Multiresolution Feature Extraction from Unstructured Meshes

Andreas Hubeli, Markus Gross

Department of Computer Science
ETH Zurich, Switzerland

Abstract

We present a framework to extract mesh features from unstructured two-manifold surfaces. Our method computes a collection of piecewise linear curves describing the salient features of surfaces, such as edges and ridge lines. We extend these basic techniques to a multiresolution setting which improves the quality of the results and accelerates the extraction process. The framework is semi-automatic, that is, the user is required to input a few control parameters and to select the operators to be applied to the input surface. Our mesh feature extraction algorithm can be used as a preprocessor for a variety of applications in geometric modeling including mesh fairing and subdivision.

CR Descriptors: Surface Representations, Geometric Modeling, Triangle Decimation, Multiresolution Models, Feature Extraction.

1 Introduction

1.1 Motivation

Recent advances in acquisition systems have resulted in the ready creation of very large, densely sampled surfaces, usually represented as triangle meshes. The impossibility of real-time interaction with these large models has motivated many researchers in the computer graphics community to design advanced mesh processing methods including subsampling, restructuring, fairing and others.

The early approaches, such as the *vertex removal* algorithm of W. Schröder [17] or the *progressive mesh* algorithm of H. Hoppe [10], use local error norms to construct multiresolution approximations of meshes by iteratively removing information from the input mesh. More recent representations are based on the generalization of fairing techniques from signal processing [18], [13], [11], or on subdivision surfaces [14], [19]. These approaches combine advanced operators with multiresolution techniques to enable interaction with large datasets and provide additional functionality, such as editing.

E-mail: hubeli@inf.ethz.ch
grossm@inf.ethz.ch

Address: Department of Computer Science
ETH Zentrum
CH - 8092 Zurich

In this paper we investigate a related problem: the detection of *mesh features*. Our goal is to extract piecewise linear features from meshes which can then be used to construct more sophisticated multiresolution representations. The most important advantage of our method is that we can force modeling algorithms, such as subdivision and fairing, to retain feature information including sharp edges or ridge lines.

1.2 Previous Work

In his pioneering work [2], Canny constructed an optimal filter to detect and extract certain types of features from images. The criteria used to design the filter include detection performance, sharp localization of the features and a unique detection of edges.

The theory of snakes [12] uses deformable models to track features. A polygonal curve is assigned an energy function and is deformed until its associated energy is minimized. The choice of particular energy functions ensures that the polygonal curve traces a feature to its end configuration.

Anisotropic diffusion, as used in [16], is a powerful tool that allows to remove high-frequency noise while preserving the feature information. Thus, the technique enables the construction of robust edge detection algorithms. This operator has been extended to the non-parameterized setting of triangle meshes with arbitrary connectivity in [3].

A problem related to feature extraction is image segmentation [9], where input images must be subdivided into regions that possess similar characteristics. A robust segmentation algorithm was used to extract features being defined as the piecewise linear curves that bound different regions in the image.

Most of the techniques discussed in this section apply to images. There are several advantages of handling images over triangle meshes: images have both a regular connectivity and a well known parameterization, properties that triangle meshes do not possess. These differences complicate the extension of these techniques to the more general domain of unstructured triangle meshes.

1.3 Paper Organization

The paper is organized as follows: in section 2 we provide some basic definitions and give an overview of the framework. In section 3 we present the first major component of our framework, the set of classification operators, followed in section 4 by a discussion of the second component, the detection operators. In section 5 we describe a multiresolution feature extraction technique. In section 6 we present some of the results we obtained using this framework and we discuss some application domains. We will conclude the paper with the description of some future challenges.

2 Overview of the Technique

The goal of the framework presented in this paper is to extract a set of mesh features from two-manifold polygonal meshes with arbitrary connectivity. To this end, we first formalize the notion of a mesh feature and the domain of our operators:

Definition: A *mesh feature* is a piecewise linear curve that describes an important characteristic of the input mesh. We restrict any mesh feature as being represented by a collection of edges of the mesh. That is, our extraction operators are not required to modify the connectivity of the mesh.

Definition: a surface is a *two-manifold with boundaries* if all its points have an open neighborhood homeomorphic to either $\Re^2$ or $\Re^2_+$.

The approach that we introduce is independent of the feature semantics, i.e. the operators are not trained to recognize particular patterns. Instead, we decided to use application-neutral operators which are based on notions such as Laplacian or curvature. The resulting framework is subdivided into two components, as shown in figure 1:

- In a *classification phase*, every edge in the model is assigned a weight proportional to the probability that the edge lies on a mesh feature. The operators used in this step are not based on any assumption on the input mesh, except that it has to be a two manifold surface with boundaries.
- In a *detection phase*, mesh features are reconstructed from the information computed in the previous step. By our definition, this step constructs piecewise linear curves, defined as collections of edges.

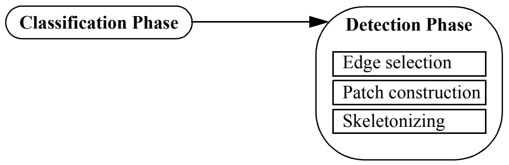

Figure 1: Overview of the feature extraction framework.

The distinction between the classification and detection operators has several advantages: first, it is possible to easily swap between classification operators and thus choose the best suitable operator for the particular input mesh. Furthermore, if additional information on the application domain is available, new optimized operators can be included into the framework more easily.

In the next two sections we will describe the two components in detail and present the set of operators that we constructed and tested in our framework. We illustrate the results by applying them to surfaces with different properties.

3 Classification Phase

The goal of the classification phase is to assign a *weight* to every edge in the input mesh. Ideally, the weight should be proportional to the probability that the edge belongs to a mesh feature: edges close to mesh features should be assigned larger weights than edges that are farther away. The information computed in this first phase will then be used to extract a set of the most important edges, which in turn will be used to reconstruct the mesh features.

