Patch Layout from Feature Graph

Matthias Nieser[a], Konrad Polthier[a], Christian Schulz[a]

[a]*Freie Universität Berlin, Mathematical Geometry Processing Group*
Arnimallee 6, 14195 Berlin, Germany

Abstract

Structuring of surface meshes is a labor intensive task in reverse engineering. For example in CAD, scanned triangle meshes must be divided into characteristic/uniform patches to enable conversion into high-level spline surfaces. Typical industrial techniques, like rolling ball blends, are very labor intensive.

We provide a novel, robust and quick algorithm for the automatic generation of a patch layout based on a topology consistent feature graph. The graph separates the surface along feature lines into functional and geometric building blocks. Our algorithm then thickens the edges of the feature graph and forms new regions with low varying curvature. Further these new regions - so called fillets and node patches - will have highly smooth boundary curves making it an ideal preprocessor for a subsequent spline fitting algorithm.

Key words: reverse engineering, surface decomposition, curvature based segmentation

1. Introduction

Reverse engineering deals with the reconstruction of CAD surfaces, typically from scanned 3D geometries. Since current CAD system are based mainly on spline geometries, a scanned triangle mesh must be converted into a highly structured and segmented data structure. Our algorithm aims to automate the reconstruction process. It is a two step process. In a first step we generate the topology of the final patch layout. This topology is encoded in a feature graph, i.e. there exists a one to one relation between feature graph elements such as nodes, edges and regions to the patches of the final layout. Furthermore the feature graph is an intersection free graph embedded on the surface whereas its smooth edges are oriented along geometric surface features. In a second step, out of the feature graph a geometrically reasonable patch layout is generated. The resulting patches have a uniform curvature distribution and are encircled by smooth boundaries. Such an automatic algorithm avoids many labor intensive manual segmentation approaches.

1.1. Previous work

Our patch layout algorithm is related to many previous techniques in surface segmentation and graph smoothing algorithms.

A general overview about surface decomposition methods is given in [23]. It starts with its roots in image processing, where surfaces are treated as height fields, i.e. there exists a canonical parametrization of the surface over a planar domain as used in [22]. The main part of [23] contains a detailed overview about segmentation algorithms working on general triangulated surfaces, showing their variety of applications and implementations. Due to different aims possible objectives range from remeshing, animation [1], shape matching [8], mesh editing to geometry compression and other areas. Here, we focus on segmentation of CAD parts for reverse engineering.

Some related work focuses on surface segmentation by approximation with several kinds of predefined types of primitives. In [3], planes are being fitted, [26] uses a collection of CAD primitives, such as spheres or rolling ball blends.

The use of parametrized shapes is suggested in [12]. This idea is further explored in [14], where the notion of morphological properties of shape templates is introduced. In this work the author proposes a two step generic algorithm to identify a surface part with an instance of a shape template. It starts to assign a shape instance to a surface region by varying its morphological properties, followed by an embedding of the matched template into the surface.

Another approach for partitioning is demonstrated in [18] [9]. There vertices of a triangulated surface are clustered into groups belonging to a specific shape type using multiresolution.

Julius et. al and Shatz et. al [13, 24] shows a tiling of a given model into nearly-developable charts. This kind of chart tiling makes it possible to recreate the given surface as paper craft model.

Levy et al. [16] use a region growing algorithm for creating patches whose boundaries run along sharp features. In a first step, some surface features are being detected. Then a set of regions is constructed, which meet at these features.

There are many approaches on computing a feature layout using Morse theory. In [5], an eigenvector of the Laplacian is computed and used as Morse function. The Morse complex which is then built from this function segments the surface into quads. In [7, 2], the construction of a Morse-Smale complex is described. With prescribing an adequate Morse function which represents the important parts of the surface, one can control the alignment of the feature layout. In [6], a curvature based Morse function is used to construct a Morse-Smale complex which aligns to surface features. This approach is applied to CAD models in [25].

We also need to smooth patch boundary curves. In [15] the use of snakes for the generation of smooth curves on triangulated surfaces is proposed. This approach requires the repeated projection of the actual curve onto a two-dimensional domain. The curve smoothness is controlled via an energy term. Recasting the problem of smooth curves on triangulated manifolds to a high dimensional optimization problem is described in [11]. Furthermore, the alignment of curves along features can be driven by the use of the feature sensitive metric introduced in [20]. Thickening of smooth curves is mentioned in [25] but without going into the actual details of the thickening procedure.

1.2. Contributions

The underlying structure of a given CAD surface is determined by its main building blocks, i.e. a set of characteristic CAD surface types. The methods mentioned above aim for such a decomposition into meaningful patches. The boundaries of these patches form an embedded graph on the surface. Assuming CAD models with round geometric feature edges, i.e. providing no clear defined boundary between primitives, the faces encircled by the graph's edges are not uniformly curved. Thus, they are not suited for low order spline fitting. Regarding CAD surfaces having round geometric features our contribution can be summarized as follows:

- an algorithm to generate a net of curves, running along geometric surface features, such as valleys or ridges, called the feature graph
- an energy formulation to align and smooth a curve within a feature region
- a method to decompose a surface into its functional parts based on a given feature graph using an edge thickening - offsetting - procedure, the single parts are encircled by smooth boundaries aligned to nearby surface features

We will show how to generate a consistent feature graph. Based on this graph a patch layout is computed in reliable and fast way. For the creation of both structures no primitive fitting, i.e. template matching, is required. Starting with a triangle mesh as shown in fig.1, left, we will end up with a decomposition like the one in fig.1, right.

