A Labeling Algorithm for Trimmed Surface Fitting

Márton Vaitkus[1], and Tamás Várady[2]

Budapest University of Technology and Economics
[1]vaitkus@iit.bme.hu, [2]varady@iit.bme.hu

Corresponding author: Márton Vaitkus, vaitkus@iit.bme.hu

ABSTRACT

Approximating irregular trimmed surface regions with tensor product parametric surfaces is a difficult problem. We search for surfaces that are aligned with the geometric features of the shape, and possess even curvature distribution beyond the trim curves, as well. Labeling is a useful technique by means of which certain segments of the trimming boundary are associated with the sides of the parametric surface to be fitted. In this way it is possible to orient the unknown surface and its 3D boundaries, and accordingly compute a good parameterization for the data points in the domain. This also makes it possible to extend the trimmed region into a quadrilateral using a set of artificial data points – first in 2D, then in 3D, as well. Techniques to perform label-driven parameterization and surface extension have been discussed in a recent paper [24], however, no explicit method was suggested to automatically define labels. In our paper we deal with this missing component and propose a new heuristic algorithm. First, we determine *label candidates* by various criteria. Then we estimate the location of *virtual corners,* and qualify them as weak or strong. Finally, the labeled segments are merged and/or discarded by various rules until we reach one of the *six admissible labeling configurations*. Our proposed algorithm yields natural labelings that mostly correspond to configurations defined by surfacing experts. This is illustrated by several examples. Surface fitting based on labeled parameterization leads to high-quality surfaces, well-suited for engineering applications.

Keywords: surface fitting, trimmed surfaces, constrained parameterization, labeling
DOI: https://doi.org/10.14733/cadaps.2019.720-732

1 INTRODUCTION

Approximating data points or triangle meshes with tensor-product (TP) parametric surfaces is a task of fundamental importance in computer-aided geometric design. Our particular interest relates to trimmed regions, bounded by an irregular multi-sided loop of boundary segments having no obvious tensor-product structure. There are two major applications motivating our work.

First, in reverse engineering [6], a CAD model is to be produced from a measured point cloud. A triangle mesh is generated, then segmented into disjoint regions, corresponding to the faces of the final B-Rep model. Each region is approximated by some surface type, such as simple primitives (planes, quadrics, etc.), procedural surfaces (sweeps, lofts, fillets, etc.), or else by truly free-form shapes, e.g. TP Bézier surfaces or NURBS, the representations used in commercial CAD systems and data exchange standards (IGES/STEP).

Second, there are non-standard surface representations, such as transfinite [25] and control-point based [26] multi-sided patches that possess advantageous properties compared to TP NURBS in certain modelling tasks. Nevertheless, for downstream CAD/CAM applications, the geometry eventually needs to be sampled and approximated by some standard surface.

While it is possible to approximate a complex region by means of a collection of quadrilateral surfaces over a topologically irregular network [10,5], in this paper we focus exclusively on fitting a single trimmed surface [18], thus avoiding the difficulties of creating supplementary structures for the quads and enforcing smooth connections.

Parametric surface fitting is a difficult problem that depends on various parameters, including error tolerances, knot vectors, regularization weights, etc. [27]. In order to formulate it as a linear least-squares problem, 'appropriate' *(u,v)* parameter values must be assigned to the data points, as this will fundamentally determine the qualities of the final surface. Previous research on initial data parameterization has been conducted along three major directions. (i) There exist parameterization methods that presume that *all four* TP boundaries are present [3,15,17,19], or that the region has some *special structure*, e.g. it is swept [4,14,23], or singly-curved/developable [21]. (ii) Other, less restrictive methods *project* the points onto some primitive surface for trimmed parameterization [27]; in many cases just the best fit plane or cylinder is used. These methods yield acceptable results for simple geometries, but otherwise projections may lead to extreme distortions, or even fail to be one-to-one. (iii) *Mesh parameterization* is a topic with a vast and diverse literature [12], including methods that minimize geometric distortions [22,16], or align parameterization isolines with geometric feature curves [13,7]. These approaches are mainly used for texture mapping or quadrilateral remeshing; however, in the surface fitting context the distortion of the parameterization is not a primary concern, and alignment with a set of curvature lines might be unsolvable due to umbilical points in the interior of the surface.