In the classification step we are faced with different problems. First, the operators must be capable to handle meshes with different resolutions. Some meshes are highly optimized meaning that the number of triangles is reduced to a minimum, while others are oversampled. In addition, the classification process is aggravated by the presence of high frequency noise in the input meshes, which could mistakenly be recognized as useful information.

To alleviate some of these problems we provide a set of different operators, each with different properties, and better suited to handle certain classes of meshes. In addition, we can optionally prefilter the input meshes using standard operators, such as [13], [8] and [4].

In the remainder of this section we will introduce the operators we use in our framework and discuss their capabilities and limitations.

3.1 Second Order Difference (SOD)

The SOD is the simplest classification operator. It assigns a weight to every edge e in the mesh proportional to the dihedral angle defined by the normals of its two adjacent triangles. The idea is based on the second order difference operator constructed in [8], which was used to fair meshes of arbitrary connectivity. The operator, described by equation (1), is locally bound and can be evaluated efficiently.

$$w(e) = \cos\left(\frac{\boldsymbol{n}_i}{\|\boldsymbol{n}_i\|} \cdot \frac{\boldsymbol{n}_j}{\|\boldsymbol{n}_j\|}\right)^{-1} \quad (1)$$

The variables $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ correspond to the normals of the two triangles that share edge e, as shown in figure 2.

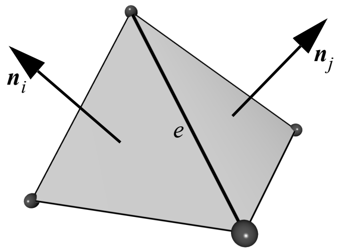

Figure 2: Support of the SOD classification operator.

This technique is best suited for coarse, pre-optimized meshes. However, SOD performs poorly on highly detailed or noisy meshes, since all computations are carried out within a small region of support.

3.2 Extended Second Order Difference (ESOD)

The ESOD operator extends the SOD operator by using a larger support to evaluate the weight of an edge e. Instead of defining $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ as the normals of the two neighboring triangles of e, we define them as the average normals computed from the triangles on the one-ring of the vertices $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ opposite to e and apply the them in equation (1). The extended support of ESOD is illustrated in figure 3.

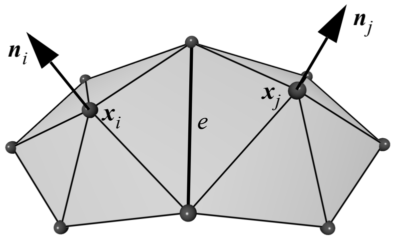

Figure 3: Support of the ESOD operator.

The increased support of the operator has the expected consequences: the influence of noise on the classification process is attenuated. However, as a the support of the operator is larger and cannot be adapted to the input mesh, ESOD does not perform well on very coarse meshes.

3.3 Best Fit Polynomial (BFP)

The BFP operator uses a different philosophy than the previous operators, in that it makes use of a parameter domain associated with every edge in the mesh. The weight assigned to an edge e is computed as follows: first, a subset of the mesh vertices are projected onto the parameter domain and a best fit polynomial $p(u)$ of degree n is computed. The curvature of the (planar) polynomial is then evaluated at the parameter position of e, as described by equation (2):

$$w(e) = p''(e) \tag{2}$$

The major challenges of this approach are the definition of the parameter domain and the proper projection of the set of vertices from 3-space. An intuitive definition of the parameter space is given in figure 4.

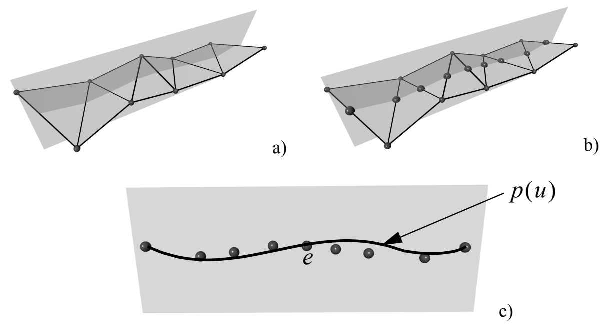

Figure 4: The BFP operator:
a) Parameter plane.
b) Intersection between the parameter plane and the mesh.
c) Best fit polynomial fitted in parameter space.

We propose to set the parameter plane to be perpendicular to the vector defined by the edge e. A unique plane is defined by requiring the midpoint of e to lie on the plane. The points used in the best fit process are computed from the intersection of the plane with a set of neighboring triangle edges, as shown in figure 4.b. The most important advantage of this strategy is that the support of the operator can be chosen freely and that it can be adapted locally for each edge. An additional degree of freedom is given by the degree of the fitting polynomial which can be adjusted to the size of the support of the operator.

This approach has the advantage of being very flexible, because the support can be adapted both globally and locally. Thus, it is less influenced by noise which is filtered out during the best fit process. Furthermore, it has the potential to be used for any type of mesh, provided that an appropriate set of parameters is chosen. The main disadvantage of the approach is that its computational cost is higher than the cost of the previous operators.

3.4 Angle Between Best Fit Polynomials (ABBFP)

The ABBFP operator is a variation of the BFP operator which is also based on best fit polynomials. As for the previous approach, polynomials are fitted in the parameter domain of every edge e (figure 5). The ABBFP operator fits two polynomials: one for the vertices that lie on one side of e in the parameter domain and another for the vertices lying on the other, as illustrated in figure 5.c. The weight assigned to e is chosen to be proportional to the angle between tangents of the two curve evaluated at the parameter position of e. The weight is then computed according to equation (3):

$$w(e) = \cos\left(\frac{(1, p_l'(e))}{\|(1, p_l'(e))\|} \cdot \frac{(1, p_r'(e))}{\|(1, p_r'(e))\|}\right)^{-1} \tag{3}$$

As for the BFP operator, the user is allowed to specify both the size of the support and the degree n of the polynomial used in the best fit process. A potential advantage of this variant is that it is adept in capturing discontinuities at the parameter position of e, which hints at the presence of a mesh feature in the neighborhood of e.