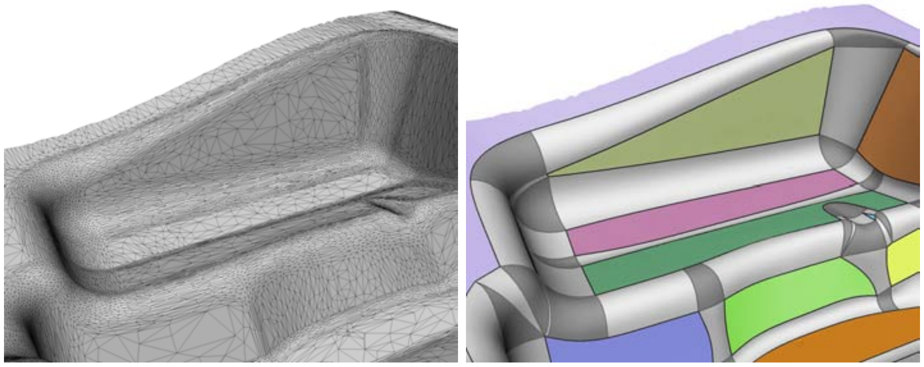

Figure 1: Left: Typical CAD part as triangle mesh. Right: Patch layout of the CAD part

Within our setting the feature graph resembles the embedded graph resulting from other methods mentioned above. Thus the feature graph needed for our patch layout generation could be replaced by any other graph structure describing a surface partitioning.

1.3. Organization of the paper

In section 2, we explain our basic concepts and underlying notions of a feature graph and a patch layout. Section 3 deals with the generation of a feature graph. The creation of the patch layout from a given feature graph is explained in section 4. All the steps described in sections 3 and 4 are illustrated on the geometry shown in fig.3, left. Finally, results of our tests are given in section 5.

2. Setting

The basic idea of our algorithm is to detect an initial set of primitives and create a corresponding patch layout. The initial primitive guess is just a rough approximation of the final layout (fig.2, left, fig.3, left), i.e. it is lacking smooth boundaries and inherent connectivity information. Thus, in an intermediate step we construct a feature graph (fig.2, left) which contains the missing connectivity information. From this *feature graph* we then derive the significant points and curves, meeting our smoothness and alignment requirements, of the final *patch layout* (fig.2, right).

For a complete description of our layout generation method we define:

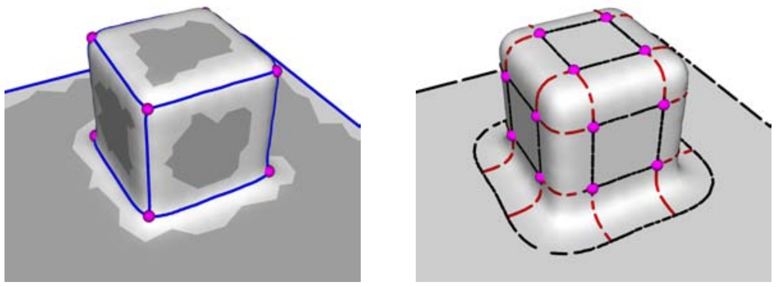

Figure 2: Left: Feature graph on CAD part consisting of faces, feature edges and node points. The dark grey parts within each face denote the plane-like or weakly curves parts whereas in the light grey part the surface starts to get curved. Right: Patch layout with face patches, fillets and node patches, as well as offset and node curves and offset nodes.

Feature graph. A graph on the surface (fig.2, left), which represents the underlying structure of a CAD surface. The feature graph is a net of smooth surface curves, which run along surface features. It consists of:

Faces	Main parts of the surface. These can be any kind of primitives (planes, cylinders, cones, ...) from which the surface is made. In this approach, we mainly focus on plane-like faces.
Feature edges	Smooth edges which separate two adjacent faces and correspond to cylindrical/conical regions.
Node points	Isolated points, which correspond to spherical/hyperbolic regions, where several feature edges meet, i.e these points are also incident to more than one feature graph face.

Patch layout. An embedded graph on the surface (fig.2, right), which decomposes the geometry into various cells. Within each cell of the patch layout, heavily changing curvature is not allowed. The possible cell types can be categorized as follows:

Face patches	Represent the planar or weakly curved part of feature graph faces.
Fillets	Connectors between adjacent face patches. They correspond to edges in the feature graph. Typically, a fillet is a cylindrical or conical part with high curvature in direction of the connecting faces.
Node patches	Connectors of several fillets. They can have a spherical or hyperbolic shape of any kind.

Regarding the boundaries between these cells we will encounter two types:

Offset curves	Encircle face patches. Each offset curve separates a face patch from an adjacent fillet. The feature edge which represents this fillet runs more or less parallel to the offset curve. An offset curve can be seen as a shifted version of a feature edge, i.e. it is obtained by translating a feature edge by a variable distance value.
Node curves	Separate fillets from node areas. In general each node patch is bounded by a sequence of smooth node curves.

Start and endpoints of these curves will be denoted as *offset nodes*. Further we will refer to the triangle mesh by M and its triangles by T.

3. Feature graph

The basis of a consistent layout is a feature graph representing the layout's topological structure. So given a triangulated mesh M we present a strategy to build all parts of a feature graph, such as faces, feature edges and nodes.