Our goal is to create doubly curved trimmed patches, where simple parameterization methods are not satisfactory. We wish to find a good orientation for the surface to be fitted and determine appropriate placements for its virtual boundaries. Accordingly, the structure of the isolines is supposed to be well-aligned with the geometric features of the shape. We need an even curvature distribution not only within the trimmed region, but also in the extended surface areas beyond the trim curves. Overall, we wish to compute parameterizations that yield surfaces well-suited for CAGD applications.

In the majority of cases no single 'best' TP patch exists for a given trimmed region, and selecting the best possible configuration is often a subjective and application-dependent issue. There is an incredible richness of case studies and tutorials on practical surfacing on the Internet. The well-known commercial reverse engineering systems (Geomagic Design X, Autodesk Powershape, CATIA, and others) offer lots of complex operations to approximate triangle meshes with parametric surfaces. At the same time, we have found that producing high-quality surfaces in the above sense is not easy. Surfaces are often parameterized by projective methods, and then the only option to set the flow of the isolines is to preset a local 2D coordinate system, placed in the middle of the region [2,11]. In other cases, the surface is oriented using feature curves extracted automatically or explicitly drawn by the user [1,8]. Finally, if results are not satisfactory, the geometry is simplified by further manual segmentation, using dominantly quadrilaterals.

In a recent paper [24], we have introduced new techniques that - according to our best knowledge – improve upon current techniques and meet complex requirements in trimmed surface fitting. The fundamental idea of *labeling* is to assign certain boundary segments to the sides of the domain rectangle. Labels were assumed to be given as part of the input, e.g. defined manually by the user. The automatic assignment of labels was identified as the most important avenue for future research.

In the current paper, we describe an algorithm for automatic labeling. As we are not aware of any theory that characterizes an 'ideal' parameterization for trimmed regions, here we propose a heuristic method. We do not claim to produce the 'best' possible solution in every scenario, as labeling often involves subjective and application-dependent considerations, however our results generally meet engineering expectations.

In Section 2, we briefly summarize our work published in [24] and describe the concept of labeling. Then, in Section 3, we present the new labeling algorithm. Finally, in Section 4, we demonstrate the effectiveness of our approach by a few examples.

2 LABELED PARAMETERIZATION AND EXTENSION FOR TRIMMED SURFACE FITTING

In [24], we introduced pre-processing techniques to support TP fitting for trimmed patches. Our aim was two-fold: (i) to help in orienting the quadrilateral surface based on the geometry of the patch, leading to a simpler and better aligned control grid; (ii) to avoid "weak" control points that have only a few data points in their support and thus could lead to unstable, oscillating fits [27].

Labeling is a powerful technique to orient the yet-unknown surface and it narrows down the set of possible parameterizations. We can assign labels to particular boundary segments – using our notations – North, West, South, East, prescribing that a segment must lie somewhere on the boundary of the TP surface to be fitted. With other words, certain segments are mapped to particular sides of the domain rectangle. Other segments may remain Unlabeled.

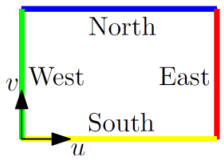
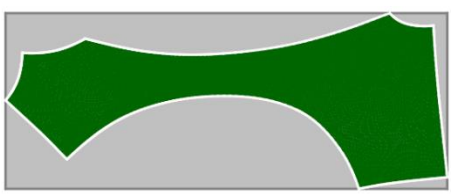


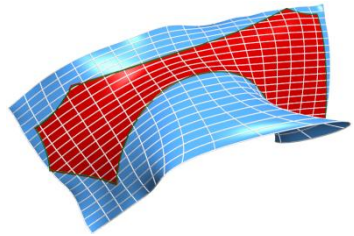
Figure 1: Labels.

In [24], it was presumed that labels have already been defined, and we pre-processed the data in two phases prior to fitting. First a *guiding frame* was constructed, which extrapolated the labeled segments into a four-sided virtual boundary loop, then the data was *parameterized* by a constrained optimization of the As-Rigid-As-Possible distortion energy [16]. Second, the 2D triangle mesh in the *(u,v)* plane was *supplemented* so that the entire domain rectangle was covered, and this was inversely *mapped back to 3D*, optimizing an energy that balanced the smoothness of the extrapolated surface and the fairness of its boundary curves. Experiments in [24] have shown that – given judiciously placed labels – constrained parameterization and extension produces accurate trimmed fits with simple and well-oriented control nets and controlled surface geometry beyond the trimmed region.

