} is the visibility between those points: 1 if visible and 0 if occluded.

To solve the above system of equations, our method, like most hierarchical radiosity algorithms, repeatedly gathers light from all other elements *j* to update the radiosity of element *i*. This is equivalent to one iteration of Jacobi's method for solving linear systems. Multiplying the equation above by A_i, we get an equation whose terms have units of power (area times radiosity):

$$A_i b_i = A_i e_i + \rho_i A_i E_i \tag{23}$$

The *irradiance* E_i is the incident power per unit area. Calculating it exactly is typically intractable, so we approximate it using the point-to-point approximation of the form factor [Cohe93],

$$E_i = \sum_j \frac{(\hat{\mathbf{n}}_i^{\mathrm{T}} \hat{\mathbf{r}}_{ij})_+ (\hat{\mathbf{r}}_{ji}^{\mathrm{T}} \hat{\mathbf{n}}_j)_+}{\pi r_{ij}^2} v_{ij} A_j b_j \tag{24}$$

where bold symbols denote vectors, $(x)_+ \equiv max(0, x)$, $\hat{\mathbf{n}}_i$ is a unit normal to surface *i*, and $\hat{\mathbf{r}}_{ij}$ is the unit direction vector from *i* to *j*. We have written the dot products in matrix notation because we will shortly be exploiting the associativity of matrix multiplication to rewrite this formula.

3.4.2. Vector-based Radiosity

The classical radiosity method assumes piecewise constant (Haar) basis functions and planar surfaces. In a hierarchy, the children are coplanar with the parent. For the purposes of projecting radiosities up and down the tree, radiosities are scalar quantities. If this method is applied to a multiresolution model, it causes curved or bumpy portions of the model to be shaded a flat colour, leading to a faceted appearance that hides the geometric detail (**Figure 19**c). This is similar to the step-function effect in constant-basis radiosity, but applying a post-process smoothing step at the leaves is no longer sufficient to cover up these discontinuities.

We now adapt the radiosity method to multiresolution models that use face clustering. Consider the light transfer from one cluster to another (**Figure 31**). We let *j* be an element in the source cluster and *i* be an element in the receiver cluster. If we assume that all (*i*,*j*) pairs are inter-visible and that the sources are close together and far from the receiver, then $\hat{\mathbf{r}}_{ij}$, $\hat{\mathbf{r}}_{ji}$, r_{ij}, and $\bar{v}_{ij}$ are independent of i and j, and we can approximate the irradiance from a single cluster as:

$$E_i \approx \left(\hat{\mathbf{n}}_i^{\mathrm{T}} \left[\frac{-\hat{\mathbf{r}}}{\pi r^2} (\hat{\mathbf{r}}^{\mathrm{T}} \textstyle\sum_j \hat{\mathbf{n}}_j A_j b_j)_+\right] \bar{v}_i\right)_+, \tag{25}$$

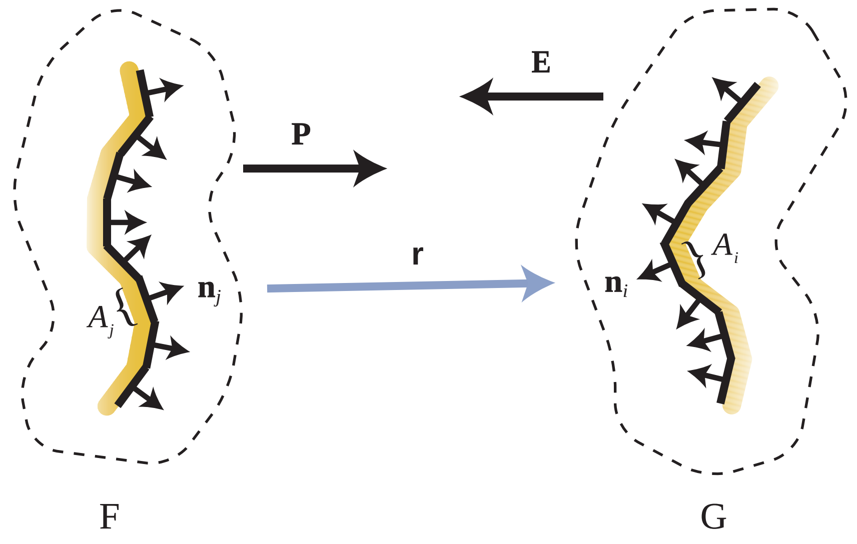


Figure 31: The transfer of radiosity between two clusters of faces. An individual face j in the source cluster F radiates energy towards an individual face i in the receiving cluster G.