We focus on detecting plane-like regions. Other types of primitives could possibly be included as an extension of the algorithm. The basic idea is to detect those primitives I_i on M and expand them to cover all of M. So we will have a one-to-one relation between initial regions I_i and expanded regions

F_i. Having covered all of M we detect node points, i.e. vertices where more than two regions meet. The node points are connected by curves, which separate adjacent regions. These curves are very jagged and run only along edges of the underlying mesh. We take these curves as a first approximation of the later feature edges. Thus, they need to be smoothed to meet our alignment and orientation requirements. The result of this last step is a net of smooth curves on the surface - the feature graph. These curves encircle the feature graph faces containing the regions I_i, the weakly curved or plane-like part of each face. The algorithm to generate the feature graph can be outlined as proposed in alg.1.

```
Algorithm 1: Generate feature graph
  Input: triangle mesh $M$
  Output: feature graph
1 Compute principle curvatures (values and directions)
2 Detect initial regions $I_i$
3 Expand regions $I_i$ by a region growing process
4 Extract nodes and edge based face boundaries
5 Create smooth feature edges from edge based face boundaries
```

The details for every step of our feature graph algorithm (alg.1) are explained in detail in the following subsections.

3.1. Principal curvatures.

The algorithm starts with computing the principal curvatures of the surface. Curvature information is computed for each vertex of the mesh. We use an approximation of the shape operator given in [10]. i.e. a stable and reliable method where no fitting needs to be performed. Other methods (e.g. [4, 21]) would also be practicable. Having curvature values at all vertices we then assign curvature information to all triangles $T \in M$ by averaging curvature information of all three incident vertices. Thus, for each triangle four unit vectors pointing in principle curvature directions ($\pm X_{max}$, $\pm X_{min}$) are given together with their corresponding curvature values (κ_{max} and κ_{min}) with $|\kappa_{max}| \geq |\kappa_{min}|$.

3.2. Detect initial faces

Initial faces I_i are taken to be the seeds for the set of feature graph faces (fig.2, left, fig.3, left). They should be thought of as the inner of the main building blocks from which the CAD surface is made of. In general, one could detect any kinds of primitives as cylinders, cones, spheres, etc. For our purpose, it is sufficient to restrict the primitives to nearly flat parts.

We use a curvature threshold τ to characterize all triangles as being part of an initial face. Thus we define the following set of flat triangles $I := \{\text{triangle } T \,|\, |k_{max}| < \tau\}$. In general, I can be split into a set of simply connected components I_i, i.e. $I = \{I_0, \ldots, I_n\}$.

Apart from using τ to characterize initial regions, it can also be considered as a measure of allowed noise within initial regions. A higher value of τ will ignore more noise.

3.3. Expand initial regions, Detection of edges and nodes

The expansion of the initial regions I_i gives us a rough approximation of the feature graph. This process sets up the final topology of the feature graph and therefore defines its faces, feature edges and node points. The detected node points are held fix during the rest of the algorithm, whereas the alignment of the feature edges gets adapted later in a smoothing step.

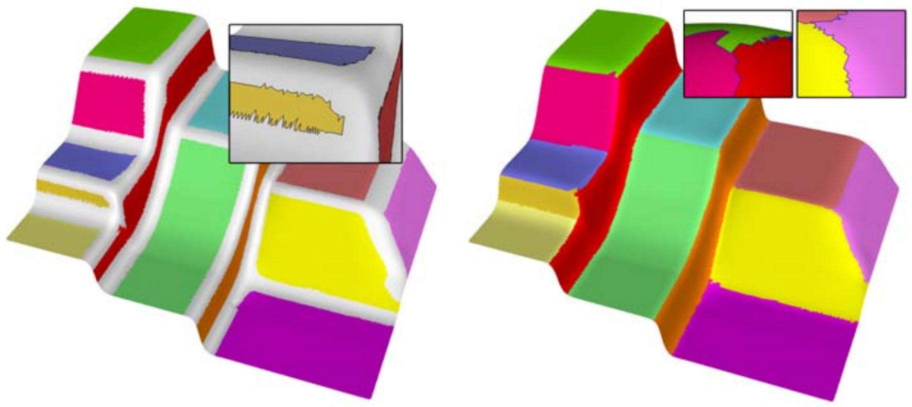
Figure 3: Left: Initial regions I_i. Right: Complete covering of M by expanded regions F_i and parts of the unsmoothed feature graph.

In order to ensure the correct placement of feature graph nodes and edges, we developed a special growing strategy based on curvature information. We use a region growing approach to realize the expansion of initial regions I_i into the uncovered part of M. The growing is controlled by a feature function, which assigns curvature related priority values to free triangles.

Feature function. We expect that feature graph edges mainly run along a ridge or in our case, where $|\kappa_{max}|$ is high. Using this scalar value as feature function for the expansion process works very well in cylindrical areas. However, when the surface becomes spherical/hyperbolical, the values of κ_{min} and κ_{max} get more and more similar. Due to noise and numerical imprecision, the principal curvature directions are very unstable. As a result, the edges of the feature layout will run more or less randomly through such an area. For the same reason we will also loose control about the location of the node points, i.e. where three or more regions meet, in spherical/hyperbolic surface parts.

Our experiments show, that node points are best placed where the Gaussian curvature K_G is high. The value of K_G can be computed stable, even in spherical regions. We therefore propose a growing strategy which uses $|\kappa_{max}|$ values within cylindrical, i. e. fillet-like regions (with stable curvature directions), whereas in spherical regions K_G values should drive the expansion. We use $|\kappa_{max} - \kappa_{min}|$ as an estimate for the stability of the curvature operator.

