Patch Layout from Feature Graph

Matthias Nieser, Konrad Polthier, Christian Schulz

February 24, 2009

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 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.

1 Introduction

Reverese 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 automize 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 surface features. In a second step, out of the feature graph a geometrically reasonable patch layout is generated. The resulting patches 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 has contact to many previous techniques in surface segmentation and graph smoothing algorithms.

A general overview about surface segmentation methods is given in [Sha06]. It contains an outline from its roots in image processing, where surfaces are treated as height fields up to segmentation algorithms working on triangulated surfaces.

The variety of segmentation algorithms is very huge due to different aims. Objectives range from remeshing, simplification, shape matching, mesh editing to geometry compression and other areas. Here, we focus on segmentation of CAD parts into reasonable surface patches.

Some related work focuses on surface segmentation by approximation with several kinds of primitives. In [CSAD04], planes are being fitted, [WLK05] uses a collection of CAD primitives, such as spheres or rolling ball blends.

A tiling of a given model into nearly-developable charts is done in [JKS05] and [STL06]. Developable surfaces can be made out of a sheet of paper. This kind of chart tiling enables to make a papercraft model of the given surface.

Another work [LPRM02] uses 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 are constructed, which meet at these features.

There are many approaches on computing a feature layout using Morse theory. In [DBG+06], an eigenvector of the Laplacian is computed and used as Morse function. The Morse complex which is then build from this Morse function, segments the surface into quads. In [EHZ01] and [CCL03], 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 [Ede05], a curvature based Morse function is used to construct a Morse-Smale complex which aligns to surface features.

We also need to smooth patch boundary curves. In [LL02] 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 [HP04b]. Further the alignment of curves along features can be driven by the use of the feature sensitive metric introduced in [PSH+04].

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. In general these building blocks are detected in a time consuming process by hand. In our work, we demonstrate how to create a consistent patch layout on CAD surfaces in a reliable and fast way.

Our main contributions can be summarized as follows:

- an algorithm to generate a net of curves, running along surface features
- 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, where the single parts are encircled by smooth boundaries aligned to nearby surface features

Starting with a triangle mesh as shown in fig.1 we will end up with a decomposition like the one in fig.2.

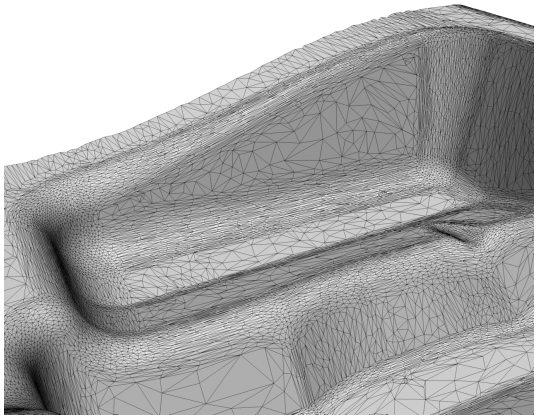

Figure 1: Typical CAD part as triangle mesh.

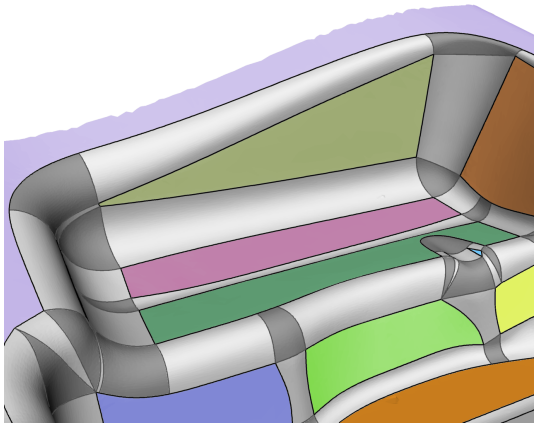

Figure 2: Patch layout of the CAD part from figure 1.

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. Finally, results of our tests are given in section 5.