where $\mathbf{r}$ is the average distance vector. Using **Equation 19** we can then rewrite the transfer in terms of two vector quantities. We find that:

$$E_i = \hat{\mathbf{n}}_i^{\mathrm{T}} \mathbf{E}, \tag{26}$$

where the *irradiance vector* $\mathbf{E}$ is defined as

$$\mathbf{E} = \bar{v} \frac{-\delta(\mathbf{S}_G, \hat{\mathbf{r}})}{\pi r^2} (\hat{\mathbf{r}}^{\mathrm{T}} \mathbf{P})_+, \tag{27}$$

with

$$\delta(\mathbf{S}, \mathbf{x}) \equiv \begin{cases} \mathbf{x} & \text{if } \mathbf{x} \cdot \mathbf{S} > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{28}$$

and $\mathbf{S}_G$ being the sum area normal of the receiving cluster. (The irradiance vector has also been employed by Arvo [Arvo94].)

The *power vector* $\mathbf{P}$ is defined as:

$$\mathbf{P} = \sum_j \hat{\mathbf{n}}_j A_j b_j = \sum_j \mathbf{S}_j b_j . \tag{29}$$

If the source cluster has constant radiosity b across its constituent faces, we can simply write:

$$\mathbf{P} = \mathbf{S}_F b , \tag{30}$$

where $\mathbf{S}_F$ is the sum area normal of the source cluster.

The irradiance vector is parallel to $\mathbf{r}$, and is at a maximum when the power vector is also parallel to $\mathbf{r}$; it points towards the source cluster. Recording this information, rather than the scalar irradiance to the average plane of the receiver, allows coarse variations in the irradiance as a function of orientation to be modelled. This eliminates most of the faceting effects of **Figure 19**c, as seen in **Figure 19**d.

We employ the power vector to permit non-planar clusters to approximate their outgoing power compactly and quickly. The magnitude of the vector approximates the total power leaving the cluster, and the direction of the vector indicates the hemisphere toward which most of the energy is directed (**Figure 32**).

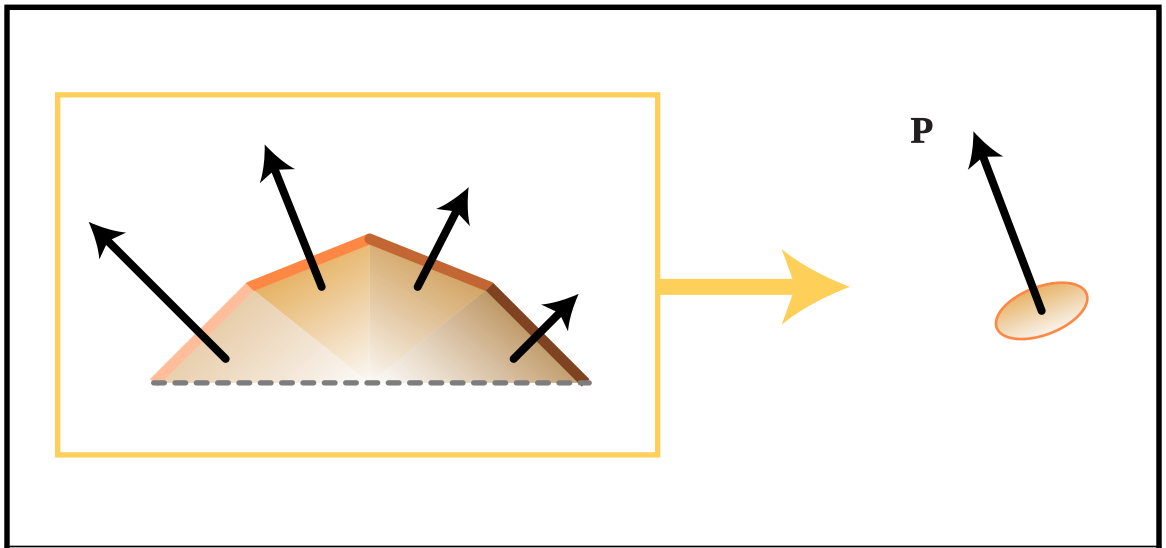


Figure 32: The power vector. The radiosity emitted by the surfaces on the left is approximated by the elemental surface on the right, whose direction is the power-weighted sum normal.

When calculating the illumination of a cluster by one or more other clusters, we transform each power vector into the corresponding irradiance vector via **Equation 27**, and sum the resulting irradiance vectors.