Region growing. We classify the free triangles, i. e. not assigned to one of the initial regions I_i, into two groups namely stable E and unstable ones N:

$$N := \{\text{triangle } T \,|\, T \notin I, |\kappa_{max} - \kappa_{min}| < t\}, \quad t \in \mathbb{R}$$
$$E := M \setminus (I \cup N)$$

Thus, our region growing will be driven by κ_{max} for $T \in E$ and K_G for $T \in N$. First the initial regions are expanded into the set E giving us a rough approximation of the feature edge within fillet like regions. During this stage the growing is controlled by k_{max}. In the second step the rest of the surface gets covered, i. e. regions are expanded into areas with spherical character, i. e. into N, using $|K_G|$ as the growing function. Finally the node points of the feature graph are found. Points of the triangle mesh where more than two regions meet get identified as nodes of the feature graph (fig.3, right). An algorithm for our two step growing strategy is given in alg.2, also containing the details of our actual region growing procedure.

Algorithm 2: Expand Regions

```
   Input: Set of initial regions $I = \{I_1, \ldots, I_n\}$ with $I_i \subset M$
   Output: Set of disjoint regions $F = \{F_1, \ldots, F_n\}$ with $\cup_{i=1}^{n} F_i = M$
1  Initialize set of patches $F = \{I_1, \ldots, I_n\}$
2  regionGrowing($F$, $k_{max}$, $N$)
3  regionGrowing($F$, $k_G$, $E$)
```

Procedure "regionGrowing(F, f, C)"

```
   Input: Set of regions $F = \{F_1, \ldots, F_n\}$, feature function $f$, one region
          into which growing is allowed $C$
   Output: Expanded regions $F_i$ with $C \subset \cup_{i=1}^{n} F_i$
1  PriorityQueue queue;
2  foreach triangle $T \in F_i$ do
3      queue.enqueue(($T, i$) with key=$f(T)$);
4  end
5  while queue not empty do
6      $(T, i)$ = queue.extractMin();
7      foreach neighbours $T'$ of $T$ do
8          if $T'$ has not been marked as patch member and $T' \in C$ then
9              Mark $T'$ as member of patch $F_i$;
10             queue.enqueue(($(T', i)$, key=$f(T')$));
11         end
12     end
13 end
```

The method is similar to a watershed technique from image segmentation, see e.g. [17], where the order when to add triangles to an initial region I_i is also done via a priority queue.

3.4. Smooth feature graph

The rough approximation to the final feature graph from the last step corresponds to the set of boundaries of the expanded initial regions (fig.3, right, fig.4, left) - a set of polygonal curves, where each curve runs along edges of the underlying triangulation. Because a feature graph with smooth edges is necessary to generate a consistent patch layout we need to smooth these curves (fig.4, right). During smoothing the node points, which were detected in the previous region growing step, are held fix.

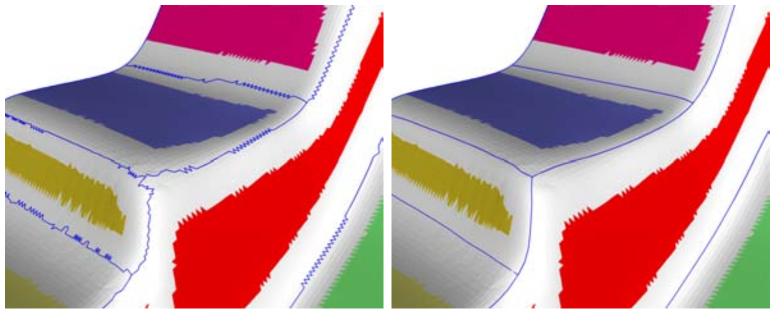

Figure 4: Smooting feature curves with end points held fixed. Left: Edge based region boundary. Right: Smoothed region boundary.

Smoothing energy. If the feature edges get smoothed using a standard method, e.g. Laplace smoothing, they cannot be guaranteed to stay in highly curved feature areas of the surface. Instead, we introduce an algorithm which alters a curve on a surface such that the curve gets aligned to a given vector field. This approach can be applied to the field of minimal principle curvature directions X_{min}. In practice this works fine, since in highly curved areas, the principal curvature directions are very stable and smooth. So we are looking for a smooth curve connecting the node points, which is aligned to the X_{min} field in the vicinity of the curve.

In general our alignment energy for a curve γ can be defined with respect to a given tangential vector field X as follows:

$$E(\gamma) := \int_{\gamma} \left(\frac{\langle \dot{\gamma}, JX \rangle}{\|\dot{\gamma}\| \, \|X\|} \right)^2 ds \tag{1}$$

where J denotes the rotation by 90 degrees in the oriented tangent planes. It can be seen as a measure of how much curve tangents and vector field deviate. The energy vanishes if the curve is an integral curve of X. The alignment energy (eq.1) is non-linear and non-quadratic in the vertex positions, which makes it more difficult to find a minimum. But since we are optimizing a 1D curve, the number of degrees of freedom stays relatively small, so even a standard Euler method finds a local minimum in a reasonable time.

