Hierarchical Face Clustering on Polygonal Surfaces

Michael Garland[*] Andrew Willmott[†] Paul S. Heckbert[‡]

Abstract

Many graphics applications, and interactive systems in particular, rely on hierarchical surface representations to efficiently process very complex models. Considerable attention has been focused on hierarchies of surface approximations and their construction via automatic surface simplification. Such representations have proven effective for adapting the level of detail used in real time display systems. However, other applications such as ray tracing, collision detection, and radiosity benefit from an alternative multiresolution framework: hierarchical partitions of the original surface geometry.

We present a new method for representing a hierarchy of regions on a polygonal surface which partition that surface dinto a set of face clusters. These clusters, which are connected sets of faces, represent the aggregate properties of the original surface at different scales rather than providing geometric approximations of varying complexity. We also describe the combination of an effective error metric and a novel algorithm for constructing these hierarchies.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—surface and object representations

Keywords: face clusters, dual contraction, quadric error metrics,surface simplification, spatial data structures

1 INTRODUCTION

Modern graphics and visualization systems are often called upon to process and display scenes of tremendous geometric complexity. Individual surface models acquired from the physical world with laser scanners, for example, might easily contain a million or more polygonal faces. Such complexity is often useful. For instance, visualizations of architectural spaces can achieve a much more convincing level of realism if intricately detailed models are available. However, this realism comes at a price: computations involving models of such complexity are typically very expensive. To address this fundamental tradeoff between accuracy and efficiency, we must have effective methods to control the level of detail at which we process and display surface models. This need is particularly acute for interactive or real-time applications, which must maintain suitable update rates.

In recent years, considerable progress has been made on developing techniques for the automatic simplification of highly detailed polygonal models into faithful approximations using fewer polygons [9]. Methods are now available that can take excessively detailed models, say from laser range scanners, medical data sources, or CAD systems, and produce more economical models which approximately describe the same shape with far fewer polygons. Simplification algorithms have also been used to construct progressive representations of polygonal surfaces [14] and real-time display systems which can adapt the level of detail of the surface to fit the current viewing conditions [15, 23, 31]. All these prior methods focus on the problem of generating geometric approximations of the original surface. However, there is another class of applications where a hierarchical representation of the original surface itself is more suitable than a family of approximate surfaces.

We have developed an algorithm which produces a hierarchy of regions on a given polygonal surface. Each region is a connected set of faces, which we term a *face cluster*, and they completely partition the input surface. As we will see, our method is closely related to prior surface simplification methods. However, unlike simplification methods, this algorithm does not alter the original surface geometry in any way, nor does it produce any new approximate surfaces. Instead, it associates aggregate properties, such as a representative surface normal for instance, with each cluster in the hierarchy, thus representing the structure of the surface at multiple scales. The primary contributions of the work described in this paper are (1) a new hierarchical structure for representing surface partitions, (2) a novel algorithm for constructing these hierarchies, and (3) an effective error metric to guide this construction process.

Various computational tasks in graphics and visualization systems can potentially benefit from the face cluster hierarchies produced by our method. By enclosing each face cluster with a bounding volume, we can produce hierarchies suitable for ray tracing, collision detection, or other algorithms which perform spatial queries. Alternatively, we can use the cluster hierarchies as a basis for efficient simulation algorithms. For example, hierarchical finite element methods often subdivide the input geometry to an appropriate level of detail to obtain an accurate solution. By using our face cluster hierarchies, we can formulate methods which can conceptually "unsubdivide" and treat entire surface regions as a single unit. We previously described how radiosity simulation [30] can be made dramatically more efficient by applying such hierarchies but did not detail how they could be constructed. Here we focus on the precise representation and efficient construction of face cluster hierarchies.

2 SURFACE HIERARCHIES

For applications such as real-time rendering, we typically desire multiresolution models that can provide geometric approximations of the original surface. Depending on the viewing conditions, the run-time system will select the appropriate geometric level of detail to display. However, there are other applications in which we are more interested in the aggregate properties of surface regions. Rather than extracting a single approximation from the hierarchy, we would like to perform computations using the hierarchy itself. For applications which rely heavily on spatial queries, we might like a hierarchy of bounding volumes that enclose successively larger regions of the surface. For other simulation-oriented applications (e.g., finite element methods), we might want

[*]University of Illinois at Urbana–Champaign. garland@uiuc.edu
[†]Carnegie Mellon University. ajw@cs.cmu.edu
[‡]Carnegie Mellon University. ph@cs.cmu.edu

I3D '2001, Research Triangle Park, NC USA
© ACM 2001 1-58113-292-1/01/01 ...$5.00

to compute coarse solutions over entire surface regions, successively refining the result by considering smaller scale elements in selected areas. We might, for instance, want to approximate successively larger regions of the surface with planar elements.

2.1 Vertex Hierarchies from Simplification

Most current surface simplification algorithms begin with a triangulated polygonal surface and iteratively apply a simplification operator to remove elements at each step. One such operator, which has become an increasingly popular choice, is edge contraction (see Figure 1). Thus, starting with the original model, iterative contraction algorithms generate a sequence of approximate surfaces until arriving at a final approximation.

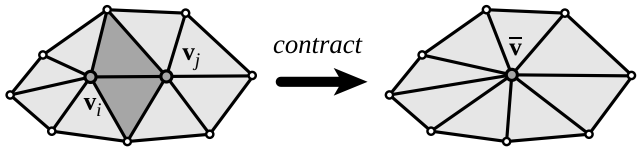


Figure 1: Edge $(\mathbf{v}_i, \mathbf{v}_j)$ is contracted; two faces and one vertex are removed.

When we generate approximations using iterative contraction, we also construct a vertex hierarchy. This is a binary tree whose leaves are the vertices of the original model. When we contract two vertices together, we create a new node in the tree and make the two vertices being contracted its children. The resulting tree provides a means of encoding the dependencies in the sequence of contractions built during simplification and has been used successfully for view-dependent reconstruction of surface approximations at run time [15, 23, 31]. But we can also think of them as hierarchies of vertex neighborhoods [9, 10]. At the leaves of the tree are the vertices of the original model: their neighborhoods correspond to their adjacent faces. When we contract two vertices together, we merge their neighborhoods and construct a new approximate neighborhood for the resulting vertex. Each node in the hierarchy corresponds to a disjoint set of vertices on the original surface. Consequently, it also corresponds to a surface region that is the union of the neighborhoods of all the vertices in its set. Note that very similar hierarchies result from the iterative application of other simplification operators, such as vertex removal.

This hierarchy of neighborhoods is itself a useful construction. For instance, Kobbelt *et al.* [20] and Lee *et al.* [21] use the resulting hierarchy to define a multiresolution parameterization of the surface. This parameterization facilitates applications such as multiresolution surface editing and morphing.