2 Setting

The two main structures needed for our algorithm are the feature graph (fig.3, left) and the patch layout. Both encode a decomposition of the surface, i.e. have a graph like structure. As will be shown the most important structure for a proper layout is the feature graph. This structure encodes all the relevant surface information needed to create the final decomposition, i.e. out of this graph we create regions encircled by a set of boundaries, meeting our smoothness and alignment requirements. The final patch layout is used to denote the actual cell decompostion of the surface, where each cell can be assigned a CAD type such as face, fillet or node area. For a complete description of our layout generation method we define:

Feature graph. A graph on the surface (fig.3, 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 with a plane-like inner part as well as the tendency to be curved along its boundary
Feature edges	Smooth edges which separate two adjacent faces and correspond to cylindrical regions.
Node points	Isolated points, which correspond to spherical/hyperbolic regions and where several feature edges meet.

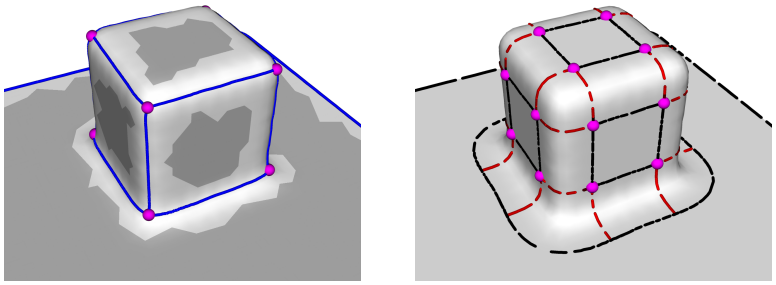

Figure 3: Left: Feature graph on CAD part consiting of faces, feature edges and node points. The dark grey parts within each face denote the plane-like or weakly curves parts whereas in the light grey part the surface starts to get curved. Right: Patch layout with face patches, fillets and node patches, as well as offset and node curves and offset nodes.

Patch layout. A graph on the surface (fig.3, right), which decomposes the surface into various cells. In contrast to the faces of the feature graph, which can be curved along their boundaries, within each cell 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 faces. 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. an offset curve results from parallel translation of 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 boundaries 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 based on our geometric requirements, where node points should be placed and feature edges are expected. So given a triangulated mesh M we present a strategy to built all parts of a feature graph, such as faces, feature edges and nodes. The basic idea is to detect plane-like regions I_i on M and expand these regions to cover all of M. So we will have a one-to-one relation between initial regions I_i and expanded regions R_i. Having covered all of M we detect node points. This gives us a first approximation to our final feature graph, because at this stage the node points are connected by edge based polygons, which 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 follows:

Algorithm 1: Generate feature graph

Input: triangle mesh M
Output: feature graph
1 Compute principle curvatures
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 [HP04a]. i.e. a stable and reliable method where no fitting needs to be performed. Other methods (e.g. [CSM03, PWY$^+$07]) 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.3 left fig.4 left). We use a curvature treshold τ to characterize all triangles as being part of an initial face. Thus we define the following set of flat triangles $F := \{\text{triangle } T \,|\, |k_{max}| < \tau\}$. In general, F can be split into a set of simply connected components I_i, i. e. $F = \{I_0, \ldots, I_n\}$.

3.3 Expand initial regions

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 fixed during the rest of the algorithm, whereas the alignment of the feature edges get adapted later in a smoothing step. In order to ensure the correct placement of feature graph nodes and edges, we developed a special growing strategy based on curvature information.

Feature function. Our feature function is required to drive the expansion process in two ways. We expect feature graph edges mainly to run along a ridge or in our case, where $|\kappa_{max}|$ is high. Further feature graph nodes should be placed within regions of high Gaussian curvature K_G. Therefore we propose