We discretized a feature curve by a curve, whose vertices lie on the surface. The edges are not forced to stay on the surface. Assuming, that the vector field

X is locally nearly parallel (its covariant derivative vanishes), the variation of the energy (eq.1) at vertex $v \in \gamma$ is approximated by

$$\delta_v E \approx 2 \int\limits_{\gamma} \langle \delta_v \frac{\dot{\gamma}}{\|\dot{\gamma}\|}, \frac{JX}{\|X\|} \rangle \, ds \approx 2 \sum_{w \in \{v-1, v+1\}} \frac{\langle e_w, X_w \rangle \, (X_w)}{\|e_w\| \, \|X_w\|^2} \tag{2}$$

In (eq.2) $X_w = (X(\gamma(w)) + X(\gamma(v)))/2$ is the mean vector of X at the edge (v, w) and $e_w = \gamma(w) - \gamma(v)$ represents an approximation to curve's tangent. Applying several steps of an explicit Euler method leads to smooth feature curves (fig.4, right).

In regions near to the two ends of the edges, the direction of minimal and maximal principle curvature may exchange due to the effect of an adjacent fillet. This situation can be detected by matching the principal curvature vectors between all adjacent triangles of the mesh. If the minimal and maximal curvature vector exchange at an edge, where the feature polygon runs through, the vectors in X are switched to be those of the maximal curvature field instead of the minimal one.

4. Patch layout

Given a consistent topological feature graph we are now able to create the final patch layout, i.e. the structure which decomposes the surface into its functional parts such as faces, fillets and node areas. Our patch layout generation process can be visualized as thickening the feature graph edges back into its faces plus cutting of areas around its nodes. The complete algorithm for the computation of patch layout related curves and nodes is given in alg.4.

Algorithm 4: Patch Layout

```
   Input: Feature graph
   Output: Patch layout
1  foreach Node point of the feature graph do
2      Compute all its offset nodes
3  end
4  foreach Face $F_i$ of the feature graph do
5      foreach Feature graph edge $\gamma_j$ bounding $F_i$ do
6          Determine offset direction $dir = getSide(\gamma_j, F_j) \in \{left, right\}$
7          Compute distance map $d_j^{dir}$
8          Smooth distance map
9          Compute offset curves $\delta_j^{dir}$
10     end
11 end
12 Compute node curves
```

The proposed thickening procedures ensures the alignment of face boundaries to nearby feature lines. Furthermore we connect feature oriented boundaries in the vicinity of nodes areas at offset nodes. After having computed a consistent loop of smooth offset curves around each face, we cut out the node areas by node curves. An illustration of the whole process is given in (fig.5).

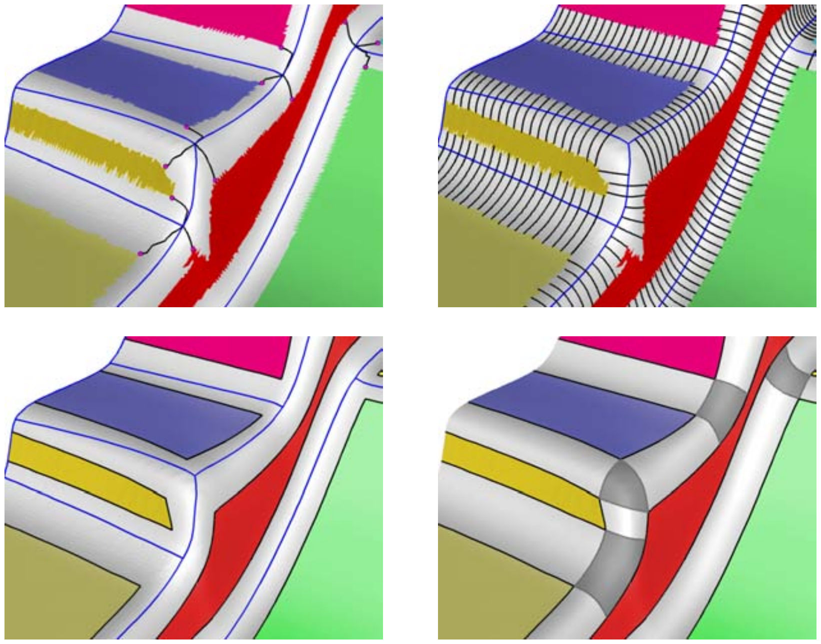

Figure 5: Generation of offset curves. Top left: Compute offset nodes as nearest points to feature nodes in each adjacent initial patch I_i. Top right: Regard distance to initial patch I_i as a one dimensional graph over the feature line. Bottom left: Smooth and aligned offset lines. Bottom right: Final offset layout after smoothing the distance function.

4.1. Offset nodes

Offset nodes are points on the surface where offset curves and node curves meet. By definition a region is bounded by a set of feature edges, which start and end in node points. So for each face of the feature graph there exists an associated set of node points. For each associated node point an offset node gets created (fig.5, top left). A canonical choice for an offset node within a certain face F_i is the point, which is contained in the corresponding flat or weakly curved part I_i and which closest to the node point of the feature graph. Here we use Dijkstra distances to determine those points. We refer to an offset node within a face by n_{ij}, where the two indices denote the offset curves, which meet there (fig.6, left).

In practice, it happens that offset nodes of two adjacent feature nodes fall onto the same geometrical position on the surface, i.e. closest points of different node point coincide. This occurs especially for nearby feature nodes which are connected by a very short feature edge. In this case, the corresponding feature edge will not be offsetted. So that the final offset layout will not be spoiled by these nearby feature nodes (fig.9, bottom right).