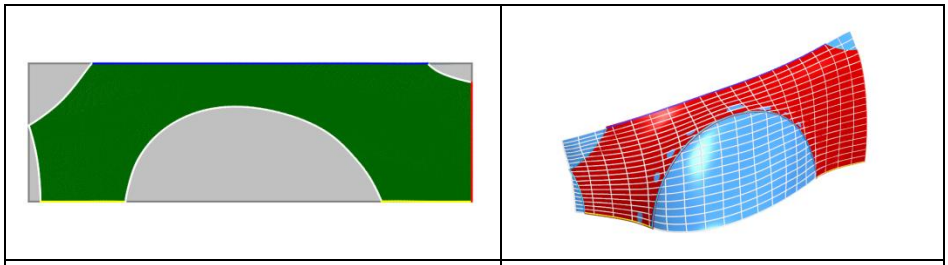
Fig. 2 demonstrates the advantages of this approach; a trimmed region representing a car body panel is to be approximated by a TP surface. Compare the parameterization and the fitted surface 'without' and 'with' labeling.



a) Parameterization without Labeling



b) Fit without Labeling



c) Parameterization with Labeling | d) Fit with Labeling

Figure 2: An example of labeled fitting.

3 THE ALGORITHM FOR AUTOMATIC LABELING

3.1 Overview

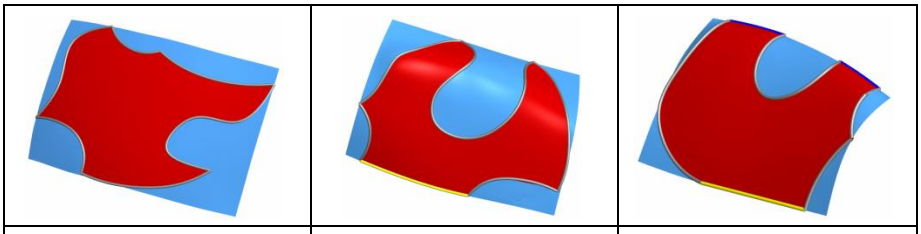
The input of our algorithm is a triangle mesh (manifold-with-boundary, orientable, possibly multiply connected) with its perimeter boundary loop segmented into a sequence of oriented polylines. Such segmentation of the boundary is naturally produced in the context of reverse engineering [6] and curve-network based surfacing [25,26].

The output is a labeling of the boundary segments as South, East, North, West, or Unlabeled. It is not necessary that all four directions are used, and the same label may be attached to more than one segment. Adjacent labels can form corners – either *real* corners when two labels share a common endpoint, or *virtual* corners formed by their extensions at the endpoints. Labels may terminate at a *no-corner*, when no sensible extension can be created. It can easily be deduced that only *six* possible label configurations exist by the number of labels (L) and corners (C), as shown by simple examples in Fig. 3. Our aim is to select the configuration best suited for the surface to be created.

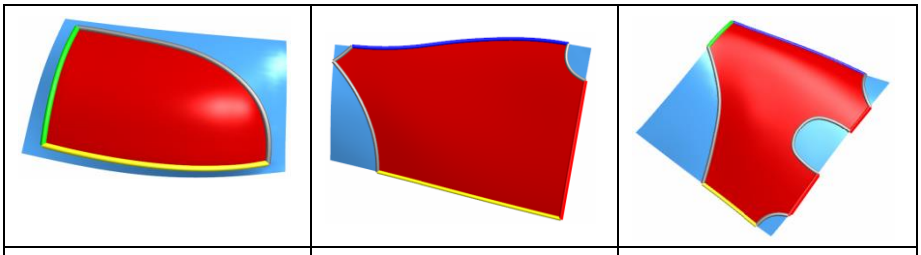
Our algorithm consists of the following steps:

i. Boundary segments that serve as *label candidates* are detected.
ii. Neighboring label candidates that form sufficiently 'weak' virtual corners are merged.
iii. Label candidates are removed, until an admissible configuration of labels and corners is reached.
iv. Labels are assigned according to the detected admissible configuration.

In the following sections, we are going to discuss the basic idea of each step.



a) $L = 0 / C = 0$ | b) $L = 1 / C = 0$ | c) $L = 2 / C = 0$



d) *L = 2 / C = 1*	e) *L = 3 / C = 2*	f) *L = 4 / C = 4*

Figure 3: Admissible label configurations.

3.2 Detecting Label Candidates

First, we identify those boundary segments which can serve as label candidates; more precisely, we exclude those segments that are not going to serve as boundaries of the presumed TP surface. We test two sets of criteria in sequence.