Push Pull

These formulas are generalizations of the standard radiosity equations. In the case of coplanar clusters, they reduce to the familiar hierarchical radiosity push-pull formulas. In the general case, for the push operation, we find that

$$\mathbf{E}_{child} = \delta(\mathbf{S}_{child}, \mathbf{E}_{parent}) + \mathbf{R}_{child}, \tag{31}$$

where $\mathbf{R}_{child}$ is the sum of the irradiance vectors incident directly on the child. For the pull operation, there are two alternatives. We can operate directly on the power vector, so that

$$\mathbf{P}_{parent} = \sum \mathbf{P}_{child}, \tag{32}$$

or we can instead operate on cluster radiosities, with

$$b_{parent} = \frac{\sum b_{child} A_{child}}{A_{parent}}, \tag{33}$$

and reconstruct the power vector with $\mathbf{P}_{parent} = \mathbf{S}_{parent} b_{parent}$, as in **Equation 30**. The former has the advantage of elegance, and the ability to better match the chief direction of radiance of a cluster whose radiosity varies considerably over its surface. The latter has the advantage that it is more amenable to error analysis and the construction of bounds, as we shall see in **Chapter 4**.

Transfer Coefficients

To model the transfer of radiosity, instead of a single transfer coefficient, we store a *transfer vector* $\mathbf{m}$, which allows us to apply **Equation 27**. Where we are working directly with power vectors, as in **Equation 32**, we can write

$$\mathbf{m} = \hat{\mathbf{r}} \sqrt{\frac{\bar{v}}{\pi r^2}}, \tag{34}$$

and we have $\mathbf{E} = \delta(\mathbf{S}_G, -\mathbf{m})(\mathbf{m}^{\mathrm{T}}\mathbf{P})_+$. Where we are working with source radiosities, we can instead use

$$\mathbf{m} = \frac{\bar{v}\delta(\mathbf{S}_G, -\hat{\mathbf{r}})(\hat{\mathbf{r}}^{\mathrm{T}} \cdot \mathbf{S}_F)_+}{\pi r^2}, \tag{35}$$

and $\mathbf{E} = \mathbf{m}b$. Currently, this latter approach is more desirable, as we have a supporting error analysis for it. The former may eventually be preferable if a matching analysis can be found.

Generally, it is easiest to estimate $\mathbf{m}$ by generating n fixed sample points across both the source and receiver clusters, and taking

$$\bar{\mathbf{m}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{m}_i \tag{36}$$

where each $\mathbf{m}_i$ is calculated using $\mathbf{r}_i$, the vector from sample point *i* on the source to sample point *i* on the receiver. These samples can also be used for determining bounds on the transfer, and thus an estimate of the error in the transfer, which can be used to drive refinement [Gibs96, Lisc94]. However, I have developed a more accurate and robust approach that relies on conservative bounds for a cluster's projected area in a given direction. An in-depth treatment of this error estimate will be presented in **Section 4.4**.

Leaf Radiosities

Finally, at the leaves of the hierarchy, where we must transform the accumulated irradiance vectors into radiosity, we apply the equation

$$b = \frac{\rho \mathbf{S}^{\mathrm{T}}\mathbf{E}}{A} + e. \tag{37}$$

Again, we can reconstruct a leaf's power vector from this by application of **Equation 30**[1].

This treatment of vector-based radiosity transfer assumes a monochromatic world. It can easily be extended to the familiar RGB colour model; we simply maintain $\mathbf{E}$ and $\mathbf{P}$ or b separately for each colour channel, and operate on each

1. In **Section 4.4.5** we will show how to trivially rewrite **Equation 37** to take account of interreflection within the leaf cluster.

pair independently. Note that the transport vector, **m**, is still wavelength independent.

3.4.3. Algorithm Description

There are three types of nodes in the hierarchy used by our algorithm. At the top, volume clusters contain all unconnected parts of the scene. In the middle, face clusters contain connected surface meshes. At the bottom, there are polygonal elements, and refinements of those elements. In our implementation, all of these nodes use a common object-oriented interface to communicate with each other. Usually much of each face cluster hierarchy remains unused; only those face clusters at the top of the tree are paged in during the solution phase.

Radiosity using vector-based transfer proceeds in much the same manner as the irradiance / radiosity method first popularised by [Gers94], and outlined in **Figure 33**. In Gershbein's method, irradiance is gathered to each node in the hierarchy, and then pushed down to the leaves, whereupon it is converted to radiosity by the application of reflectance and emittance operators, and pulled back up the hierarchy. In our algorithm, the irradiance vector, rather than scalar, is pushed to the solution leaves, and either the power vector or scalar radiosity is pulled back up the hierarchy. (Here "solution leaves" refers to the lowest level of cluster or face refinement in the hierarchy, not the input polygons.) Below is an outline of the algorithm.

Preprocessing

Face cluster hierarchies are generated for the input models. These hierarchies are dependent only on the geometry of the models, and can be reused over multiple instantiations of each model. This process is done off-line, and typically only once, whenever a new model is acquired.