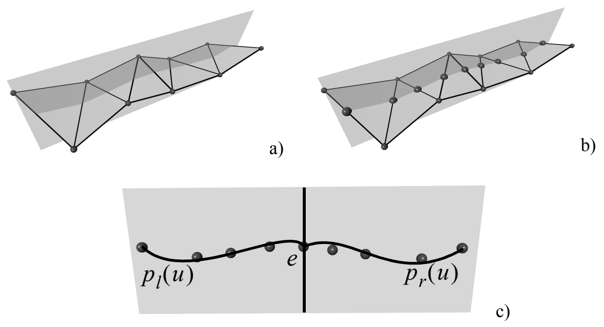

Figure 5: The ABBFP operator:
a) Parameter plane.
b) Intersection between the parameter plane and the mesh.
c) Two polynomials are fitted on both sides of e and the angle between their tangents at e is measured.

3.5 Comparison of the Operators

In order to compare the different operators that were described in this section, we apply them to two different types of meshes.

Figure 6.a depicts a coarse mesh that represents a motor part using few triangles. Figure 6.b-c illustrate the result computed using the SOD and BFP operators respectively. In this example the most efficient operator is SOD, since it can capture all the important information efficiently and it can be evaluated more effectively than BFF. The BFP operator is capable of generating good results, but its parameters must be tuned carefully.

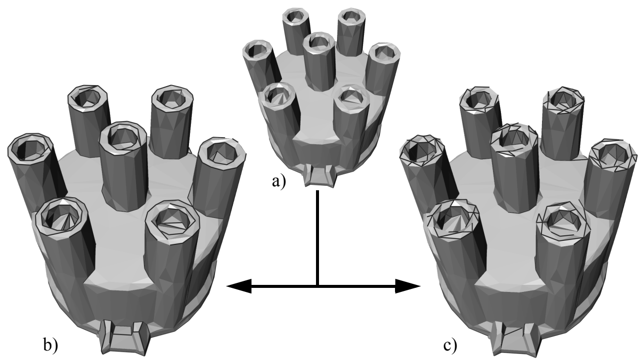

Figure 6: Classification operators applied to an optimized mesh:
a) Input mesh.
b) Result computed using the SOD operator.
c) Result computed using the BFP operator.

Figure 7.a shows a highly detailed mesh that represents a geological surface using a large number of triangles. The results computed using the SOD and BFP operators are depicted in figure 7.b-c respectively. The SOD operator recognized important regions in the mesh, but noisy regions were also recognized as a result of the limited support of the operator. The BFP operator performed better, since its support has been tuned to filter out most of the noise present in the surface.

For both examples presented in figure 6 and figure 7 the same parameters were chosen for the hysteresis which was used to select the set of most important edges. A description of the hysteresis function is given in section 4.1.

4 Detection Phase

The classification step discussed in the previous section assigns a weight to every edge in the input mesh, which is proportional to the probability that the edge belongs to a mesh feature. The detection phase analyzes these values and reconstructs a set of important mesh features in four steps:

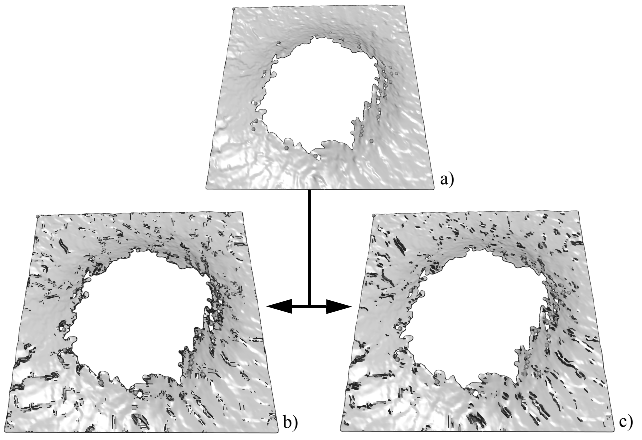

Figure 7: Classification operators applied to a highly detailed mesh:
a) Input mesh.
b) Result computed using the SOD operator.
c) Result computed using the BFP operator.

- First, the set of the most important edges is computed. The importance of an edge is defined both by its weight and by the weight of the neighboring edges.
- The set of edges is analyzed to construct a set of patches. A patch is a mesh region that is likely to contain one or more mesh features.
- The mesh features are extracted from the patches using a skeletonizing algorithm which iteratively simplifies every patch to a collection of piecewise linear curves.
- Optionally, unimportant mesh features can be removed from the result. The importance of a mesh feature is computed from its length and from the weights associated with its edges.

In the next four subsections we will describe each of these steps in detail and illustrate the results with an example.

4.1 Selection of Important Edges

The first step in the detection phase is crucial, since it identifies the edges close to the mesh features. This process is heavily mesh dependent: meshes contain a varying number of mesh features and their edge density is variable. Furthermore, the process is also influenced by the user who might be interested in only a subset of the mesh features. Hence, we require the user to specify the bounds of the hysteresis function used to mark the set of important edges.