3.2.1 Concave Angles

Our trimmed region can be interpreted as a remainder, after cutting off parts from a 'convex' quadrilateral surface using Boolean operations that often produce areas with concave angles. Consequently, boundary segments meeting at concave angles need to be removed from the set of label candidates. Two such examples are shown as the green curves in Fig. 4, for the surface of Test Example 1. The corner angles are computed by adding up the mesh angles formed by subsequent edges meeting at the corner.

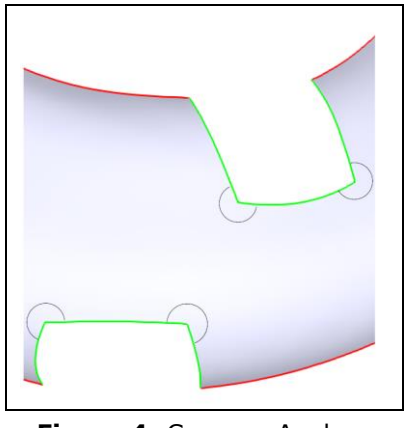


Figure 4: Concave Angles.

3.2.2 Moving Frame Rotation

Trimming a tensor product surface by intersecting it with another smooth surface might create trim curves without sharp or concave angles and these would not be detected by our previous test. In general, isoline boundaries tend to be geometrically simple curves, while trimmed boundaries obtained by surface-surface intersections may have more complex geometry [18]. To qualify the complexity of a boundary segment, we might imagine a sweep of orthogonal 'moving frames' defined by the tangents and the surface normals along the curve. For a complex trim curve, these frames will turn considerably by rotating in their tangent plane as we move along the curve; while for isolines they generally stay roughly parallel. An example is shown in Fig. 5a: here the blue vectors stay almost parallel moving from one end of a boundary segment to the other, the green and red arrows in contrast

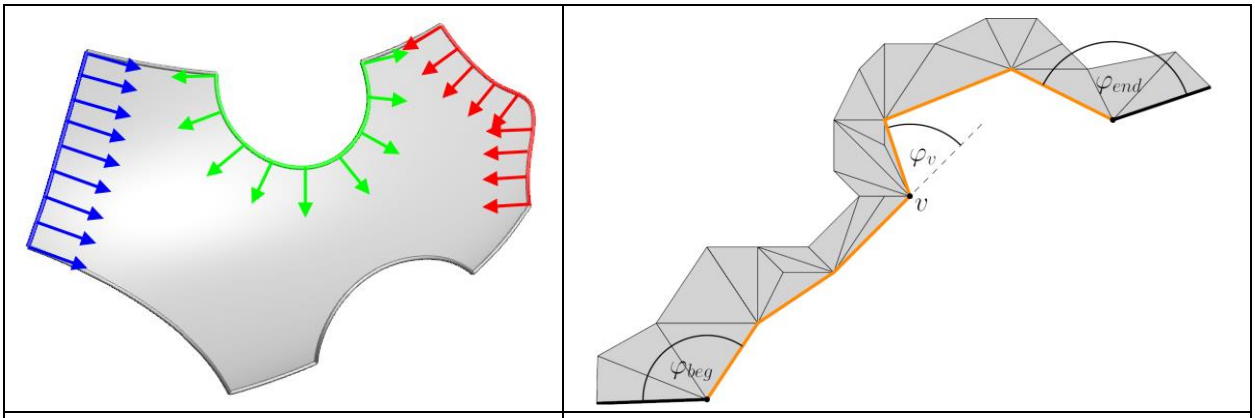
indicate large rotations. This can be formalized in mathematical terms, as follows. The cross product of the initial curve tangent direction and the surface normal defines a local coordinate system for its tangent plane. The triangle fans adjacent to the curve are isometrically flattened, by rotating their faces in sequence into this plane, as illustrated in Fig. 5b. Forming the sum of the (squared) turning angles between subsequent edges of this planar polyline, we effectively get an approximation of its (squared) geodesic curvature integral [9]:

$$\tilde{E}_{Label}(L) = \sum_{v \in L} (\varphi_v)^2 \approx \int_L {\kappa_g}^2(s) ds. \tag{3.1}$$

We also take into account the angles $\varphi_{beg}, \varphi_{end}$ spanned with the tangents of the adjacent boundary segments at the beginning and the end of the curve and define the label energy as

$$E_{Label}(L) = \tilde{E}_{Label}(L) + \left(\frac{\pi}{2} - \varphi_{beg}\right)^2 + \left(\frac{\pi}{2} - \varphi_{end}\right)^2. \tag{3.2}$$

Segments that have energy values above a certain threshold are then removed from the set of label candidates.



a) Moving frame rotations | b) Curve flattening

Figure 5: Assessing label candidates.