4.2. Offset curves

Each feature edge γ_i gets offsetted into its two adjacent faces resulting in two offset curves δ_i^{left} and δ_i^{right}. The upper index refers to the offset direction as seen from γ_i, whereas the lower index indicates the feature curve this offset curve belongs to. Each of these curves is created from a scalar function denoted by $d_i^{left}(t)$ and $d_i^{right}(t)$ defined along γ_i. Here the indices of $d_i^j(t)$ are defined in the same manner as for the offset curves. In the remaining section we skip the indices on, i.e. $d_i^j(t)$ becomes $d(t)$, to shorten the notation. $d(t)$ measures the distance between a feature edge and the flat part of the corresponding adjacent face. This is in general a nonsmooth function, so we apply convolution to get rid of spikes within the set of distance values. The resulting distance values then encode points on the final offset curve.

Distance function: The parameterized feature edge $\gamma(t)$ is represented by a set of points uniformly distributed along the feature edge. From each of these points, a geodesic ray is shot perpendicular to the curve until the corresponding initial region is hit. How to extend a ray geodesically can be found in [19]. The length of the ray defines the distance $d(t)$ (fig.5, top right). If the ray does not hit the corresponding initial region between the two offset nodes, the value of $d(t)$ is set to be undefined. The result is a distance function $d : [a, b] \to \mathbb{R}$ for which all values in $[a, b]$ are well defined and the rays emanating from $\gamma(a)$ resp. $\gamma(b)$ run into the corresponding offset nodes.

Convolution: We smooth $d(t)$ by convolution with a hat function with large support (e.g. half of the length of the feature curve). Let $d : [a, b] \to \mathbb{R}$ be the distance map after parameterizing the supporting interval on the feature curve by arc length (as described above). By construction, the function values $d(a)$ and $d(b)$ at the endpoints are the geodesic distance of the feature curve to the offset nodes (fig.6, right).

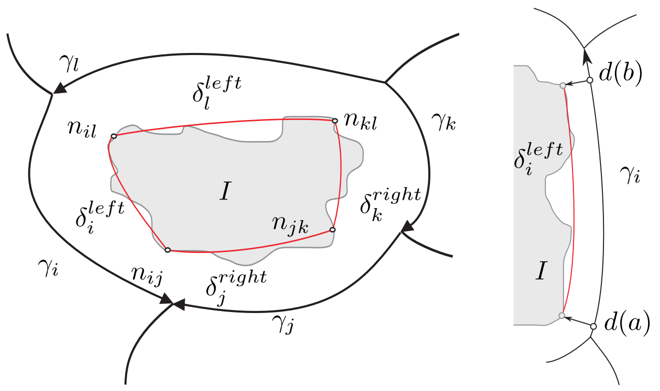

Figure 6: Left: Initial patch I of one face with adjacent feature curves γ_i and corresponding offset curves δ_i^{dir} and offset nodes n_{ij}. Right: Offsetting a feature curve γ_i, definition of the distance function $d_i^{dir}(t)$ only on a subset of γ_i

When keeping these values fix during the smoothing process, the resulting offset curves will start and end in offset nodes. Thus, we have to convolute d with a hat function and keep the function values at the endpoints. The trick for doing this is to extend d to a larger domain in $\mathbb{R}$ by mirroring at the endpoints, i.e.

$$\begin{aligned} d(a-x) &:= 2d(a) - d(a+x), \\ d(b+x) &:= 2d(b) - d(b-x), \quad x \in [0, b-a]. \end{aligned}$$

Convolution of this function with a hat function will (by symmetry) not change the function values at a and b. These convoluted distances define a sequence of points along the feature curve. Out of this sequence our polygonal offset curve lying on the surface is created (fig.5, bottom left).

4.3. Node curves

There is one node area for each node point of the feature graph. In most cases, a node area gets encircled by a sequence of node lines, which start and end in offset nodes. As illustrated in (fig.7) let v be a feature graph node and γ_i a feature line emanating from v. In general, there are two offset curves δ_i^{left}, δ_i^{right}, which arise by offsetting γ_i into the two adjacent faces. So a node curve needs to be created between the offset nodes n_{ij}, n_{ik} to separate the node area from the fillet corresponding to γ.

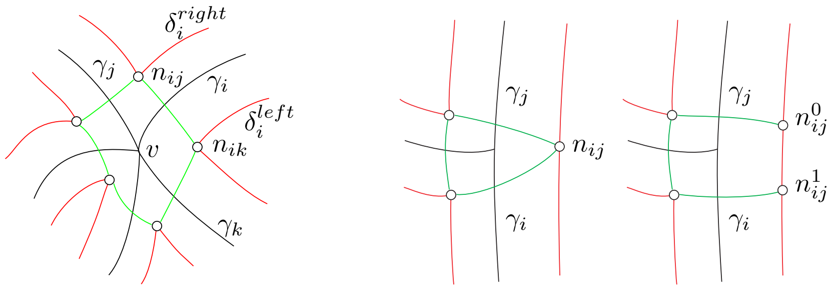

Figure 7: Left: Node area. The node lines (green) connect the endpoints of offset curves. Right: Two feature edges meet at node point with angle close to 180, the corresponding offset node gets split into two new ones.

This is done by constructing a plane out of the two points n_{ij}, n_{ik} and their normals. The plane is defined to contain the vector connecting n_{ij} and n_{ik} and the average of the two normals. The intersection curve of this plane and the mesh will then be our actual node curve. If two feature curves γ_i and γ_j meet at a node point with an angle close to 180 degrees (fig.7, right), the corresponding offset node n_{ij} gets split into two new ones. The two new nodes n_{ij}^0 and n_{ij}^1 are found using the distance map. We look for the first point within the valid range of $d(t)$. The actual node curve is constructed as in the usual case. In the actual implementation, we used a threshold of 120 degrees.