Hysteresis Thresholding

This approach requires two user-defined values which serve as a lower bound B_l and as an upper bound B_u of the hysteresis. An edge e in the mesh is selected if it satisfies one of two conditions:

Condition 1: the weight $w(e)$ of e is larger than the upper bound B_u of the hysteresis function:

$$w(e) \geq B_u \tag{4}$$

Condition 2: The weight $w(e)$ of e is larger than the lower bound B_l of the hysteresis function and the set of neighboring edges $N(e)$ contains a selected edge e':

$$w(e) \geq B_l \wedge \exists e';\ e' \in N(e);\ e' \text{ is selected} \tag{5}$$

If neither condition is satisfied, the edge e is discarded.

Figure 8 illustrates the binary hysteresis function with upper bound B_u and lower bound B_l and as implemented in our framework:

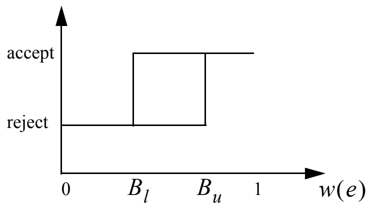

Figure 8: Hysteresis function for the threshold values B_l and B_u.

An hysteresis has several advantages versus standard thresholding strategies: first, thresholding can be simulated by the hysteresis function by setting the upper and lower bounds to the same value:

$$B_l = B_u \tag{6}$$

Furthermore, the hysteresis clusters edges better than thresholding. As a result, the patching algorithm is capable of generating less patches with larger area, which in turn allows us to extract mesh features more robustly.

Figure 9 illustrates the set of patches obtained by using different bound values for the same set of input weights. If the upper and lower bounds are set to the same value, as in figure 9.b, the hysteresis process is reduced to standard thresholding and the selected edges have not been clustered well. The use of different bounds, as illustrated in figure 9.c, enables us to remove some of the isolated edges and generate better clusters. The choice of a very large upper bound and a small lower bound allow us to select edges close to the most important features, as shown in figure 9.d.

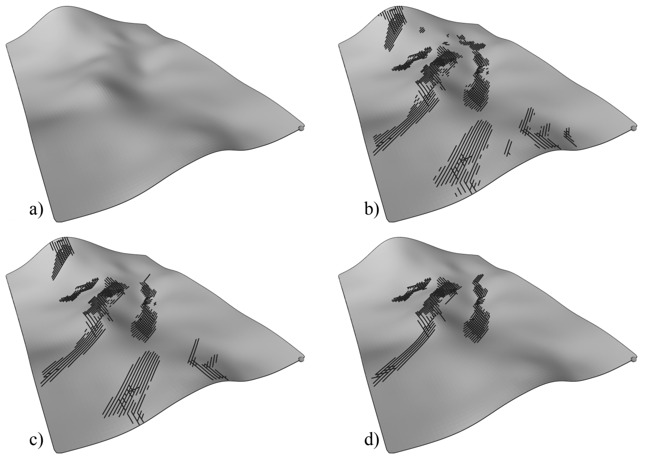

Figure 9: Edges selected by the hysteresis function:
a) Input mesh.
b) Result generated using the parameters $B_u = 0.92$, $B_l = 0.92$.
c) Result generated using the parameters $B_u = 0.96$, $B_l = 0.92$.
d) Result generated using the parameters $B_u = 0.995$, $B_l = 0.9$.

4.2 Patch Construction

The construction of the patches is actually not required in the detection phase. However, the use of patches allows us to apply extensions of standard skeletonizing algorithms from the field of computer vision [6]. This is advantageous, since the problem has been well studied in that field and many robust skeletonizing algorithms are available.

As mentioned in section 4.1, the hysteresis operator has good clustering capabilities. The goal of the patching algorithm is to transform these clusters into uniform patches, where all edges are marked. Additionally, the algorithm is not allowed to increase the size of the clusters, but only fill the interior. The result is a simple condition that is checked to select additional edges and insert them into the set created in section 4.1:

Condition: An edge e is inserted into a patch if it was marked in the previous step, or if both of its endpoints belong to other edges present in the patch, regardless of the weight $w(e)$ of e.

This condition is similar to the second condition used by the hysteresis function. Again, edges with a small weight are inserted into the set of patches if they are close to important edges. The requirement that important edges are present on both sides of e ensures that patches do not grow unnecessarily large. The absence of requirements for the weight $w(e)$ of e guarantees that all unmarked edges in the interior of a cluster will be added to the cluster thus resulting in more uniform patches.

Figure 10 depicts the result obtained by our patching algorithm. The algorithm is initialized with the set of edges displayed in figure 10.a. The insertion of edges according to the condition specified in this section yields the set of patches illustrated in figure 10.b.

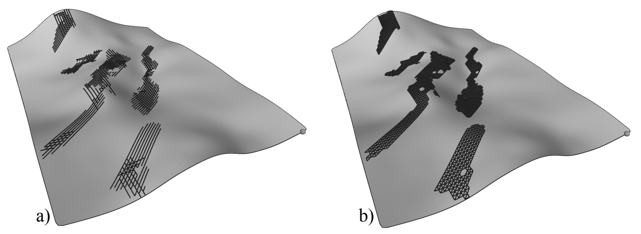

Figure 10: Patch construction:
a) Set of edges selected by the hysteresis function.
b) Patches generated by the framework.

Note that the holes in the patches depicted in figure 10.b are a consequence of the requirement that patches should not grow more than needed.