Initialisation

The scene description is read in. Hierarchical radiosity elements are then created for all root face cluster nodes in the scene (there are typically a small number per model), and these are volume clustered to complete the initial element hierarchy.

Solution

The solver proceeds according to the pseudocode in **Listing 1**. The `refine` procedure follows that outlined in [Cohe93]. When a face cluster element needs to be subdivided, its two child elements are created using position and sum area nor-

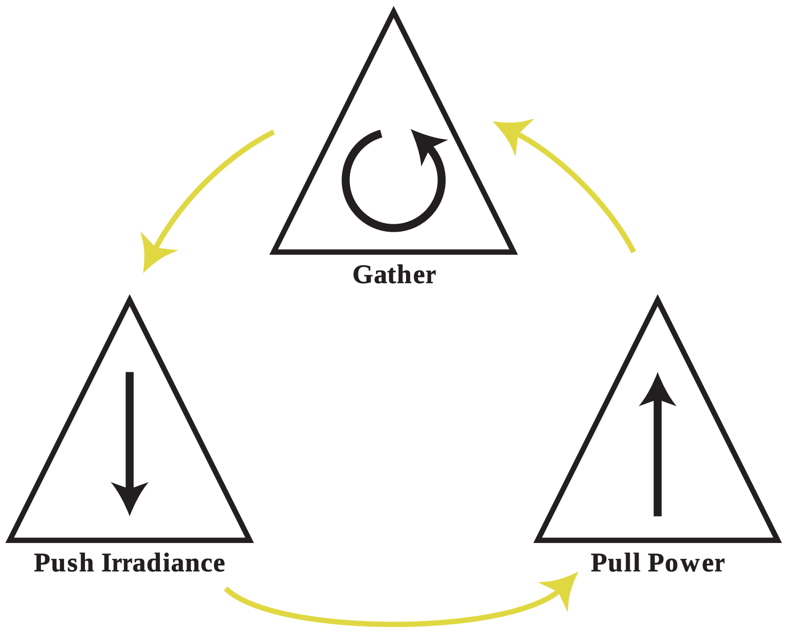


Figure 33: Schematic of the Hierarchical Radiosity Cycle. The current set of element radiosities are *gathered* across transfer links to find the corresponding irradiances at different levels of the solution hierarchy. These irradiances are then *pushed* to the leaves of the hierarchy to find the sum irradiance there, transformed to radiosities by application of the reflection operator at each leaf, and then *pulled* back up the hierarchy to calculate the average radiosity at each level of detail.

mal information retrieved from the cluster file on disk. When the input polygons are reached, the children become Haar elements, and refinement proceeds as in a standard hierarchical radiosity algorithm. Storage for irradiance vectors and other such element information is only required for those face cluster nodes actually being used by the solver. The solver runs in time and space sublinear in k.

Post Processing

After the solution algorithm has terminated, the radiosity solution is propagated to the leaves of the model by applying the irradiance vectors at the leaves of the transport tree to all their descendents (e.g., node T of **Figure 24**c). This final proc-

```
solve(tree root)
    while (not converged)
        gather(root)
        pushpull(root, 0)
        refine(root, $\varepsilon$)

gather(element i)
    $\Delta\mathbf{E}_i = -\sum_{\text{links j->i}} \bar{v}_{ij}\mathbf{m}_{ij}\mathbf{m}_{ij}^{\mathrm{T}}\mathbf{P}_j$
    foreach (child c of i)
        gather(c)

pushpull(element i, vector $\mathbf{E}$)
    // $\mathbf{E}$ is irradiance on i from parent
    $\mathbf{E} = \mathbf{E} + \Delta\mathbf{E}_i$
    if (i is a leaf)
        // convert irradiance to power
        $\mathbf{P}_i = \mathbf{S}_i(\mathrm{Max}(\hat{\mathbf{S}}_i^{\mathrm{T}}\mathbf{E}_i, 0)\rho_i + e_i)$
    else
        $\mathbf{P}_i = 0$
        // push irradiance, pull power
        foreach (child c of i)
            pushpull(c, $\mathbf{E}$)
            $\mathbf{P}_i = \mathbf{P}_i + \mathbf{P}_c$
```

Listing 1: Pseudo-code for the Face Cluster Radiosity algorithm

ess is $\Theta(k)$, but is typically insignificant compared to the solution time. The radiosities of the vertices of the models in the scene are then written out to disk.