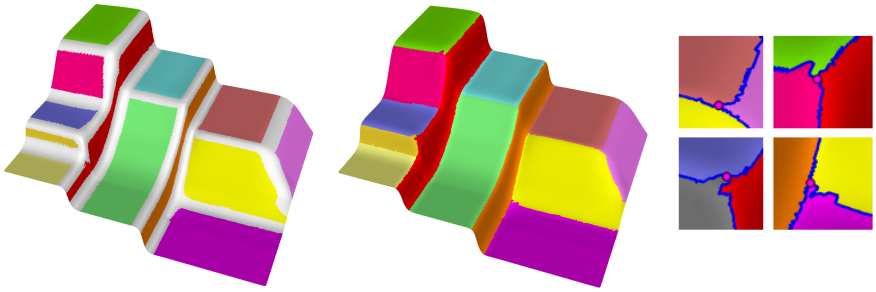

Figure 4: Left: Initial regions I_i. Middle: Complete covering of M by expanded regions R_i. Right: Parts of the unsmoothed feature graph.

a growing strategy which uses $|\kappa_{max}|$ values within cylindrical, i. e. fillet-like, regions, whereas in spherical/hyperbolical regions K_G values should drive the expansion.

Region growing. We classify the free triangles, i. e. not assigned to one of the initial regions I_i, into the following two categories:

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

So our region growing will be driven by κ_{max} for $T \in N$ and K_G for $T \in E$. First the initial regions are expanded into the set N 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/hyperbolic character, i. e. into E, using $|K_G|$ as the growing function.

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 $R = \{R_1, \ldots, R_n\}$ with $\cup_{i=1}^{n} R_i = M$

```
Initialize set of patches R = ∅
foreach Region I_i do
    Mark all T ∈ I_i as members of region R_i
    add region R_i to R
end
regionGrowing(R, k_max, N,)
regionGrowing(R, k_G, E)
```

Procedure "`regionGrowing`(R, f, C)"

Input: Set of regions $R = \{R_1, \ldots, R_n\}$, feature function f, one region into which growing is allowed C

Output: Expanded regions R_i with $C \subset \cup_{i=1}^{n} R_i$

```
1  PriorityQueue queue;
2  foreach triangle T ∈ R_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' ∈ C then
9  |  |  |  Mark T' as member of patch R_i;
10 |  |  |  queue.enqueue((T', i), key=f(T'));
11 |  |  end
12 |  end
13 end
```

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.4 right). An algorithm for our two step growing strategy is given in alg.2, also containing the details of our actual region growing procedure. The method is similiar to a watershed technique from image segmentation, see e.g. [MW99], 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.4 right, fig.5 left) - a set of polyons, where each polygon runs along edges of the underlying triangulation. Because a feature graph with smooth edges is neccessary to end up with a consistent patch layout we need to smooth these polygons (fig.5 right).
Smoothing energy. If the feature edges just get smoothed, e.g. by Laplace smoothing, they would leave the 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 our already detected node points and being aligned to the X_{min} field within that curved area.

In general our alignment energy for a curve γ can be defined with respect to

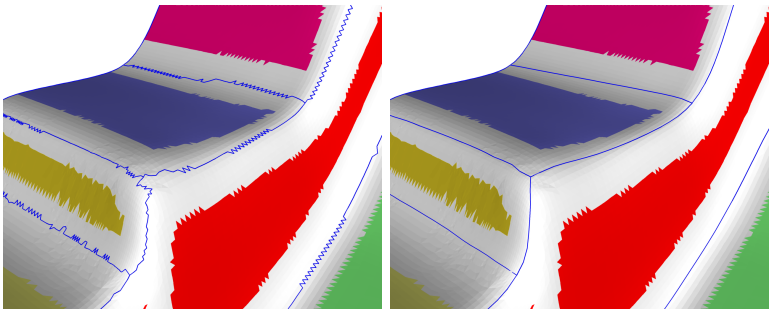

Figure 5: Smooting feature curveswith end points held fixed. Left: Edge based region boundary. Right: Smoothed region boundary.

a given a 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 polygon, 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 polygon, whose vertices lie on the surface. The edges are not forced to stay on the surface. Assuming, that the vectorfield X is locally nearly parallel (its covariant derivative vanishes), the variation of the energy (eq.1) at vertex $v \in \gamma$ is approximated by

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

In (eq.2) $X_w = (X(\gamma(w)) + X(\gamma(v)))/2$ is the mean vector of X at the edge (v, w) and $e_w = \gamma(w) - \gamma(v)$. Applying several steps of an explicit Euler method leads to smooth feature curves (fig.5 right).