4.3 Skeletonizing

The most important step in the detection phase consists in the extraction of the final mesh features from the patches computed in the previous step. We accomplish this goal using a skeletonizing algorithm similar to the ones used computer vision. In particular, we constructed a thinning technique: patches are thinned to a set of mesh features by burning the edges from the boundary of the patches down to a set of piecewise linear curves. We do not use existing algorithms from computer vision, since the skeletonizing algorithm is expected to handle meshes with arbitrary connectivity and not only height field data. The thinning algorithm is described by the following pseudo-code fragment:

```
void patchThinning(list< int > patch) {
  for all edges e in patch
    if (isBoundaryEdge(e) == true)
      edgeList.insert(e);

  while edgeList is not empty do {
    e = edgeList.front();  // Retrieve the first edge
    edgeList.pop_front();  // Remove it from the list

    if(belongsToMeshFeature(e) == false) {
      removeFromPatch(e);
      edgeList.insert(newBoundaryEdges);
    }
  }
}
```

The thinning operation is initialized by inserting all the edges that lie on the boundary of a patch into a linked list. The function `isBoundaryEdge` checks the following condition for the edge e:

Condition: consider an edge e that belongs to a patch and its two adjacent triangles t_1 and t_2. If any of the other four edges of t_1 and t_2 does not belong to the patch, then e is a boundary edge.

The boundary edges are extracted from the list one at a time, and they are analyzed. The function `belongsToMeshFeature` inspects two conditions to decide whether an edge e belongs to a feature:

Condition 1: If only one of the endpoints of e belongs to another edge in the patch, e is only preserved if the endpoint belongs to only one other marked edge. If the endpoint belongs to multiple edges in the patch, then it must be removed. This condition enables us to remove edges that are perpendicular to the mesh feature being extracted.

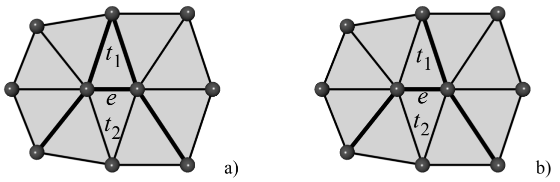

Figure 11: Thinning operator; selected edges marked as a bold polyline:
a) Edge *e* is removable.
b) Edge *e* is not removable, since it would disconnect the feature locally.

Condition 2: If both endpoints of e belong to other edges in the patch, then e can only be removed if it does not disconnect the patch locally. The patch is not disconnected by the removal operation if and only if one of the two adjacent triangles t_1 and t_2 of e is defined by three marked edges. Consider the example in figure 11: the edge e in the configuration displayed in figure 11.a can be removed safely, since all the edges in t_1 are marked. Conversely, the edge in figure 11.b cannot be removed without disconnecting the patch locally.

If an edge is removed from a patch, two new edges will become boundary edges and they must be inserted into the list. After the edge e has been analyzed, the next edge is extracted from the list. This process continues until the list is empty.

The end configuration is guaranteed to be a collection of piecewise linear curves. Since the patches were not disconnected in the thinning process, the mesh features are also guaranteed to be connected. The mesh features can intersect, i.e. two or more edges that belong to mesh features can intersect at a vertex. Therefore, complex features can be recognized and extracted by the algorithm.

Figure 12 illustrates the mesh features extracted from the set of patches computed in the previous section. The input data is visualized in figure 12.a and the set of output mesh features in figure 12.b. By analyzing the result thoroughly, one can notice that one of the patches has been thinned into two mesh features implying that the patch has been disconnected. The two features were connected by a third feature that has been removed from the result.

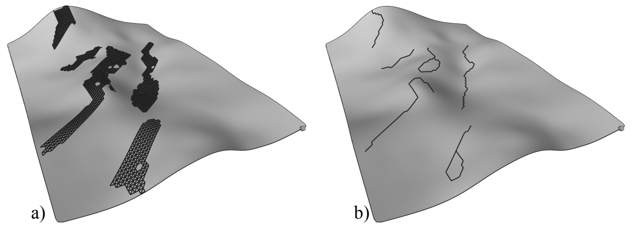

Figure 12: Mesh features generated by the thinning algorithm:
a) The set of patches used to initialize the algorithm.
b) The set of resulting mesh features.

4.4 Importance Function

The set of mesh features can be optionally reduced so that only "important" features are returned to the user. In our opinion, the importance of a mesh feature is determined by two factors: its length, i.e. the number of edges present in the mesh feature and the average weight of the edges. The elimination of "unimportant" features is performed by ranking the mesh features according to an importance function such as

$$I(F_i) = |F_i| \cdot \frac{1}{n} \sum_{e \in F_i} w(e) \tag{7}$$

The term $|F_i|$ describes the length of the mesh feature F_i and $w(e)$ the weight of an edge e that belongs to F_i.

Next, the mesh features with smallest importance are removed from the result and the remaining features are returned to the user.

5 Multiresolution Feature Extraction

The classification and detection techniques that we discussed in the previous two subsections are efficient and generate meaningful mesh features. However, as already shown in image processing [15], the use of multiresolution techniques can improve the overall framework, both in terms of efficiency and in terms of quality of the results. In particular, the use of a multiresolution representation of the input mesh enables us to better capture low frequency mesh features than by extending the support of the classification operators.

5.1 Multiresolution Representation

The literature offers many different multiresolution representations for triangulated two-manifold surfaces, based on operators such as vertex removal [17] and edge collapse [10], [5]. We selected the progressive mesh algorithm in our framework, since the edge collapse and vertex split operators naturally complement our own operators.

In addition, we obtain the following advantageous properties:

- The coarse approximation of the input mesh only contains the structural information on the mesh, i.e. the collapse operator preserves the salient features of the mesh during the simplification process.
- During the refinement process, the vertex split operations can be analyzed on the fly to update the set of mesh features. Thus we obtain the full-resolution features after all the vertices have been re-inserted into the mesh.
- A multiresolution approach has the potential of accelerating the extraction process. This can be accomplished by computing the weight of a subset of the edges of the input surface that are close to mesh features.

5.2 Multiresolution Feature Extraction

As mentioned in the previous subsection, multiresolution and in particular the edge collapse operator can be used effectively to extract features incrementally from a full resolution mesh. This process is implemented in three steps:

I. Given an input mesh, its progressive mesh representation is first constructed. The result consists of a coarse approximation of the mesh and a set of vertex split operations that allow us to reconstruct the original surface.