While useful, these hierarchical neighborhoods have some drawbacks for certain applications. First and foremost, local neighborhoods high in the tree are generally nonplanar (e.g., cone-shaped with the vertex at the apex). This is because any planar vertex neighborhood will be removed very early during simplification. In addition, the set of triangles on the original surface corresponding to a particular vertex in the hierarchy may have a very irregular shape. For some applications it is preferable that the shape of this region be regular, yet most standard simplification algorithms provide no means to control this shape. They are also generally unsuitable for building spatial data structures: the bounding box of a node is not guaranteed to fully contain the surface of its children.

2.2 Face Hierarchies from Clustering

Face hierarchies, which are the focus of this paper, are a natural alternative to vertex hierarchies, and their properties complement those of vertex hierarchies very nicely. Instead of iteratively merging vertex neighborhoods, we can iteratively cluster neighboring groups of faces. This allows us to avoid some of the specific drawbacks of vertex hierarchies listed above. Because the clusters are disjoint sets of faces, they partition the surface in a well-defined way and consequently have a well-defined surface area and perimeter. Face clusters are also more likely to have a single normal which is a good match for the surface normals of the faces in the cluster, and thus be more likely to be nearly planar.

To highlight the difference between these kinds of hierarchies, consider the example of a cube. Near the root of a vertex hierarchy, there will be a level at which we have eight "conical" neighborhoods, one for each corner of the cube. In contrast, near the top of a face hierarchy we will have 6 planar face clusters, one for each face of the cube. If we are trying to formulate a single planar approximating element, or a single normal for the entire region, we will get a much better result with the face hierarchies than with the vertex hierarchies. And because the surface geometry never changes, we can trivially guarantee that bounding volumes within the hierarchy fully enclose the surface of all the nodes below them.

3 FACE CLUSTERING ALGORITHM

To construct a face hierarchy, we begin by forming the *dual graph* of the surface. The dual graph is defined by mapping every face of the surface to a node in the dual graph, and connecting two dual nodes by an edge if the corresponding faces are adjacent on the surface. While the algorithm we present can be applied to non-manifolds, we will assume that the surface is a manifold with boundary[1] for efficiency reasons. Consequently, every surface edge has at most 2 adjacent faces. This guarantees that the complexity of the dual graph will not be too great; the number of dual edges will be no greater than the number of edges in the surface mesh. To simplify the discussion, we will also assume that all input polygons have been triangulated; thus, every dual node will have at most 3 neighbors.

In the dual graph, each node will correspond to a *face cluster*: a connected set of faces that have been grouped together. For the initial dual graph, each cluster consists of a single face of the input model, and these clusters will form the leaves of the hierarchy. An edge contraction in this graph merges two dual nodes into one. This corresponds to grouping their associated faces, which must necessarily be adjacent, into a single cluster. Thus in general, dual edge contraction corresponds to merging two adjacent face clusters into a single cluster. Figure 2 illustrates a simple example. The underlying mesh and the dual graph are shown in dashed and solid lines, respectively. 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 contracting a single

[1]Recall that a manifold with boundary is a surface all of whose points have a neighborhood which is topologically equivalent to a disk or half-disk.

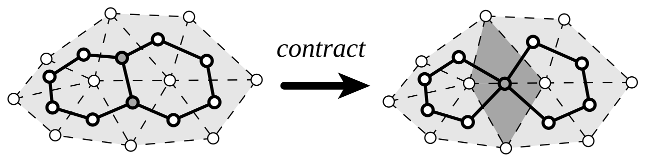


Figure 2: Edge contraction in the dual graph (shown with solid lines). The two faces of the surface (shown with dashed lines) corresponding to the endpoints of the dual edge are merged to form a single face cluster.

dual edge, the two darkened triangles have been merged into a single cluster.

To construct a complete hierarchy, we use a simple greedy procedure very similar to existing simplification algorithms. Each dual edge is assigned a "cost" of contraction, and the system iteratively contracts the dual edge of least cost. After each iteration i, we will have constructed a partition N^i of the surface into disjoint sets of connected faces. This is in contrast to simplification, where at each iteration we would have constructed an approximate surface.

Let us emphasize that *the geometry of the original surface is not altered in any way* by this clustering process; every vertex remains in its original position and the connectivity of the surface mesh is unchanged. Instead, we begin with an initial surface partition where every face belongs to its own singleton cluster. The process of iterative dual contraction produces a sequence of partitions with successively fewer clusters. If run to completion, this will produce a single face cluster for each connected component of the surface. Also note that while clusters are always connected sets of faces, they need not be simple — they may have holes.

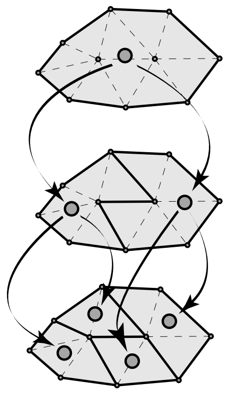


Figure 3: Part of a face hierarchy superimposed on the local geometry. The four clusters at the bottom are merged into one root cluster at the top.

Just as with iterative edge contraction, iterative dual contraction produces a natural hierarchy. When two dual nodes are contracted together (i.e., two face clusters are merged), we can make them both children of their new parent node which represents the union of their associated clusters. See Figure 3 for a simple example. At the bottom of the hierarchy are four face clusters, each of which already contain two or three faces. The two leftmost clusters are merged together, forming a single parent node. The two rightmost clusters are also merged. The two parent clusters, which together partition the mesh into two disjoint sets of faces, are merged together to produce a single root cluster which spans the entire mesh.

3.1 Related Clustering Methods

Iterative clustering, as a general class of algorithms, has been in use for decades [1]. While a substantial number of algorithms have been developed, most appear only tangentially related to the problem of building face cluster hierarchies on surfaces. In particular, most clustering algorithms have focused on the problem of clustering point sets, frequently in high dimensions, rather than clustering surface elements. And since surface elements possess adjacency and orientation which point sets do not, they require a different sort of clustering approach.

The process of face clustering which we have just described is closely related to the simplification algorithm of Kalvin and Taylor [19]. They also partitioned the surface into a set of disjoint face clusters, or "superfaces." Their algorithm was based on growing a single cluster around a random seed face, adding one face at a time until the cluster exceeded a planarity threshold. In contrast, our algorithm is based on pairwise cluster merging. The primary advantage of pairwise merging is that it produces a hierarchical structure of clusters rather than a single static partition. It also avoids the need to designate special seed faces.