5. Results

We tested the algorithm on several CAD parts provided by our industry partner Tebis AG. Here we discuss two parts in detail: the first part belongs to a scan of a BMW motorcycle (fig.8). As can be seen, the algorithm finds a suitable decomposition of the complex surface. The right lower picture shows the patch layout on the part. The surface contains approximately 100k triangles and the whole patch layout generation process took about 1 minute. The main part was the curve smoothing, which took about 40 seconds. The second part is a deformed metal plate (fig.9), where the algorithm took also less than 2 minutes. As can be seen this geometry contains fillets with varying thickness (fig.9, bottom left) which are well detected by our method. The final patch layout also contains nearby node points showing the ability of our method to work in cases of degenerated offset curves (fig.9, bottom right).

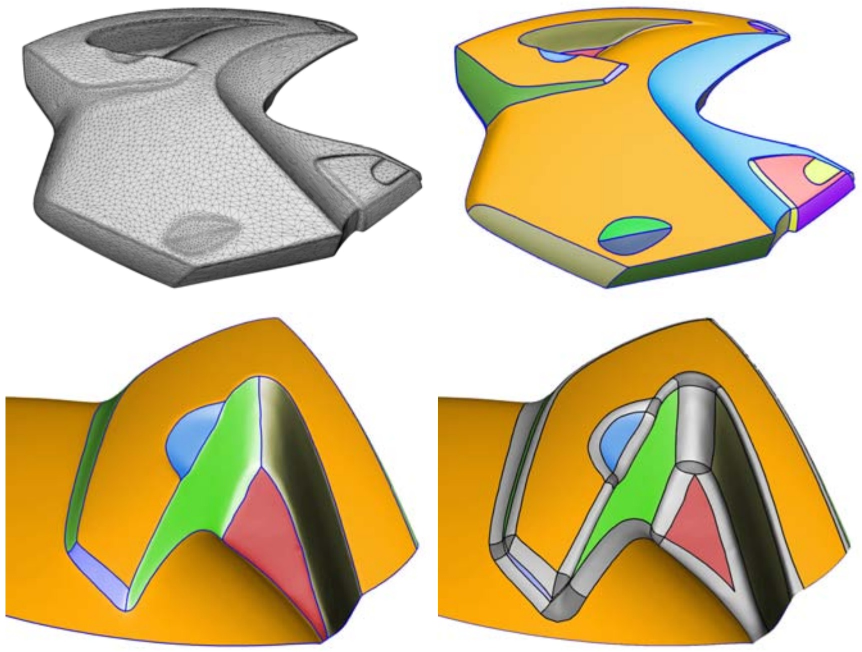
Figure 8: Feature layout on the BMW motorcycle part. Top: given triangulated model. 2nd, 3rd: Feature graph from our method. Bottom: Final patch layout.

Our method is made to work with geometries having round features, because at sharp edges a fillet region would be not well defined. Having noisy data the geometry could be smoothed using [10] or the noise can captured by our threshold defining the flat parts.

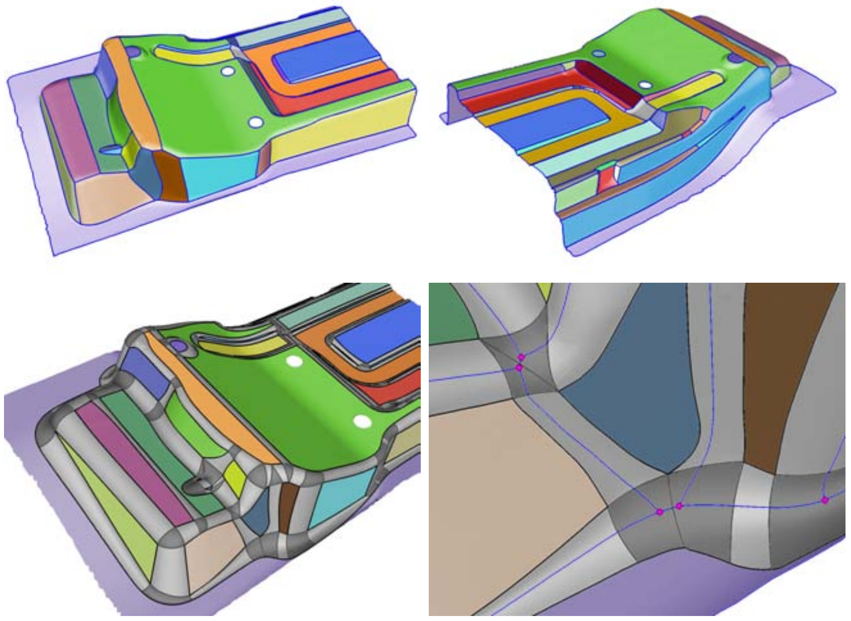

Figure 9: Top left and right: Feature graph on CAD model. Bottom left: Patch layout. Bottom right: offset curves and feature graph in regions with nearby node points

6. Conclusion

We presented a method which fully automatically decomposes a given surface into patches which are suited for spline fitting. Just a few parameters, e.g. the curvature threshold, are necessary to drive the generation process, i.e. to control the look of the final layout. Furthermore having an existing reverse engineering pipeline based on graph like structure our method could easily be plugged in to create a patch layout.