II. The techniques described in section 3 and section 4 are applied to the coarse representation of the mesh to extract the most important mesh features. This operation can be performed efficiently, since both the classification and the detection steps are applied only to a subset of the edges of the input mesh.

III. Finally, the input mesh is reconstructed from the coarse mesh by re-inserting the vertices and edges using the set of vertex split operations. During this process, it is crucial to update the mesh feature information which needs to adapt to the changes of the underlying mesh. The set of masks needed to perform these updates is illustrated in figure 13.

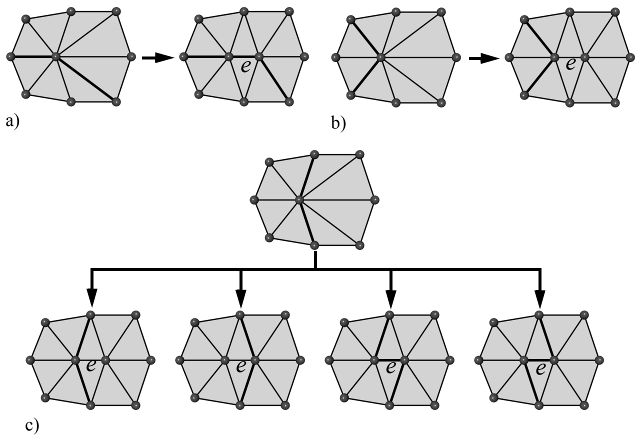

Figure 13: Feature update masks; feature displayed as a thicker polyline:
a) The new edge *e* is inserted into the feature.
b) The new edge *e* is not inserted into the feature.
c) The new edge *e* changes the shape of the feature locally.

During the insertion of a vertex $\boldsymbol{x}_j$ which is split from a vertex $\boldsymbol{x}_i$, the neighborhood of the vertex split is analyzed. If $\boldsymbol{x}_i$ belongs to a mesh feature, we need to check whether the feature needs to be modified. In principle, many different configurations could be investigated in order to compute the best possible update. In practice however, the masks displayed in figure 13 proved to be adequate and only three configurations must be considered:

- The case depicted in figure 13.a is unambiguous: the new edge e must be included into the mesh feature, otherwise the insertion operation would split the feature into two components.
- The second case shown in figure 13.b is also straightforward: the insertion of the new edge e does not affect the feature. Therefore e is not inserted into the feature.
- The third case shown in figure 13.c is more complex. The two edges that are split by the vertex split operation belong to the mesh feature. Therefore, the mesh feature can be updated in four different ways. The choice of the best path for the mesh feature is determined by analyzing the weights of the edges locally. The mesh feature is updated using the path with the largest average weight.

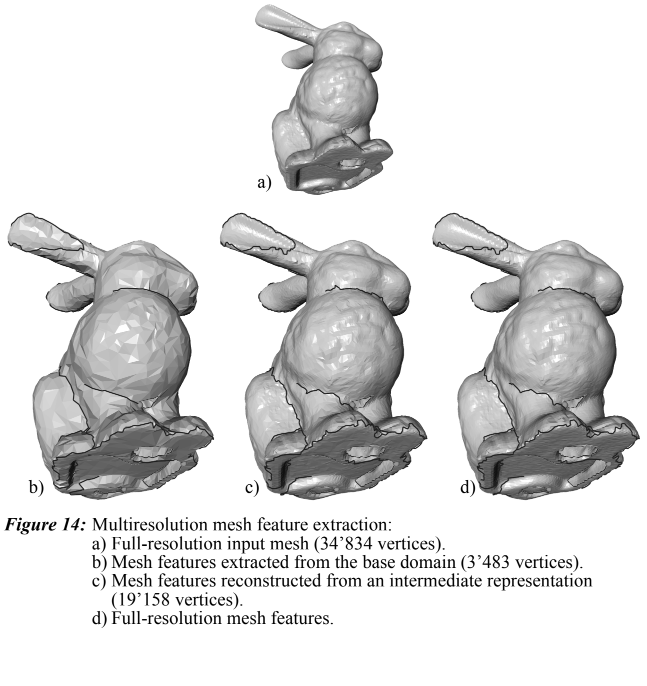

Figure 14: Multiresolution mesh feature extraction:
a) Full-resolution input mesh (34'834 vertices).
b) Mesh features extracted from the base domain (3'483 vertices).
c) Mesh features reconstructed from an intermediate representation (19'158 vertices).
d) Full-resolution mesh features.

Of course, there are special cases that must be analyzed, such as the split of vertices at the end of a mesh feature, or the split of vertices where two or more mesh features meet. These configurations can also be handled using straightforward variations of the masks discussed previously and will not be discussed further.

Consider figure 14 which illustrates our multiresolution approach. A coarse mesh is computed from the input surface (figure 14.a) and its mesh features are computed using the techniques discussed in section 3 and 4 (figure 14.b). The set of mesh features is then continuously updated during the refinement process. An intermediate state is shown in figure 14.c, where 50% of the vertices were already re-inserted. The end configuration is displayed in figure 14.d.

This approach could be further improved, since the coarse approximation of the input surface does not necessarily contain all the mesh features. Therefore, the algorithm should support the creation of new mesh features in the reconstruction process. This results in a multilevel feature detection strategy where the operators discussed in section 3 and section 4 are applied at different levels of resolutions.

6 Results and Applications

In this section we present some of the results generated by our framework. We use both well known meshes, such as the "Stanford bunny", the "mannequin" and the "golf club", as well as models from the domain of geoscience. Most of these meshes have irregular connectivity and possess a set of features readily identifiable. The exception is the geological surface displayed in figure 17 which does not contain any prominent feature.