DeRose *et al.* [7] proposed a related algorithm for generating hierarchies of bounding boxes on subdivision surfaces to support collision detection. Given an initial quadrilateral mesh, they iteratively merge a maximal independent set of adjoining clusters until only a single cluster remains. Since no criterion, other than adjacency, is used to select which clusters to merge, the individual clusters may or may not contain roughly coplanar elements. Faugeras and Hebert [8] iteratively merged regions of range images in order to segment them for approximation by planar or quadric patches.

Delingette [6] used a geometric construction based on the dual graph, which he called simplex meshes, to represent surfaces during reconstruction. Finally, Willersinn and Kropatsch [29] used the method of dual edge contraction to construct irregular image pyramids. While this method is designed for an entirely different domain — images rather than surfaces — it uses the same formalism of iterative dual contraction thus producing a hierarchy of pixel regions.

3.2 Dual Quadric Metric

In order to evaluate the cost of a dual contraction, we need some idea of what qualities a face cluster should have. Naturally, there are many potential criteria from which to choose. But for many applications, a good criterion is the planarity of the cluster. This means that a given face cluster can be approximated by a planar element without undue inaccuracy. We will adopt planarity as our primary criterion, modified by selected bias terms discussed in succeeding sections. As we will see, this planarity criterion can be expressed using a dual form of the quadric error metric [11, 10].

Every cluster has an associated set of faces $\{f_1, \dots, f_n\}$ and a set of points $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ determined by the vertices of these faces. Let us suppose that we want to find the least squares best fit plane to this set of points. For a plane specified by a unit normal $\mathbf{n}$ and a scalar offset d, the distance of a point $\mathbf{v}$ to this plane is $\mathbf{n}^\mathsf{T}\mathbf{v} + d$. The *fit error* of a given plane $\mathbf{n}^\mathsf{T}\mathbf{v} + d = 0$ is the average squared distance of all the points in the cluster to the plane

$$E_{fit} = \frac{1}{k}\sum_{i=1}^{k}(\mathbf{n}^{\mathsf{T}}\mathbf{v}_i + d)^2 \tag{1}$$

The least squares best plane is the one which minimizes this error. Notice that this error formula is nearly identical to the error metric used in quadric-based simplification [11, 10]. Aside from the averaging factor, the sole difference is that we are summing over a set of points with a fixed normal rather than a set of normals with a fixed point. We can define a *fit quadric*

$$P_i = (\mathbf{A}_i, \mathbf{b}_i, c_i) = (\mathbf{v}_i\mathbf{v}_i^{\mathsf{T}}, \mathbf{v}_i, 1) \tag{2}$$

$$P_i(\mathbf{n}, d) = \mathbf{n}^{\mathsf{T}}\mathbf{A}_i\mathbf{n} + 2\mathbf{b}_i^{\mathsf{T}}(d\mathbf{n}) + c_i d^2 \tag{3}$$

which requires ten coefficients to represent the symmetric 3×3 matrix $\mathbf{A}$, the 3-vector $\mathbf{b}$, and the scalar c. Now, we can expand and rewrite the inner term of E_{fit}

$$\begin{aligned}(\mathbf{n}^{\mathsf{T}}\mathbf{v}_i + d)^2 &= (\mathbf{n}^{\mathsf{T}}\mathbf{v}_i + d)(\mathbf{v}_i^{\mathsf{T}}\mathbf{n} + d) \\ &= \mathbf{n}^{\mathsf{T}}(\mathbf{v}_i\mathbf{v}_i^{\mathsf{T}})\mathbf{n} + 2d\mathbf{n}^{\mathsf{T}}\mathbf{v}_i + d^2 \end{aligned} \tag{4}$$

so that we can evaluate this error using a corresponding set of quadrics:

$$E_{fit} = \frac{1}{k}\sum_i P_i(\mathbf{n}, d) = \frac{1}{k}\left(\sum_i P_i\right)(\mathbf{n}, d) \tag{5}$$

where the addition of quadrics is defined by component-wise addition of the constituent matrices, vectors, and scalars.

Every dual node will have an associated fit quadric P and a best-fit plane $(\mathbf{n}, d)$ such that $P(\mathbf{n}, d)$ measures the planarity of the cluster associated with the node. The cost of contracting two dual nodes together is reflected by the sum of the fit quadrics of each node $(P_i + P_j)(\mathbf{n}, d)$. Note that the form of this dual quadric error (3) differs slightly from the original due to the presence of the d and d^2 terms.

This kind of least squares planarity measure is of course quite common. For example, Faugeras and Hebert [8] used it in their region merging algorithm. However, we believe its formulation as a quadric error metric is novel, and is critical for the overall efficiency of the algorithm. [19] used a closely related planarity metric which measured the maximum distance of any point to the plane, as opposed to the average squared distance. They attempted to create clusters which would respect a given L_∞ error bound whereas we attempt to minimize an L_2 error for a given number of clusters.

Finding the optimal plane which minimizes $P(\mathbf{n}, d)$ is not quite as simple as finding the optimum of the original quadric metric. The standard technique, based on principal component analysis (PCA) [18], is to construct the sample covariance matrix:

$$\mathbf{Z} = \frac{1}{k-1}\sum_{i=1}^{k}(\mathbf{v}_i - \bar{\mathbf{v}})(\mathbf{v}_i - \bar{\mathbf{v}})^{\mathsf{T}} \tag{6}$$

where $\bar{\mathbf{v}}$ is the mean of the vertices $\bar{\mathbf{v}} = \left(\sum_i \mathbf{v}_i\right)/k$. The three eigenvectors of the matrix $\mathbf{Z}$ determine a local frame with $\bar{\mathbf{v}}$ as the origin. The eigenvector corresponding to the smallest eigenvalue is the normal of the least squares best plane through the set of points $\{\mathbf{v}_i\}$. This method, which is identical to the least squares method of normal equations, is frequently used to estimate or define local tangent planes [16, 22, 5] when reconstructing surfaces from sets of points. Note that the normal computed in this fashion is only unique up to sign. In practice, it is helpful to track the average normal $\bar{\mathbf{n}}$ of all the faces in the cluster to resolve this sign ambiguity. For the remainder of the discussion, we will drop the $1/(k-1)$ averaging factor from the covariance matrix formula. This has no effect on the algorithm because we are only interested in the eigenvectors of $\mathbf{Z}$ and the relative scales of its eigenvalues.

The covariance matrix $\mathbf{Z}$ is symmetric and positive semi-definite; thus its eigenvalues will always be real and non-negative. And note that, as long as the points $\mathbf{v}_i - \bar{\mathbf{v}}$ span three dimensional space, it will have three non-zero eigenvalues. Since the points $\mathbf{v}_i$ form the vertices of a set of triangles, we know that they must at minimum span a two dimensional space, and thus $\mathbf{Z}$ has at most one zero eigenvalue. Consequently, we can always successfully derive three orthogonal vectors for our local frame. However, if two or more of the eigenvalues are equal, the optimal fitting plane is not uniquely defined. For example, all three eigenvalues will be equal if the points are uniformly distributed on a sphere. In such cases, we must simply choose any one of the possible orientations for the plane.