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

Algorithm 4: Patch Layout

Input: Feature graph
Output: Patch layout

```
foreach Node point of the feature graph do
    Compute all its offset nodes
end
foreach Face F_i of the feature graph do
    foreach Feature graph edge γ_j bounding F_i do
        Determine offset direction dir = getSide(γ_j, F_j) ∈ {left, right}
        Compute distance map d_j^dir
        Smooth distance map
        Compute offset curves δ_j^dir
    end
end
Compute node curves
```

cutting of areas around its nodes. The complete algorithm for the computation of patch layout related curves and nodes is given in alg.4.

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.6).

4.1 Offset nodes

Offset nodes are points on the surface where offset curves and node curves meet. By construction a region is bounded by a set of feature edges, which start and end in node points. So to each face of the feature graph exists an associated set of node points. For each associated node point an offset node gets created (fig.6, top left). A canonical choice for an offset node within a certain face is the point closest to the node within the flat or weakly curved part, i.e. the corresponding initial region I. 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.7 left).

4.2 Offset curves

Each feature edge γ_i gets offset 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 remainding section we skip

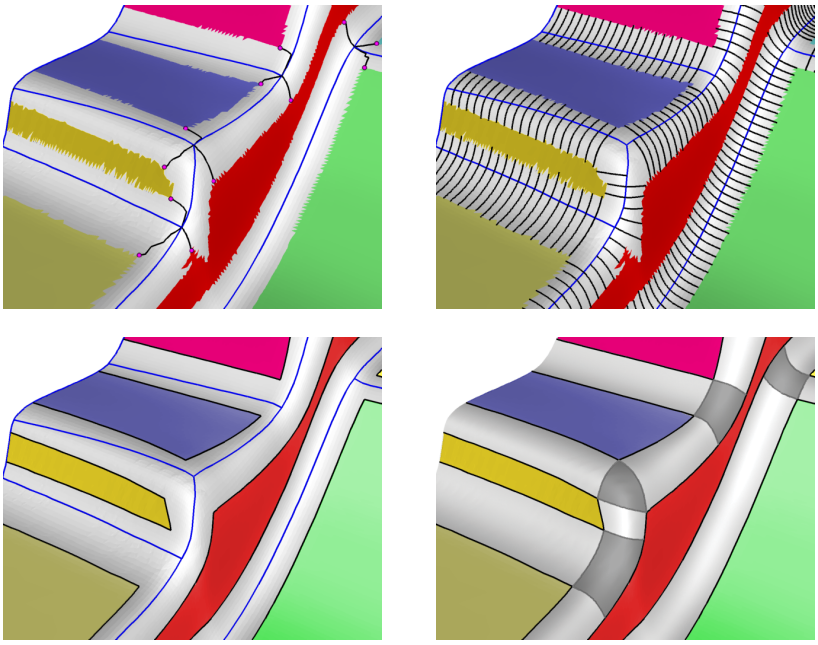

Figure 6: Generation of offset curves. Top left: Compute offset nodes as nearest points to feature nodes in each adjacent initial patch I_i. Top right: Regard distance to initial patch I_i as a one dimensional graph over the feature line. Bottom left: Smooth and aligned offset lines. Bottom right: Final offset layout after smoothing the distance function.

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 an 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: $d(t)$ is represented by a set of points uniformly distributed along the feature edge. Measuring distances is done along rays which start at these points and point into the face. How to extend a ray geodesically can be found in [PS98]. We also restrict the domain of $d(t)$ to a subset of γ, because close to offset nodes distance measures of different feature edges would overlap, see (fig.7).