In figure 15 we applied the framework to the mannequin head. To test the capabilities of the framework we first applied three Loop subdivision steps to the input data, so that the algorithm had to work on a very smooth domain. The results illustrated in figure 15 have been computed using the BFP operator and a large support. The most important components of the face, such as the eyes, nose, ears and mouth, have been properly recognized. The localization of the mesh features is good given the amount of information present in the mesh.

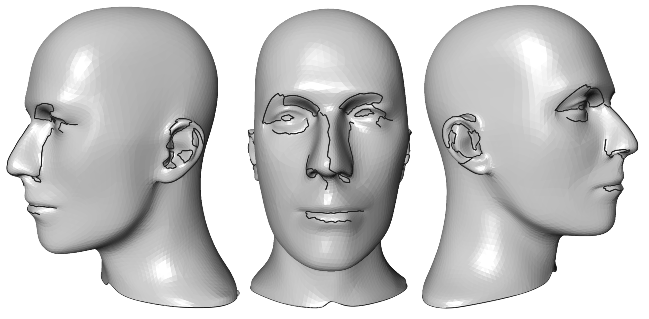
Figure 15: The features of the mannequin.

Next, we present the mesh features extracted from the Stanford bunny in figure 16. For this mesh we used the ABBFP operator and selected a support that filtered out most of the noise present in the fur of the bunny. The important components, such as the ears, tail, neck, legs and even some of the paws of the bunny, have been properly recognized. The hysteresis function removed other important features, such as the eyes and mouth. However, it should be noted that the magnitude of these features is almost the same as the magnitude of the noise present in the fur.

We believe that automatic feature extraction algorithms for meshes with arbitrary connectivity can be applied in different processes of geometric modeling. The first application we discuss is feature preserving fairing that has been proposed in our non-manifold fairing framework [11]. Using the techniques presented in this paper, high frequency noise can be removed from complex models without removing important features. As an example consider figure 17 depicting a geological model. The noise of the input surface can be eliminated either using standard fairing techniques (figure 17.b) or using feature-preserving fairing (figure 17.c).

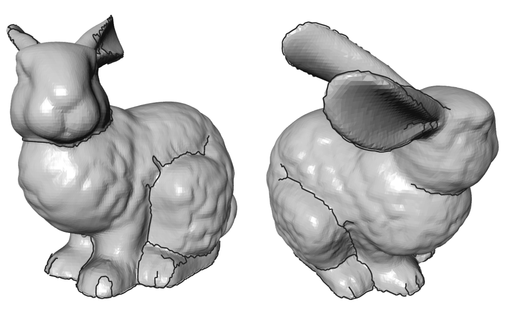
Figure 16: The features of the Stanford bunny.

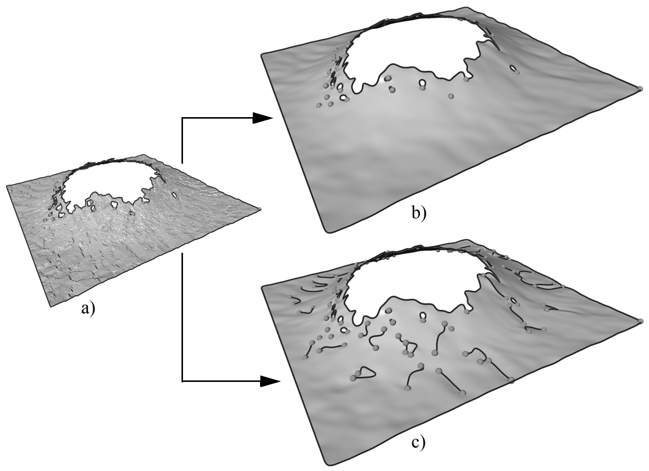

Figure 17: Feature-preserving fairing:
a) The input model.
b) Smooth model generated by standard fairing.
c) Smooth model generated by feature preserving fairing.

Mesh features can be used in conjunction with other operators as well in order to preserve information about the input model. Any subdivision operator can be extended to preserve features. This is accomplished by handling the mesh features as boundaries [1] and by applying the one-dimensional subdivision operators on them. An example is displayed in figure 18. The input mesh (figure 18.a) is first analyzed and its most important mesh features are extracted. The use of standard subdivision generates results of decent quality, but the magnitude of features is attenuated, as shown in figure 18.b. The use of a feature-preserving subdivision operator enables us to preserve the important mesh features much better, as illustrated in figure 18.c.

Finally, mesh features could also be used to govern a simplification algorithm: an edge collapse operator should not collapse edges perpendicular to a mesh feature, but rather parallel edges. Of course, error norms that control the simplification already strive to preserve features, but explicit feature preservation would guarantee that important information will not be removed from any of the approximations constructed by the algorithm.

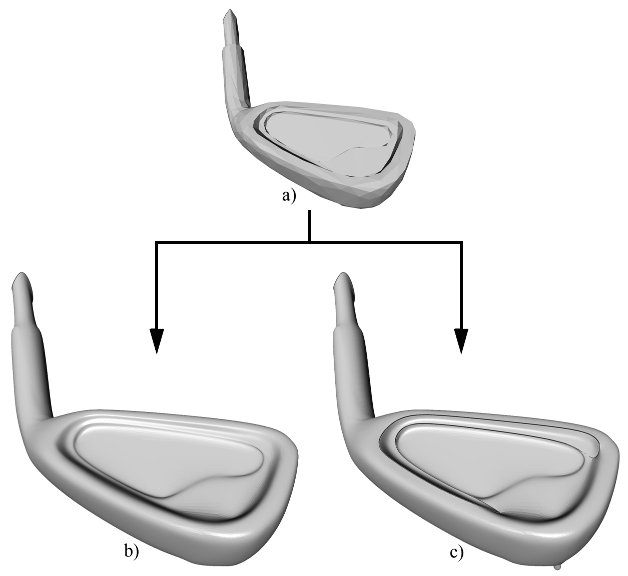

Figure 18: Feature-preserving subdivision:
a) The input mesh.
b) The set of mesh features generated by our framework.
c) Smooth surface generated by standard subdivision.
d) Smooth surface generated by feature preserving subdivision.