Looking at the definition of $\mathbf{Z}$, we might expect that it can be expressed in terms of the fit quadric P, and this is indeed the case. If we expand the equation for $\mathbf{Z}$ and collect terms, we find that

$$\mathbf{Z} = \sum \mathbf{v}_i\mathbf{v}_i^{\mathsf{T}} - k(\bar{\mathbf{v}}\bar{\mathbf{v}}^{\mathsf{T}}) \tag{7}$$

$$= \mathbf{A} - \frac{\mathbf{b}\mathbf{b}^{\mathsf{T}}}{c} \tag{8}$$

Thus, the optimal plane through the set of points can be computed directly from the corresponding fit quadric P. Its normal $\mathbf{n}$ will be the eigenvector of $\mathbf{A} - \mathbf{b}\mathbf{b}^{\mathsf{T}}/c$ corresponding to its smallest eigenvalue, and if we make the standard assumption that the plane passes through the mean, then $d = -\mathbf{n}^{\mathsf{T}}\bar{\mathbf{v}} = -\mathbf{n}^{\mathsf{T}}\mathbf{b}/c$.

3.3 Orientation Bias

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 optimal plane will not be a good fit for all the surface normals in the region. For some applications, this may be irrelevant, but for many others it is a problem we would like to avoid. To combat this problem, we will also use an additional error term which measures the average deviation of the plane normal $\mathbf{n}$ from the surface normals:

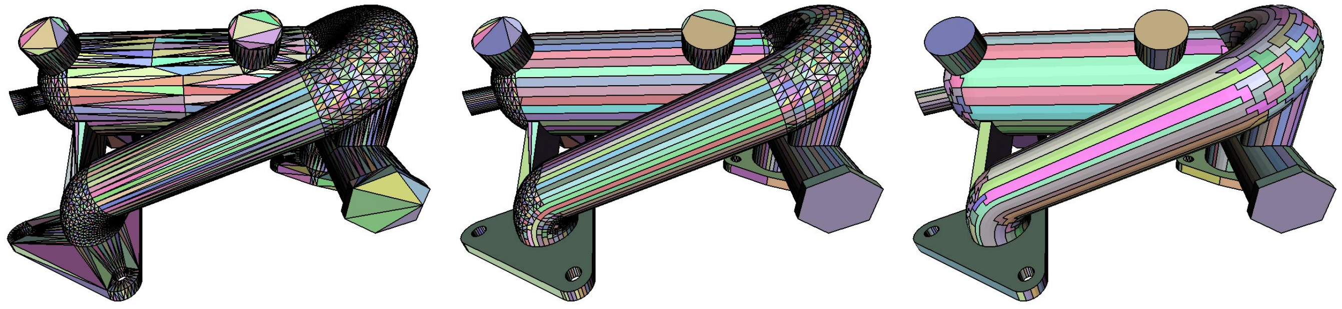
$$E_{dir} = \frac{1}{w}\sum_i w_i(1 - \mathbf{n}^{\mathsf{T}}\mathbf{n}_i)^2 \tag{9}$$

where w_i is the area of face f_i and $w = \sum_i w_i$ is the total area of the face cluster. We can write this metric as a quadric as well

$$E_{dir} = \frac{1}{w}\sum_i w_i R_i(\mathbf{n}) = \frac{1}{w}\left(\sum_i w_i R_i\right)(\mathbf{n}) \tag{10}$$

where

$$R_i = (\mathbf{D}_i, \mathbf{e}_i, f_i) = (\mathbf{n}_i\mathbf{n}_i^{\mathsf{T}}, -\mathbf{n}_i, 1) \tag{11}$$



(a) Original 11,036 clusters (b) 6000 clusters (c) 1000 clusters

Figure 4: Face cluster partitions produced by iterative pairwise merging.

and

$$R_i(\mathbf{n}) = \mathbf{n}^\mathsf{T}\mathbf{D}_i\mathbf{n} + 2\mathbf{e}_i^\mathsf{T}\mathbf{n} + f_i \tag{12}$$

Kalvin and Taylor [19] proposed a related "face-axis rule" which limited the range of orientations permitted within a cluster. As with our planarity metric, the primary difference is that we use the average deviation as a penalty rather than placing a bound on the maximum deviation. It is this formulation that allows us to use the quadric metric representation. Rather than having to traverse a list of faces every time we evaluate errors, we merely need to evaluate the value of the quadric.

Given these error metrics, the clustering algorithm has a very simple form. For every initial dual node, it computes quadrics P and R. And for every initial dual edge, we sum the quadrics of the endpoints, find the optimal plane, and evaluate its error as $E_{fit} + E_{dir}$. Following this initialization step, we place all dual edges in a heap keyed on cost, and greedily contract the minimal cost edge. Every time we contract two nodes together, we assign the quadrics of the resulting node to be the sums $P_i + P_j$ and $R_i + R_j$ and update the costs for each edge connected to the resulting node.

Some examples of the results produced by this algorithm are shown in Figure 4. The original model (a) has 11,036 faces, each of which corresponds to an individual cluster. The partition of the surface shown in (b) contains only 6000 clusters. Note how the clusters are growing along the cylindrical parts of the surface and curving along the rounded parts. This effect is even more apparent in partition (c) containing 1000 clusters. Also note how each planar region has been grouped into a single cluster. As clustering proceeds, the regions will expand over more and more of the surface. Naturally, at some point an individual region may no longer be well-approximated by any plane, since it corresponds to a significant part of the model. If run to completion, the algorithm will produce a final root cluster which contains the entire surface.

3.4 Compact Shape Bias

If planarity is our only clustering criterion, then the algorithm described above performs quite well. However, there are applications where we are also interested in the shape of these regions. In particular, we might want them to have a fairly compact shape; in other words, we might like each cluster to be as nearly circular as possible. For example, using long skinny regions in radiosity simulation would greatly increase the likelihood of a shadow discontinuity falling across the cluster. Irregularly shaped regions are problematic in applications such as simplification where we might want to retessellate clusters [19]. They also interfere with the construction of tight-fitting bounding volumes. Fortunately, it is fairly easy to add a simple compactness heuristic which significantly improves the regularity of the clusters.