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. 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. 7, right). When keeping these values fix during the smoothing process, the resulting offset curves will start and end in offset nodes. Thus, we have to

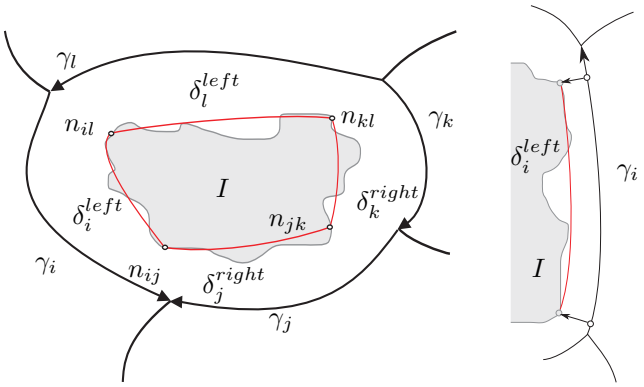

Figure 7: Left: Initial patch I of one face with adjacent feature curves γ_i and corresponding offset curves δ_i^{dir} and offset nodes n_{ij}. Right: Offsetting a feature curve γ_i, definition of the distance function $d_i^{dir}(t)$ only on a subset of γ_i

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 these sequence our polygonal offset curve lying on the surface is created (fig. 6, 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 illsustrated in (fig.8) 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 arised 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 seperate the node area from the fillet corresponding to γ. 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. 8, 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

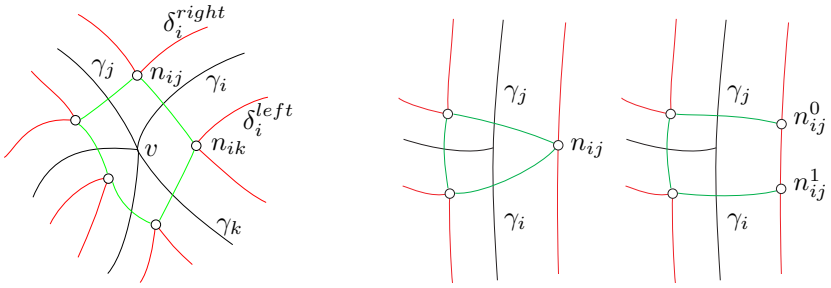

Figure 8: Left: Node area. The node lines (green) connect the endpoints of offset curves. Right: Two feature edges meet at node point with angle close to 180, the corresponding offset node gets spilt into two new ones.

map. We look for the first point within the valid range of $d(t)$. The actual node curve is constructed as in the ususal case.

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.9). 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 algorithm 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.10). Common state of the art industrial software, such as Geomagic ® , produces similiar feature graphs. It could be used as a possible starting point for our patch layout method. Thus, having an existing reverse engineering pipeline based on graph like structure our method could easily be plugged in to create a 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 [HP04a] or the noise can captured by our treshold defining the flat parts.

6 Acknowledgement

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

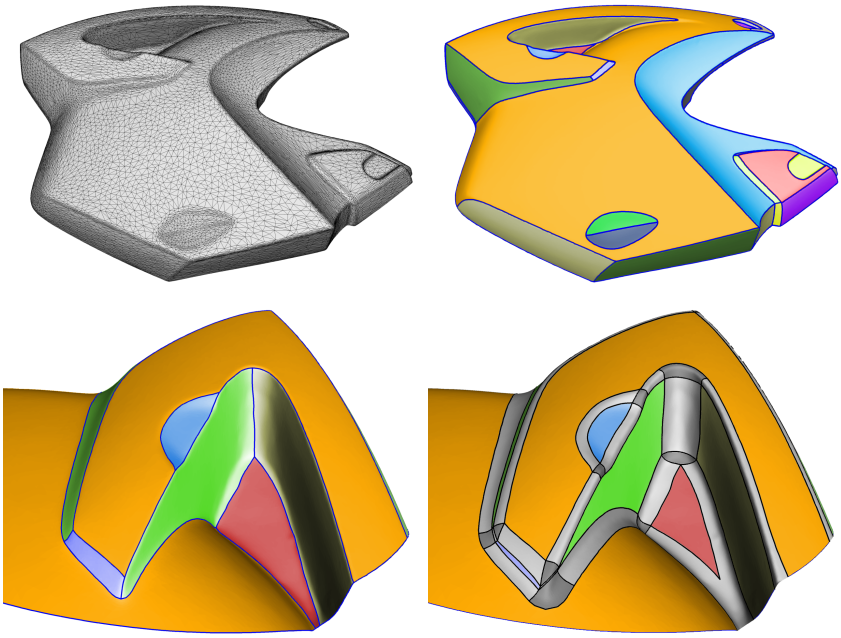

Figure 9: Feature layout on the BMW motorcycle part. Top: given triangulated model. 2nd, 3rd: Feature graph from our method. Bottom: Final patch layout.