7 Conclusions and Future Work

In this paper we presented a framework for the detection of features in meshes with arbitrary connectivity. We proposed a two-stage process consisting of a *classification phase* and a *detection phase*. In order to handle a variety of different meshes, we provide a set of operators with different properties. We extended our framework to a multiresolution setting, which clearly improves the quality of the results and the performance of the algorithms.

The user must select the operators as well as a few parameters for the classification and detection steps. As such, the process is not fully automatic and requires manual assistance and tuning for optimal performance. We do not consider this as a major drawback, since all algorithms can be computed efficiently.

We believe that the results produced by our framework are of useful quality. We envision the following optimizations and extension on these techniques:

- *Improved classification operators*: one of the major difficulties encountered in the classification phase is to distinguish between high frequency noise and feature information, a well known problem in computer vision. We addressed this problem by constructing operators with adjustable support, which worked well in practice. However, we believe that further improvements could by obtained by mesh decomposition [7].
- *Improved skeletonizing operators*: the quality of the results generated by the thinning algorithm could be improved by using the weights computed in the classification phase more aggressively. Currently, only a topological measure is used. The advantage of a topology-driven thinning algorithm is improved robustness at the cost of an inferior localization of the mesh features.
- *Improved interaction*: the final goal of this project is, of course, to provide a sophisticated mesh analysis tool able to determine the optimal operators and most of the parameters autonomously, requiring only a few intuitive parameters from the users.

Acknowledgments

This research was made possible by grants from Schlumberger ATC, Austin, TX. The authors would like to thank Richard Hammerlsey and Karen Lu for many helpful discussions. Further thanks to Kuno Meyer and Philippe Zürcher who implemented parts of this framework.

Literature

[1] H. Biermann, A. Levin, and D. Zorin. “Piecewise smooth subdivision surfaces with normal control.” In K. Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 113–120. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[2] J. Canny. “A computational approach to edge detection.” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 8:679–698, 1986.

[3] U. Clarenz, U. Diewald, and M. Rumpf. “Anisotropic geometric diffusion in surface processing.” In *Proc. of the 11th Ann. IEEE Visualization Conference (Vis) 2000*, 2000.

[4] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. “Implicit fairing of irregular meshes using diffusion and curvature flow.” In *SIGGRAPH ’99 Proceedings*, Computer Graphics Proceedings, Annual Conference Series. ACM SIGGRAPH, ACM Press, Aug. 1999.

[5] M. Garland and P. S. Heckbert. “Surface simplification using quadric error metrics.” In T. Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 209–216. ACM SIGGRAPH, Addison Wesley, Aug. 1997. ISBN 0-89791-896-7.

[6] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, Reading, MA, USA, 1992.

[7] M. H. Gross and A. Hubeli. “Eigenmeshes.” Technical report, ETH Zurich, Mar. 2000.

[8] I. Guskov, W. Sweldens, and P. Schröder. “Multiresolution signal processing for meshes.” In *SIGGRAPH ’99 Proceedings*, Computer Graphics Proceedings, Annual Conference Series. ACM SIGGRAPH, ACM Press, Aug. 1999.

[9] R. Haralick and L. Shapiro. “Image segmentation techniques.” *Applications of Artificial Intelligence II*, April 9-11.

[10] H. Hoppe. “Progressive meshes.” In H. Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 99–108. ACM SIGGRAPH, Addison Wesley, Aug. 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[11] A. Hubeli and M. Gross. “Fairing of non-manifolds for visualization.” In *Proc. of the 11th Ann. IEEE Visualization Conference (Vis) 2000*, 2000.

[12] M. Kass, A. Witkin, and D. Terzopoulos. “Snakes: Active contour models.” In *Proc. of IEEE Conference on Computer Vision*, pages 259–268, London, England, 8-11 1987.

[13] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. “Interactive multi-resolution modeling on arbitrary meshes.” In M. F. Cohen, editor, *Computer graphics: proceedings: SIGGRAPH 98 Conference proceedings, July 19–24, 1998*, Computer Graphics -proceedings- 1998, pages 105–114, New York, NY 10036, USA and Reading, MA, USA, 1998. ACM Press and Addison Wesley.

[14] C. Loop. *Smooth Subdivision Surfaces Based on Triangles*. PhD thesis, Utah University, 1987.

[15] D. Park, K. Nam, and R. Park. “Multiresolution edge-detection techniques.” *Pattern Recognition*, 28(2):211–229, February 1995.

[16] P. Perona and J. Malik. “Scale-space and edge detection using anisotropic diffusion,” 1990.

[17] W. J. Schröder, J. A. Zarge, and W. E. Lorensen. “Decimation of triangle meshes.” In E. E. Catmull, editor, *Computer Graphics (SIGGRAPH ’92 Proceedings)*, volume 26, pages 65–70, July 1992.

[18] G. Taubin. “A signal processing approach to fair surface design.” In R. Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 351–358. ACM SIGGRAPH, Addison Wesley, Aug. 1995. held in Los Angeles, California, 06-11 August 1995.

[19] D. Zorin, P. Schröder, and W. Sweldens. “Interactive multiresolution mesh editing.” In T. Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 259–268. ACM SIGGRAPH, Addison Wesley, Aug. 1997. ISBN 0-89791-896-7.