Given a cluster with area w and perimeter ρ, we define the *irregularity* γ of the cluster as a ratio of its squared perimeter ρ^2 to its area w

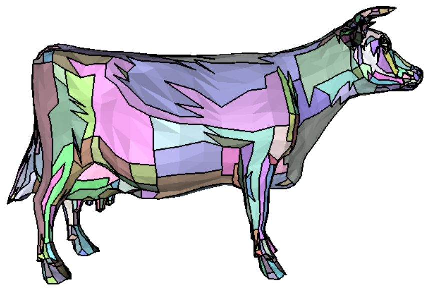
$$\gamma = \frac{\rho^2}{4\pi w} \tag{13}$$

This can also be interpreted as the ratio of the squared perimeter ρ^2 to the squared perimeter of a circle with area w. A circle will have irregularity $\gamma = 1$ and larger values of γ correspond to more irregular (less compact) regions. This definition of irregularity is fairly natural and has been widely used in fields ranging from image processing [17] to the analysis of U.S. Congressional districts to detect gerrymandering [24]. Kalvin and Taylor [19] used this definition of irregularity (without the 4π term) for directing the construction of superfaces for simplification.

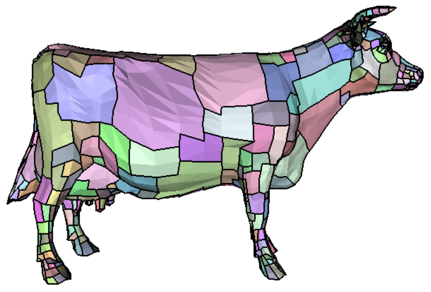
Now suppose we have have two adjacent clusters with irregularity γ_1 and γ_2, respectively. Let γ be the irregularity of the cluster formed by merging them together. We define the shape penalty as the relative change in irregularity due to merging the two regions together

$$E_{shape} = \frac{\gamma - \max(\gamma_1, \gamma_2)}{\gamma} \tag{14}$$

If the irregularity of the cluster arising from a dual contraction is worse than the two original clusters, that contraction will incur a penalty ($E_{shape} > 0$). On the other hand, if the irregularity improves it will incur a negative penalty, or bonus. Based on our experience, *requiring* the irregularity to improve at each iteration over-constrains the clustering algorithm and leads to bad results. It can also lead to cases where the greedy algorithm cannot make progress because every possible contraction would result in more irregular regions. No doubt part of the problem is due to the fact that



(a) No shape bias



(b) With bias

Figure 5: Face cluster partitions of the cow model, with 1000 clusters each. The shape bias produces much more compactly shaped clusters.

we are using a greedy algorithm which only looks one step ahead. Nevertheless, within the framework of greedy contraction, using a penalty term such as E_{shape} seems to be more effective.

In order to compute this shape penalty, we need to know both the area and the perimeter of our clusters. Clearly, the area is quite easy to track; it is merely the sum $w = w_1 + w_2$ of the constituent areas. While the perimeters are not additive, we can track them quite simply. For each cluster, we record its perimeter ρ_i in the relevant dual node. When two clusters are merged, the perimeter of the resulting cluster will be the sum of the perimeters of the constituent clusters minus twice the length of the boundary which separated them. We associate these boundary lengths with the corresponding dual edges. This allows us to compute the perimeter of the merged cluster directly from the perimeters of the merging clusters and the length associated with the dual edge being contracted.

Figure 5 demonstrates the effect of using the additional shape penalty. Clustering without bias (a) can produce highly irregular regions. Note the clusters that stretch all along the rear leg. In terms of finding clusters which are best fit by a plane, this behavior is desirable, but it does not produce compact regions. In contrast, when we use the shape bias term (b), the algorithm produces very regularly shaped regions.

3.5 Combined Error Metric

To produce the total error metric for our clustering algorithm, we combine the planarity metric E_{fit} with the two bias terms E_{dir} and E_{shape}. The cost of a dual contraction will be determined by the error metric

$$E = E_{fit} + \alpha_1 E_{dir} + \alpha_2 E_{shape} \tag{15}$$

where α_1 and α_2 are constants which must be chosen by the user. For the results reported here and in the description of our companion radiosity algorithm [30], we have simply chosen the values $\alpha_1 = 1$ and $\alpha_2 = 0$ when the shape bias is disabled and $\alpha_2 = 1$ when the shape bias is enabled. However, a more careful selection of coefficients might potentially result in improved results. Assuming that the planarity term E_{fit} is meant to be the dominant term, we could normalize the error be choosing values for α_1, α_2 which are some user-specified fraction of the diameter[2] of the model. Alternatively, if regularity of region shape is the paramount concern, we would want to select a substantially larger value for α_2.

4 APPLICATIONS

We believe that the face hierarchies produced by our clustering algorithm are potentially useful in several application areas. In particular, they are intended for use in applications which are more concerned with the aggregate properties of surface regions rather than their exact geometry.

4.1 Distance & Intersection Queries

One natural application of these face hierarchies is for constructing hierarchical bounding volumes. Recall that, for each cluster in the hierarchy, we compute a best fit plane. Given this plane, it is a fairly simple task to compute an oriented bounding box which tightly encloses the cluster. A simple, but relatively inefficient, method is to enumerate all the points in the cluster, project into the local frame defined by the plane, and compute an axis-aligned bounding box in this local frame. Alternatively, we can avoid having to traverse large sets of points by tracking the convex hull [25] of the point set [12].

Hierarchies of oriented boxes such as this can be used to efficiently measure the distance from a point to a surface, an operation common in many modeling systems. They are also an effective means of accelerating the kind of spatial queries common in applications such as ray tracing [2]. They can be constructed over both curves [3] and surfaces [4, 12]. In contrast to spatial partitions such as octrees [27], hierarchies which are attached to the surface, like those produced by our algorithm, are guaranteed to have a size linear in the size of the input model [4].

4.2 Collision Detection

Another natural application for this bounding box hierarchy is collision detection, which finds uses in a number of areas, from physical simulation to haptic interfaces. Indeed, the set of bounding boxes constructed from our face cluster hierarchy is very similar to the OBBTree structure introduced by Gottschalk *et al.* [12]. They also used PCA to compute best fit planes, and thus oriented bounding boxes. However, they produced the vertex sets by a top–down partition of the

[2]The diameter of a surface is the distance between the pair of points on the surface which are farthest apart.

vertices of the original model. In contrast, our clustering algorithm computes a bottom–up hierarchical partition of the surface and can subsequently derive bounding boxes for each region. This provides an interesting additional benefit. By design, our hierarchies produce regions which are, to the extent possible, well approximated by a plane. This means that, except at the very top of the hierarchy, most interior nodes will correspond to surface regions with a fairly well-defined orientation. Thus, we can approximate collisions against the object with collisions against bounding boxes, and the normals of the associated planes will actually allow us to predict the change in motion resulting from the collision.