The method shows good results on a given class of *CAD modeled* geometries. We only allow geometries which made of more or less planar patches with smooth connector parts (fillets and nodes).

There are many other algorithms for patch layout generation, but the most of them aim at a slight different problem. They focus on generation of a feature graph. This is less useful for spline fitting, since the connection between different patches are often rounded. The generation of a patch layout using offset curves has not occurred to us yet. However, the feature graph from our method is quite similar to that from [25]. One could of course use any other method for feature graph generation and use it directly as input to our second step for computing the offset layout.

To be able to extend the algorithm to surfaces made of more complicated building blocks, additional study is necessary. The basic idea is to detect the initial regions I_i as given surface primitives (cylinders, spheres, cones, ...) which are found as a subset of the surface. Therefore it would be necessary to explore the curvature related properties of common CAD building blocks.

Another extension would be to include sharp edges into the layout. Sharp edges will not be offsetted, because the adjacent patches are directly connected without any smooth connection.

7. Acknowledgement

We thank DFG research center Matheon and Tebis AG for supporting this research and providing the CAD models.

References

[1] Miklós Bergou, Saurabh Mathur, Max Wardetzky, and Eitan Grinspun. Tracks: toward directable thin shells. *ACM Trans. Graph.*, 26(3):50, 2007.

[2] F. Cazals, F. Chazal, and T. Lewiner. Molecular shape analysis based upon the morse-smale complex and the connolly function. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry*, pages 351–360, New York, NY, USA, 2003. ACM.

[3] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 905–914, New York, NY, USA, 2004. ACM Press.

[4] David Cohen-Steiner and Jean-Marie Morvan. Restricted delaunay triangulations and normal cycle. In *Proc. of Symp. on Comp. Geom.*, pages 312–321. ACM Press, 2003.

[5] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart. Spectral surface quadrangulation. *ACM Trans. Graph.*, 25(3):1057–1066, 2006.

[6] Herbert Edelsbrunner. Surface tiling with differential topology. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*, page 9, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[7] Herbert Edelsbrunner, John Harer, and Afra Zomorodian. Hierarchical morse complexes for piecewise linear 2-manifolds. In *SCG '01: Proceedings of the seventeenth annual symposium on Computational geometry*, pages 70–79, New York, NY, USA, 2001. ACM Press.

[8] Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. Modeling by example. *ACM Trans. Graph.*, 23(3):652–663, 2004.

[9] Michael Garland, Andrew Willmott, and Paul S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 49–58, New York, NY, USA, 2001. ACM.

[10] Klaus Hildebrandt and Konrad Polthier. Anisotropic filtering of non-linear surface features. *Computer Graphics Forum*, 23(3):391–400, 2004.

[11] Michael Hofer and Helmut Pottmann. Energy-minimizing splines in manifolds. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 284–293, New York, NY, USA, 2004. ACM Press.

[12] Imre Horvth Joris S. M. Vergeest, Sander Spanjaard and Jos J. O. Jelier. Fitting freeform shape patterns to scanned 3d objects. *J. Comput. Inf. Sci. Eng.*, 1(3):218–225, September 2001.

[13] Dan Julius, Vladislav Kraevoy, and Alla Sheffer. D-charts: Quasi-developable mesh segmentation. In *Computer Graphics Forum, Proceedings of Eurographics 2005*, volume 24, pages 581–590, Dublin, Ireland, 2005. Eurographics, Blackwell.

[14] Thomas Robin Langerak. *Freeform feature recognition and manipulation to support shape design.* PhD thesis, TU Delft, 2008.

[15] Yunjin Lee and Seungyong Lee. Geometric snakes for triangular meshes. *Computer Graphics Forum*, 21(3):229–238, 2002.

[16] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 362–371, New York, NY, USA, 2002. ACM.

[17] Alan P. Mangan and Ross T. Whitaker. Partitioning 3d surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, 1999.

[18] Bianca Falcidieno Marco Attene and Michela Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22(3):181–293, March 2006.

[19] Konrad Polthier and Markus Schmies. Straightest geodesics on polyhedral surfaces. In Hans-Christian Hege and Konrad Polthier, editors, *Mathematical Visualization*, pages 135–150. Springer Verlag, Heidelberg, 1998.

[20] Helmut Pottmann, Tibor Steiner, Michael Hofer, Christoph Haider, and Allan Hanbury. *Computer Vision - ECCV 2004*, chapter The Isophotic Metric and Its Application to Feature Sensitive Morphology on Surfaces, pages 560–572. Springer, 2004.

[21] Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, Yu-Kun Lai, and Shi-Min Hu. Principal curvatures from the integral invariant viewpoint. *Comput. Aided Geom. Design*, 24:428–442, 2007.

[22] Nickolas S. Sapidis and Paul J. Besl. Direct construction of polynomial surfaces from dense range images through region growing. *ACM Trans. Graph.*, 14(2):171–200, 1995.

[23] Ariel Shamir. Segmentation and shape extraction of 3d boundary meshes. In *EUROGRAPHICS '06: STARS*, pages 137–149, 2006.

[24] Idan Shatz, Ayellet Tal, and George Leifman. Paper craft models from meshes. *Vis. Comput.*, 22(9):825–834, 2006.

[25] Tamás Várady, Michael A. Facello, and Zsolt Terék. Automatic extraction of surface structures in digital shape reconstruction. *Comput. Aided Des.*, 39(5):379–388, 2007.

[26] Jianhua Wu and Leif Kobbelt. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum*, 24:277–284(8), September 2005.