References

[CCL03] 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.

[CSAD04] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation, 2004.

[CSM03] 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.

[DBG+06] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J.C. Hart. Spectral surface quadrangulation. *ACM SIGGRAPH*, 2006.

[Ede05] 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.

[EHZ01] 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.

[HP04a] Klaus Hildebrandt and Konrad Polthier. Anisotropic filtering of non-linear surface features. *Computer Graphics Forum*, 23(3):391–400, 2004.

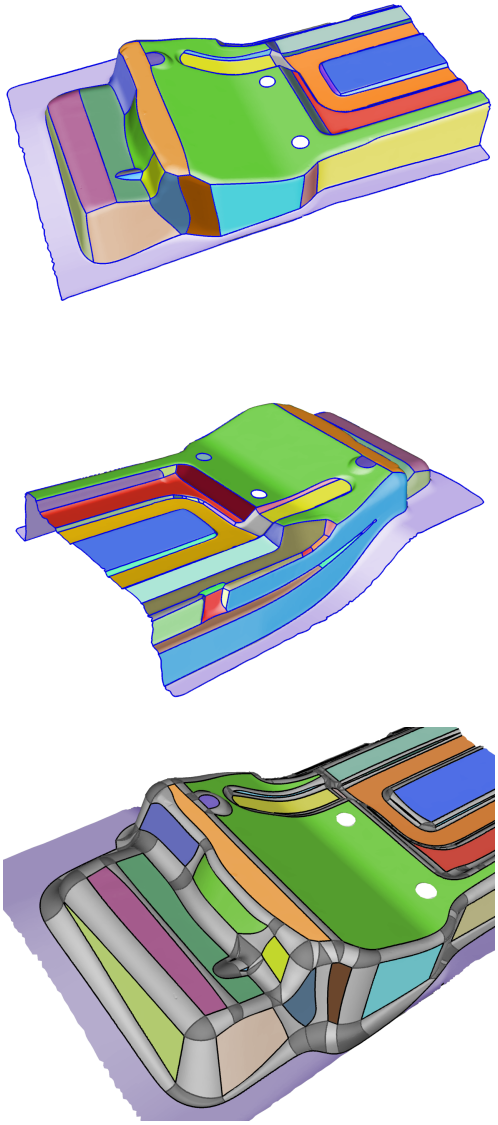

Figure 10: Top: Feature graph on CAD modell. Bottom: Offset curves.

[HP04b] 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.

[JKS05] 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.

[LL02] Yunjin Lee and Seungyong Lee. Geometric snakes for triangular meshes. *Computer Graphics Forum*, 21(3):229–238, 2002.

[LPRM02] 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.

[LRL06] Wan-Chiu Li, Nicolas Ray, and Bruno Lévy. Automatic and interactive mesh to T-spline conversion. In *Proc. Eurographics/ACM Symp. on Geom. Proc.*, pages 191–200, 2006.

[MW99] 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.

[PS98] 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.

[PSH+04] 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.

[PWY+07] 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.

[SAKJ01] Jami J. Shah, David Anderson, Yong Se Kim, and Sanjay Joshi. A discourse on geometric feature recognition from cad models. *J. Comput. Info. Sci. Eng.*, 1(1):41–51, 2001.

[Sha06] Ariel Shamir. Segmentation and shape extraction of 3d boundary meshes. In *EUROGRAPHICS '06: STARS*, pages 137–149, 2006.

[STL06] Idan Shatz, Ayellet Tal, and George Leifman. Paper craft models from meshes. *Vis. Comput.*, 22(9):825–834, 2006.

[WLK05] Jianhua Wu Leif Kobbelt. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum*, 24:277–284(8), September 2005.