Another related hierarchical representation useful for collision detection is the BOXTREE developed by Barequet *et al.* [4]. It also builds a bottom–up hierarchy of boxes by merging "adjacent" boxes. However, their algorithm is much more focused on the properties of the bounding boxes rather than the surface regions being bounded. Their definition of box adjacency is based on spatial overlap rather than the connectivity of corresponding regions on the surface. And they rank pairs of boxes for merging based on properties of the resulting boxes, such as volume, rather than any property of the local surface.

4.3 Surface Simplification

Our algorithm can also easily be used for surface simplification. Following Kalvin and Taylor [19], we can partition the surface into clusters, simplify their boundaries, and re-triangulate the resulting simplified clusters. The results of such a system would likely be similar to Kalvin and Taylor's "Superfaces" algorithm, with some key differences. While Superfaces provides guaranteed bounds on the *maximum* deviation of any point from its approximating plane, our dual quadric error metric seeks to minimize the *average* deviation without any guaranteed bounds on the maximum.

4.4 Multiresolution Radiosity

Radiosity is commonly used to simulate global illumination in diffuse environments — interactive architectural walkthroughs are a particularly popular application area. It is also a computationally intensive process that can benefit enormously from hierarchical representations. We have shown elsewhere [30] that face cluster hierarchies constructed using the algorithm developed here can be used to provide an effective means for accelerating such simulations.

The standard hierarchical radiosity method [13] has a time complexity of $O(n + k^2)$ for scenes containing k polygons and for which n elements are necessary for accurate simulation. The initial input polygons can be adaptively subdivided, producing a hierarchy of surface elements. These methods allow light transport to be simulated at an appropriate level of detail. Nearby surface patches can interact at a fairly fine grain while interactions between distant patches can be represented at a coarser level. While efficient for simple scenes where $n \gg k$, it is not as effective for very complex scenes where objects contain excess geometric detail, and hence $n \ll k$. To handle complex scenes, we need to group input polygons into higher-level clusters in addition to allowing subdivision of input polygons, thus eliminating the k^2 term from the time complexity. Consider the scene shown in Figure 9 which contains several geometrically complex models. The serpentine dragon on the table, for example, contains 870,000 triangles. But given its relatively small

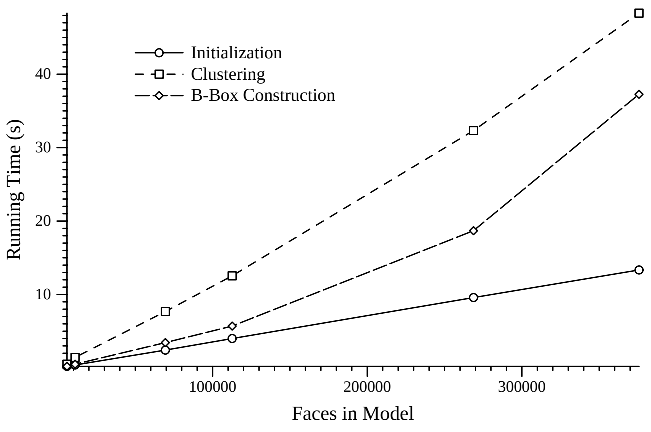


Figure 6: Running time of clustering algorithm.

size, an accurate radiosity solution could be computed on a much coarser set of surface patches than these input triangles.

One proposed method for building hierarchies above the input polygon level is volume clustering [28]. Faces of the model are grouped into spatial cells, typically axis-aligned boxes, and these cells are grouped into higher and higher levels. This approach appears to work well for clustering *objects*, such as the leaves of a tree. However, it does not perform as well for clustering *surface regions*. A particular region on a smooth surface will tend to have a dominant orientation, which is not captured by enclosing the region in an axis-aligned box. Since these volumetric hierarchies provide poor approximations of the surface geometry, the radiosity simulation is forced to descend very deep in the hierarchies to achieve an acceptable solution, thus losing much of the benefits that the hierarchy could provide. An object-aligned box will produce a much tighter fit to the surface, and it will have an orientation that reflects the orientation of the surface. The iterative face clustering algorithm that we have presented provides a convenient way to construct a hierarchy of exactly this sort of tight-fitting box.

5 RESULTS

Figure 6 illustrates the performance of our clustering algorithm on surface models of various sizes. All performance measurements for the clustering algorithm were made on a 450 MHz Intel Pentium III system, and the algorithm was run with the shape bias enabled ($\alpha_2 = 1$). The running time of the system has been decomposed into three parts. First is *initialization*, where the dual graph and initial quadrics are constructed. Second is *clustering*, where the system greedily contracts dual edges until each connected component has been merged into a single cluster. Finally there is *bounding box construction*, where each leaf in the tree propagates its vertices up to the root to compute bounding boxes for every node.

Given an input surface with n faces, both the initialization and clustering phases of the algorithm will take $O(n \log n)$ time. This is because each of them performs $O(n)$ heap operations, which themselves have $O(\log n)$ complexity. The bounding box computation phase will also have $O(n \log n)$ running time, if the resulting hierarchy is balanced. If it is not, this final phase may take $O(n^2)$ time. The algorithm we

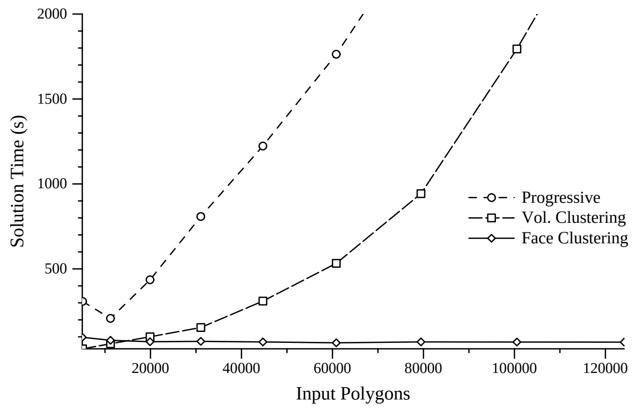


Figure 7: Running time of radiosity algorithms on the museum scene. Running time for face cluster radiosity is relatively independent of scene complexity.

have given does not attempt to produce balanced hierarchies, and the potentially greater cost of bounding box construction would seem to be apparent in the empirical data shown in Figure 6. Also note that this analysis relies on the fact that there are $O(n)$ dual edges, which is a consequence of our assumption that the input surface is a manifold (possibly with boundary), and that the degree of every node must be bounded by a constant.

Our preliminary performance analysis indicates that the Jacobi iteration procedure [26] for computing the eigenvectors of the covariance matrices can account for up to one half of the total running time. This suggests that efficiency could be increased quite a bit by applying PCA only at the upper levels of the hierarchy and using a cheaper method to compute approximating planes at the lower levels.