3.4.4. Examples

Detailed results of the face cluster radiosity algorithm will be presented in **Chapter 7**. In the meantime, here are some informal examples of the algorithm in action. **Figure 34** shows results for a simple scene similar to that of **Figure 14**. In this scene, however, the surface is bumpy, containing 13,000 triangles. Whereas volume clustering takes four minutes to calculate the illumination of the surface, face cluster radiosity takes just over a second. The use of surface-based clusters also helps avoid the artifacts present in the volume clustering solution. These dark patches are caused when the crest of a particular bump belongs to a volume cluster that lies above the cluster containing the rest of the bump. This crest cluster then shadows the cluster underneath, causing the rest of the bump to be underlit. Face clusters explicitly follow the shape of the mesh, avoiding this kind of overlap problem.

Figure 35 shows how a clustered object can also be used as a light source; a tessellated torus is used to illuminate a bumpy surface. Ray tracing methods such as Radiance have particular problems with such large distributed light sources; they must distribute samples over the light source whenever its contribution is being calculated, which requires some knowledge of its geometry. A common approach is to simply sample over the bounding box of the light, which in the case of the torus would result in a large number of samples missing the light source.

Finally, **Figure 36** shows a scene containing a bust lit from two different directions, along with details of the illumination transfer.

3.5. Discussion

3.5.1. Strengths and Weaknesses

Our algorithm gives the greatest benefit for finely tessellated objects. For more intricate, space-filling objects such as trees, we rely on volume clustering to provide a good simulation. This is because of the limitation in our algorithm that each topologically connected surface has its own face cluster tree. For instance, a pile of pebbles might need a separate tree for each pebble, if those pebbles were modelled separately. While we currently aggregate disconnected components by using volume clustering, better methods for tightly packed components such as the pebble pile are an avenue for future research.

From any given viewpoint, some of the input polygons in a scene containing large models can be invisibly small. Arguably a similar picture from that viewpoint could be generated using a much simpler scene. However, this ignores the possibility of using an output mesh in a walk-through, where we might wish to pass much closer to some of the models in the scene. Also, it is tedious to man-