Figure 8 shows the result of clustering on a simple paraboloid surface with 1896 triangles. Shown with each face cluster partition are the corresponding fit planes derived by minimizing the planarity term E_{fit}. The extent of each plane is limited by the bounding box fit during the final phase of clustering. Note that the planes do fit the surface fairly well, but that they seem tilted somewhat in partition (b). When the cluster is fairly flat and extends roughly equally in all directions in the plane, PCA does not always orient the local frame in a way which would minimize the bounding box size. A much more complex surface, containing about 376,000 triangles, is shown in Figure 10. Note how the size and shape of the clusters conform to the shape of the surface: broad in smooth areas; small and narrow in highly curved areas. However, when few clusters remain (e) the algorithm is forced to choose between either irregular planar clusters or regular non-planar clusters. Since we favor planarity over regularity, large irregular regions may develop.

To demonstrate the significant advantages of face clustering for radiosity, we briefly summarize the performance of our system [30]. Figure 7 illustrates the comparative running times for the simulation phase of three different radiosity algorithms running on a 195 MHz R10000 SGI machine with 1 GB of main memory. The performance of the algorithms is shown as a function of input scene complexity, which was controlled by using surface approximations generated using the QSlim simplification package [10]. Progressive radiosity, which uses all of the input polygons during the solution process, requires rapidly increasing amounts of time and memory as the input complexity increases. Hierarchical radiosity with volume clustering requires substantially less resources, but its requirements also grow fairly rapidly with complexity. This is largely due to the fact that volume clusters do not provide an accurate fit for surface regions. Consequently, the solution computed at high levels in the hierarchy is quite inaccurate, and the algorithm is forced to descend far down in the tree to achieve an acceptable solution. In fact, it must typically descend all the way to the level of the input polygons. In contrast, the algorithm based on face cluster hierarchies exhibits essentially unchanging running times once the input complexity has risen to the level of 30,000 polygons. The algorithm is able to find a level in the hierarchy, far above the level of the input triangles, at which an acceptable solution can be computed. Because it never needs to descend deeper into the hierarchy, it requires much less time and far less resident data than the other algorithms.

6 CONCLUSION

We have described an efficient clustering algorithm which partitions a given surface into a hierarchy of disjoint face clusters. The resulting clusters are chosen so that they may be reasonably approximated with planar elements. To provide an efficient means of assessing the planarity of clusters during clustering, we have introduced a dual quadric error metric. The face cluster hierarchies produced by our algorithm can be used in a number of applications. Collision detection, ray tracing, and shape analysis for simplification appear particularly promising. We have previously demonstrated the use of these face cluster hierarchies to perform some of the largest, fastest radiosity simulations to date [30].

There are a number of promising directions in which this work could be extended. We suspect that the efficiency of the basic algorithm we have presented here may be improved by selectively using cheaper methods to compute fitting planes rather than always using PCA. Our choice of a planarity criterion was motivated by the application to radiosity, but other alternative error metrics could also be considered. A more flexible combination of the planarity and bias terms would be useful, particularly if it provided finer control over the tradeoff between them. Because certain applications might benefit from balanced hierarchies, it might also be useful to consider modifying the algorithm to provide them, either by parallel merging of a maximal independent set of dual edges [31] or by adding a balance bias to the error metric [15].

Further information, including our experimental implementation, can be found online at http://graphics.cs.uiuc.edu/~garland/research/cluster.html.

7 ACKNOWLEDGEMENTS

This research was funded in part by the National Science Foundation (grants CCR-9505472, CCR-9619853, and DMI-9813259). We would like to thank Joel Welling for arranging access to the large SGI server and Tom Stahovich for helping to fund this work.

References

[1] Michael R. Anderberg. *Cluster Analysis for Applications.* Academic Press, New York, 1973.

[2] James Arvo and David Kirk. A survey of ray tracing acceleration techniques. In Andrew Glassner, editor, *An Introduction to Ray Tracing*, pages 201–262. Academic Press, 1989.

[3] Dana H. Ballard. Strip trees: A hierarchical representation for curves. *Communications of the ACM*, 24(5):310–321, 1981.

[4] Gill Barequet, Bernard Chazelle, Leonidas J. Guibas, Joseph S. B. Mitchell, and Ayellet Tal. BOXTREE: A hierarchical representation for surfaces in 3D. *Computer Graphics Forum*, 15(3):387–96, 484, 1996.

[5] Jens Berkmann and Terry Caelli. Computation of surface geometry and segmentation using covariance techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(11):1114–1116, 1994.

[6] Hervé Delingette. Simplex meshes: A general representation for 3D shape reconstruction. In *Conf. on Computer Vision and Pattern Recognition (CVPR '94)*, June 1994.

[7] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. In *Proceedings SIGGRAPH 98*, pages 85–94, 1998.

[8] O. D. Faugeras and M. Hebert. The representation, recognition, and positioning of 3-D shapes from range data. In Takeo Kanade, editor, *Three-Dimensional Machine Vision*, pages 301–353. Kluwer Academic Publishers, 1987.

[9] Michael Garland. Multiresolution modeling: Survey & future opportunities. In *State of the Art Report*, pages 111–131. Eurographics, September 1999. http://www.uiuc.edu/~garland/papers.html.

[10] Michael Garland. *Quadric-Based Polygonal Surface Simplification*. PhD thesis, Carnegie Mellon University, CS Dept., 1999. Tech. Rept. CMU-CS-99-105. http://www.uiuc.edu/~garland/research/thesis.html.

[11] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH 97 Proc.*, pages 209–216, August 1997. http://www.uiuc.edu/~garland/research/quadrics.html.

[12] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. In *Proceedings SIGGRAPH 96*, pages 171–180, 1996.

[13] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics (SIGGRAPH '91 Proc.)*, 25(4):197–206, July 1991.

[14] Hugues Hoppe. Progressive meshes. In *SIGGRAPH '96 Proc.*, pages 99–108, August 1996. http://research.microsoft.com/~hoppe/.

[15] Hugues Hoppe. View-dependent refinement of progressive meshes. In *SIGGRAPH 97 Proc.*, pages 189–198, August 1997. http://research.microsoft.com/~hoppe/.

[16] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 71–78, July 1992. http://research.microsoft.com/~hoppe/.

[17] Anil K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall International, London, 1989.

[18] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[19] Alan D. Kalvin and Russell H. Taylor. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Appl.*, 16(3), May 1996. http://www.computer.org/pubs/cg&a/articles/g30064.pdf.

[20] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proc. SIGGRAPH 98*, pages 105–114, 1998.

[21] Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar, and David Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. In *Proc. SIGGRAPH 98*, pages 95–104, 1998.

[22] Ping Liang and John S. Todhunter. Representation and recognition of surface shapes in range images: A differential geometry approach. *Computer Vision, Graphics, and Image Processing*, 52:78–109, 1990.