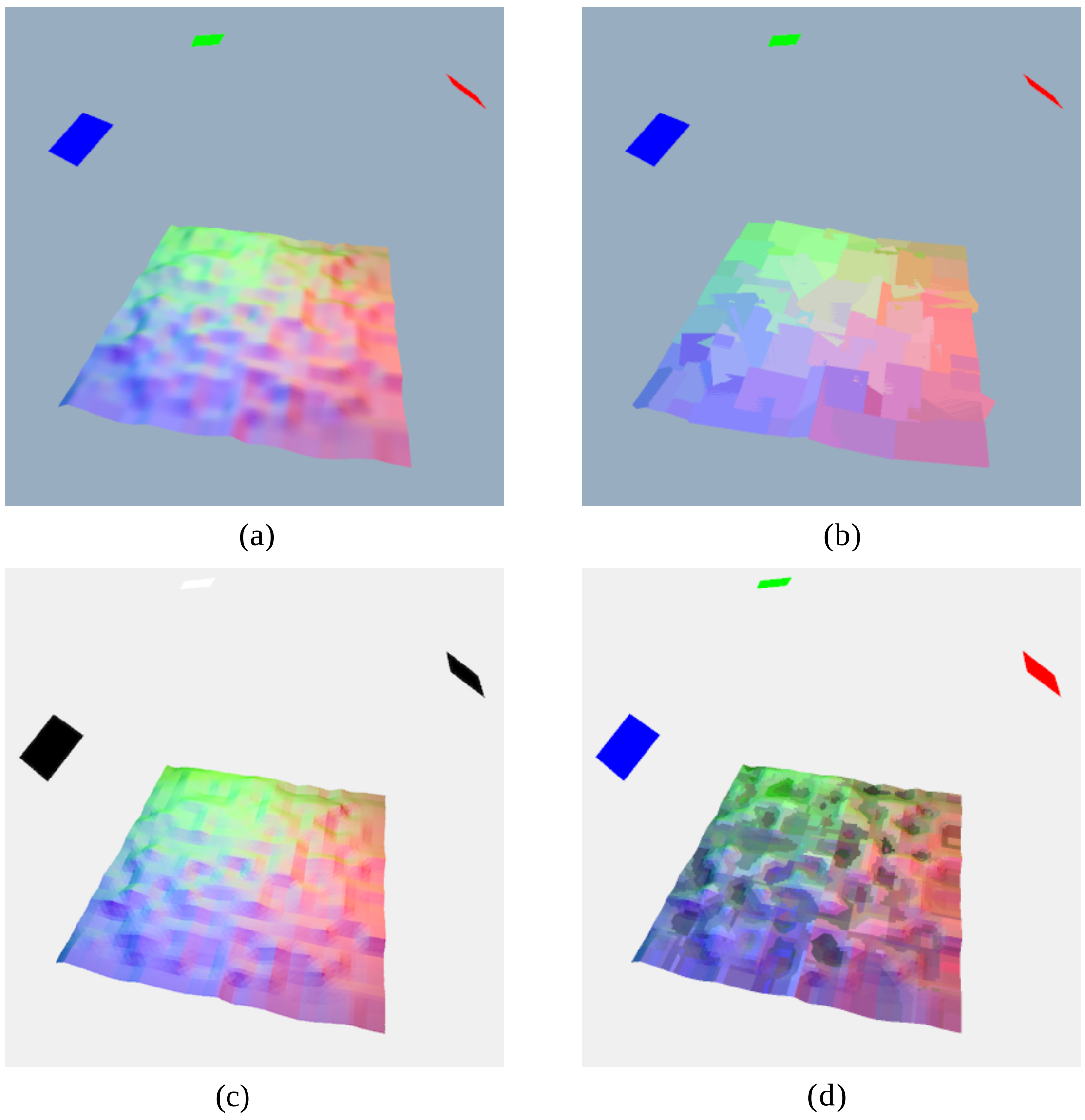


Figure 34: Face-cluster radiosity on a bumpy surface. In (a) face-clustering handles bumpy surfaces as well as a flat mesh. (b) shows the corresponding clusters. This solution took under two seconds to generate. As reference, we have both (c), the results of running Radiance (288s) and (d), volume clustering (260s) on the same scene.