[23] David Luebke and Carl Erikson. View-dependent simplification of arbitrary polygonal environments. In *SIGGRAPH 97 Proc.*, pages 199–208, August 1997.

[24] Richard G. Niemi, Bernard Grofman, Carl Carlucci, and Thomas Hofeller. Measuring compactness and the role of a compactness standard in a test for partisan and racial gerrymandering. *Journal of Politics*, 52(4):1155–1181, 1990.

[25] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: an Introduction*. Springer-Verlag, New York, NY, 1985.

[26] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Second edition, 1992. http://www.nr.com.

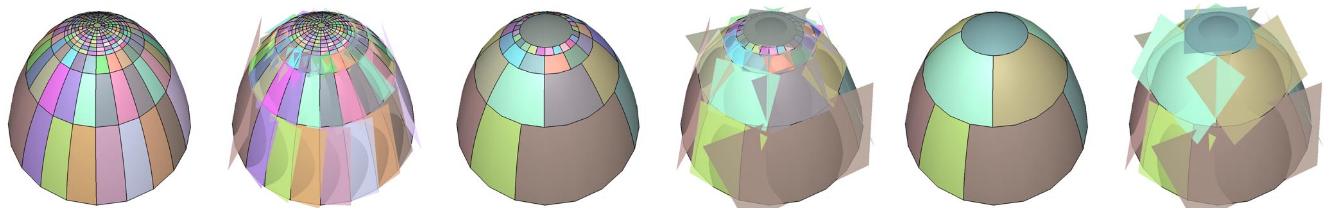
[27] Hanan Samet. *Applications of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1990.

[28] Brian Smits, James Arvo, and Donald Greenberg. A clustering algorithm for radiosity in complex environments. In *Proceedings of SIGGRAPH '94*, pages 435–442, July 1994.

[29] Dieter Willersinn and Walter G. Kropatsch. Dual graph contraction for irregular pyramids. In *Proc. 12th IAPR Intl. Conf. on Pattern Recognition*, volume III, pages 251–256, October 1994.

[30] Andrew J. Willmott, Paul S. Heckbert, and Michael Garland. Face cluster radiosity. In *Eurographics Workshop on Rendering*, June 1999. http://www.cs.cmu.edu/~ajw/paper/fcr-eg99/.

[31] Julie C. Xia and Amitabh Varshney. Dynamic view-dependent simplification for polygonal models. In *Proceedings of Visualization '96*, pages 327–334, October 1996.



(a) 250 clusters (b) fit planes (c) 50 clusters (d) fit planes (e) 12 clusters (f) fit planes

Figure 8: Face clusters & associated fit planes on a paraboloid. Corresponding clusters and planes are drawn in same color.

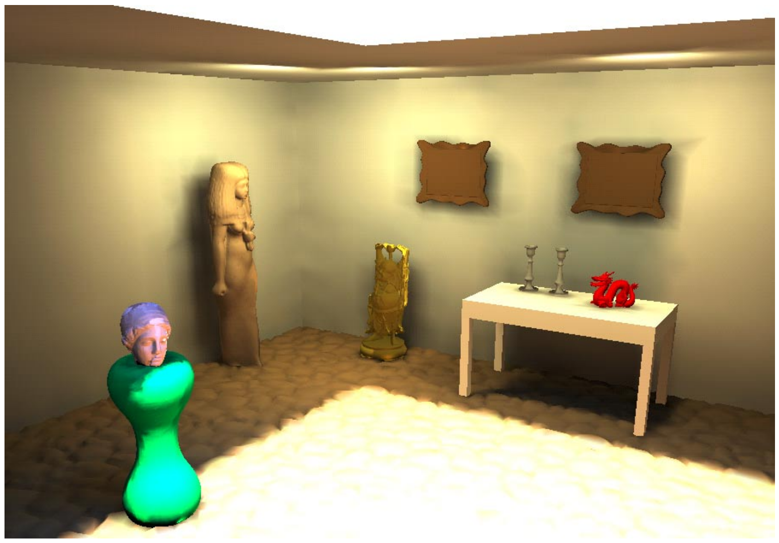
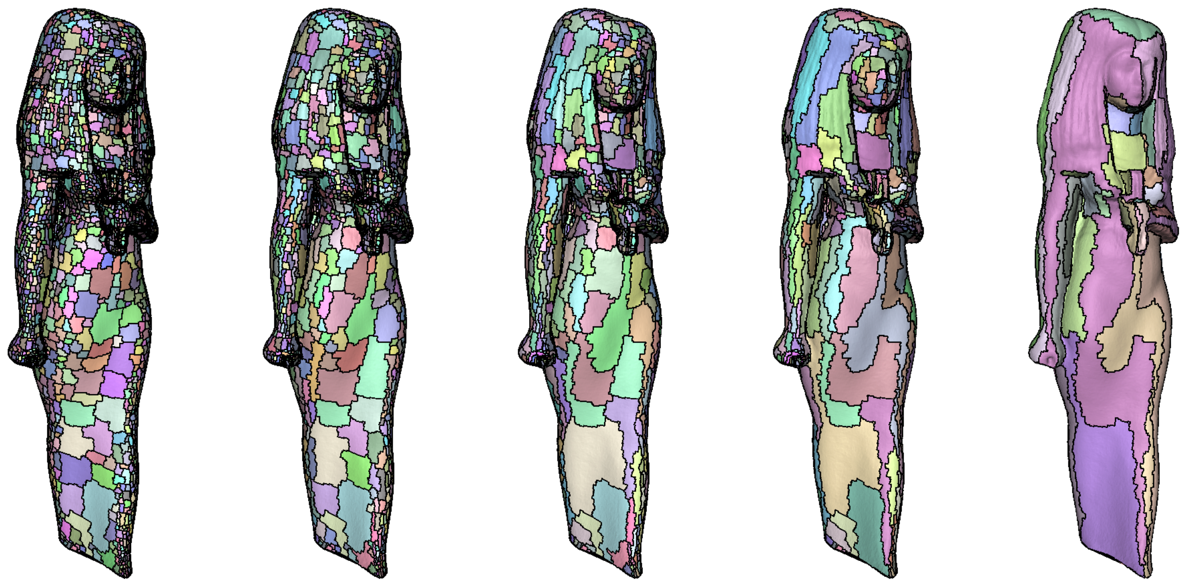


Figure 9: Face cluster radiosity solution, computed in only 3 minutes, for a scene with 2.7 million input polygons.



(a) 10,000 clusters (b) 5000 clusters (c) 2500 clusters (d) 1000 clusters (e) 100 clusters

Figure 10: Face clusters computed for Isis statue, composed of 375,736 triangles. Cluster shape adapts to surface shape.