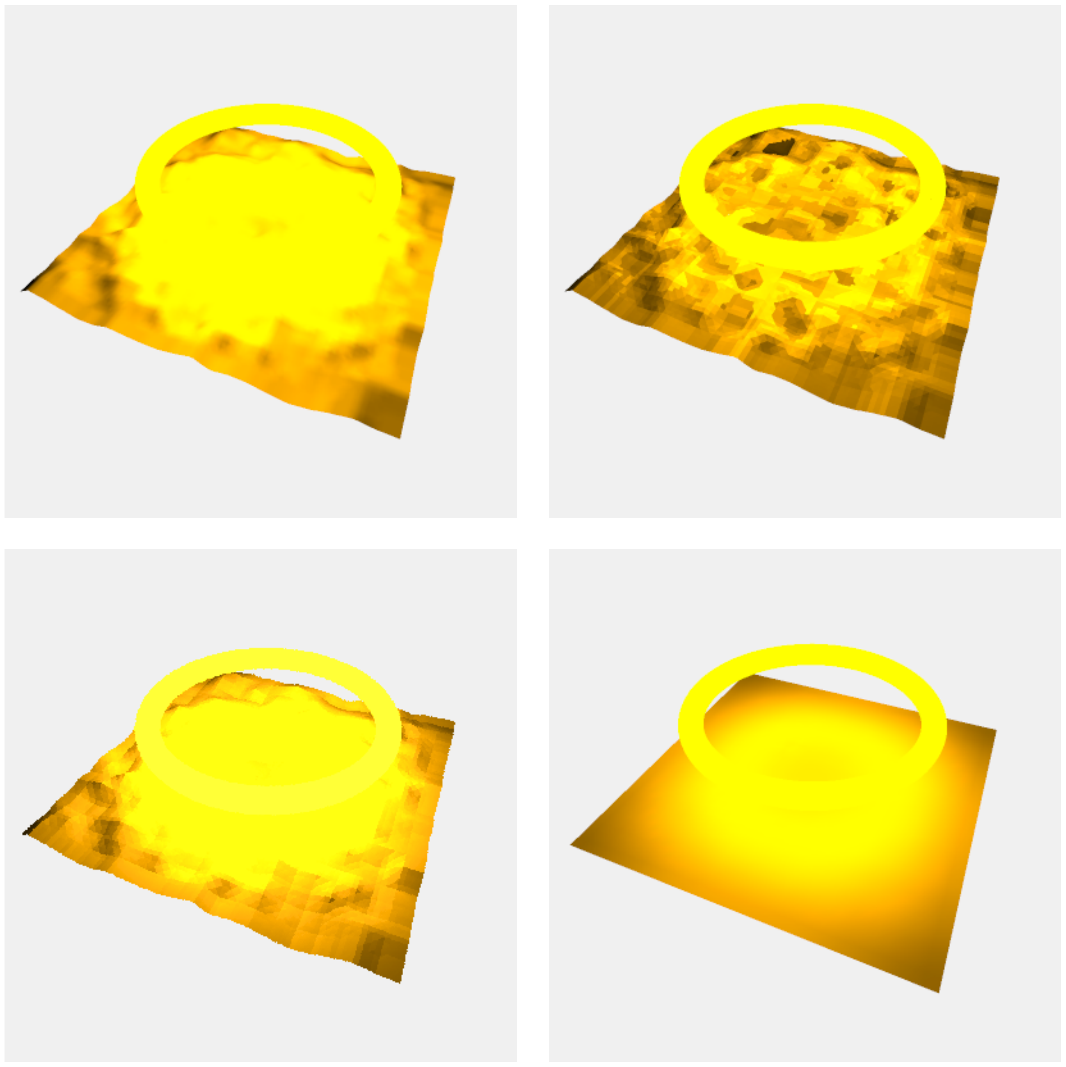


Figure 35: A highly tessellated, curved light source. The light source is a torus containing 8,200 polygons. Top left; face cluster radiosity took 28 seconds to calculate this (view-independent) solution. In contrast, volume clustering (top right) took well over an hour to calculate the corresponding image, and Radiance (bottom-right) took five hours. A solution for a flat surface is shown for comparison purposes, bottom right.

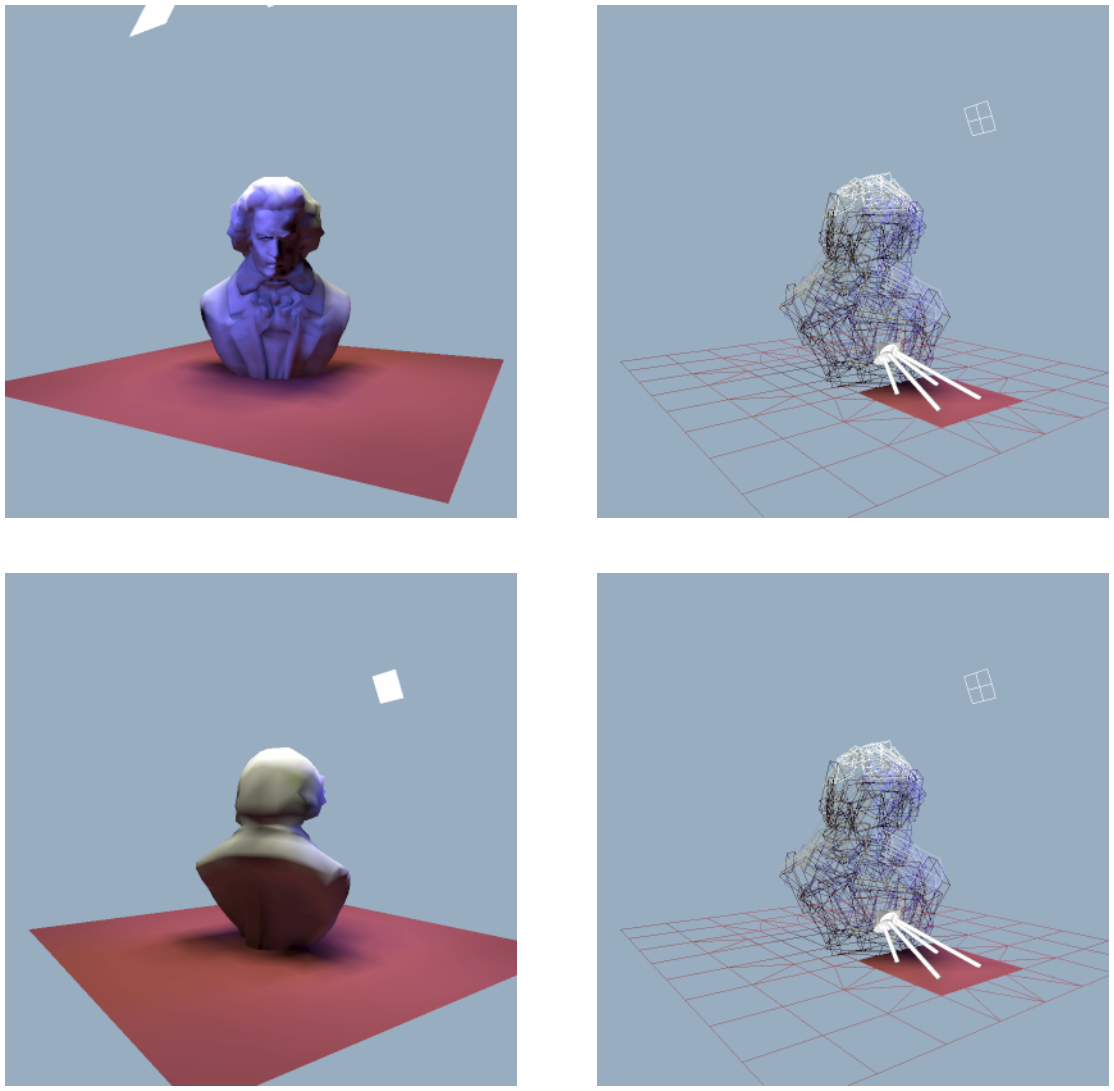


Figure 36: The bust of Beethoven. The bust is lit from overhead by a white light, and from above and to the left by a blue light. Shown are (a) the results of FCR, (b) the corresponding face clusters, (c) reflection off the red carpet onto Ludwig's back, (d) the contribution of nearby patches to one of the back clusters.

ually preprocess a scene in a view-dependent manner to optimise radiosity simulations. Ultimately, we wish the radiosity simulation to be as independent of the model resolution chosen for viewing as possible.

3.5.2. Memory Locality

Radiosity computations are traditionally extremely memory intensive. Because of this, the memory locality of a radiosity algorithm is a critical aspect of its performance, albeit one that is often ignored. For instance, progressive radiosity is well known for its good total memory use; in many cases it is better than that of hierarchical radiosity methods. However, it has very poor locality; each iteration of the algorithm is essentially a linear sweep through several large vectors. Hierarchical radiosity has much better locality, in the sense that each solution iteration, which requires a tree-traversal of the data set, accomplishes much more than the equivalent progressive iteration, resulting in significantly fewer iterations to generate a solution.

This has interesting implications in terms of which algorithm is the speediest. For scenes where both progressive and hierarchical radiosity can fit their computations into physical memory, hierarchical radiosity is quickest. However, if the scene becomes large enough, the hierarchy will no longer fit in physical memory, and progressive radiosity becomes the quicker algorithm. Finally, when the scene is so large that both algorithms must be accommodated by virtual memory, the superior locality of hierarchical radiosity makes it again quicker than the progressive algorithm; in other words, hierarchical radiosity thrashes better than progressive radiosity.

Face cluster radiosity inherits the good data structure locality from the hierarchical radiosity algorithm. To take advantage of this, face cluster files are written in breadth-first order, so we get correspondingly good memory locality. (These files are simply memory-mapped for use by the solver.) FCR also has the added advantage of much smaller active memory use; it is almost always the case that only the first small section of each face cluster file is used during solution.

3.5.3. Surface Material Maps

The visual complexity of a scene is often enhanced by the application of various types of surface maps to the original models. Most of these are handled naturally by our algorithm. Texture maps can be pre-sampled at the leaves and filtered over the vertex or face cluster hierarchy as described in [Gers94]. A similar approach can be taken to emittance maps, although the quantity pre calculated for the hierarchy needs to be an emittance vector, similar to the power vector.

In particular, our approach extends well to the use of displacement maps. Displacement mapping is a useful tool for adding geometric complexity to simple models. For instance in Pharr et al. [Phar97] a highly complex scene of 5 million primitives is constructed from a number of simpler source primitives, each containing on the order of 10,000 polygons, by applying displacement maps. Displacement-mapped objects are characterized by highly tessellated surfaces with a high degree of planar coherence, but widely varying normals. In such cases it is possible to approximate the irradiance of large areas of the model with a single link, applying the irradiance vector to the local surfaces only at the time of the final render. Similarly, having the irradiance vector available makes applying bump maps [Blin78] to surfaces trivial.

3.5.4. Animated Models

The face cluster hierarchies generated by the method outlined here can be reused over different scenes, and even over successive frames of an animation, provided the models they are generated from do not change. Of course, this is not always the case, especially when a model takes the part of a central character in the animation, rather than a part of the static backdrop. It can be shown that rigid body transformations and uniform scaling do not affect the construction of the hierarchy; hierarchies would not have to be rebuilt for model animations featuring only such transformations.

However, deformations [Barr84, Gudu90], commonly used to bend and reshape models on-the-fly in a non-linear manner, can and do affect the hierarchy. In such cases, a new face hierarchy would have to be generated for the model for each frame in the animation in which it is deformed, thus undoing some of the benefit of my algorithm. (It should be noted that standard volume clustering has to rebuild the entire hierarchy for each new frame regardless, although arguably the same hierarchy pre-calculation techniques outlined here could be applied to volume clustering also.) Deformations result in only incremental changes to a model’s geometry from scene to scene, though, so we would hope that a local pass could refit the hierarchy at each step of the animation, without the cost of rebuilding it from scratch.

3.6. Summary

We can summarise the steps in the face cluster radiosity algorithm as follows:

- We construct a face cluster hierarchy file for each new model acquired. (Time and space super-linear in k. As we shall see in **Section 5.6**, this is in practice linear in k.)
- We create a scene from these models.
- The radiosity program reads in the scene description, and adds the root face cluster nodes to a volume cluster hierarchy. (Time and space linear in s, where s is the number of disconnected surfaces in the scene.)
- The gather / push-pull / refine solver is run. (Sub-linear in k.)
- The radiosity solution is propagated to the polygonal leaves of all models, and written to disk. (Linear in k.)

As usual, k is the number of polygons in the input scene.