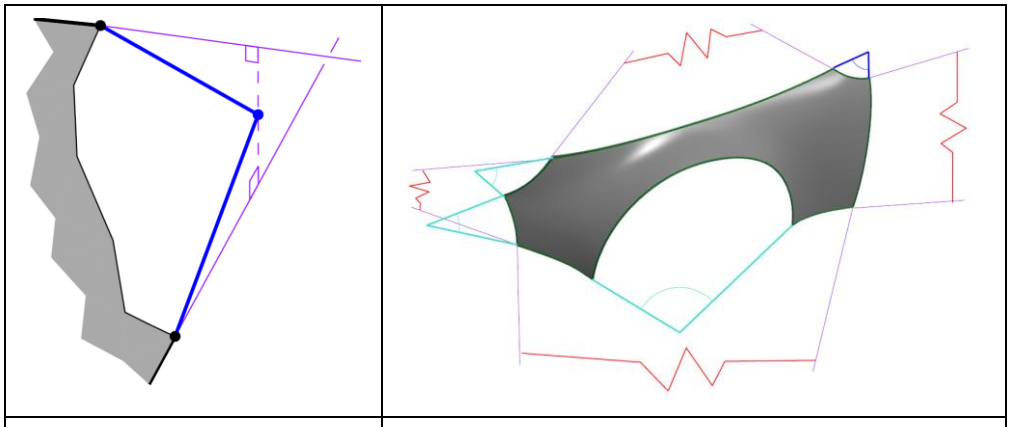
3.3 Classifying Virtual Corner Candidates

In the subsequent phases of our algorithm, we will merge and remove label candidates with the aim of arriving at one of the admissible label configurations. Decisions about which labels are to be merged or removed will be based on the properties of the real or virtual corners defined by pairs of label candidates.

We estimate the position of virtual corners by computing the normal transversal of the lines spanned by their opposing endpoint tangents. The estimated corner is formed at the midpoint of the transversal, as illustrated in Fig. 6a.

We presume that the missing corners of the tensor-product patch (i) are close to the trimmed region, (ii) form an angle that deviates from 90 degrees to a controlled extent, (iii) are relatively flat without highly curved portions. Our aim is to retain only those corners that satisfy these requirements. To do so, we choose to classify corners as either *concave*, *parallel*, *weakly convex*, or *strongly convex*.

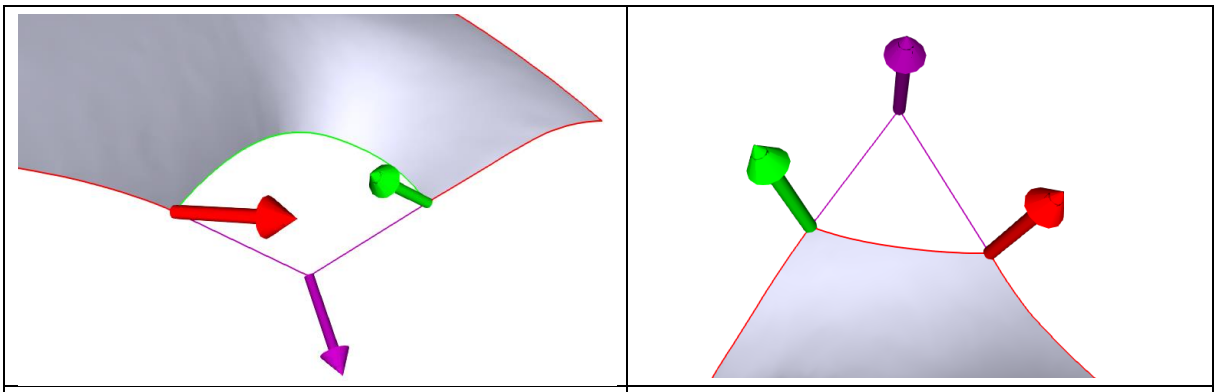
A virtual corner is classified *concave*, when the normal transversal lies in the opposite direction of one of the endpoint tangents. When a meaningful transversal exists, a virtual corner is classified *parallel* when its angle is sufficiently small, and the transversal is far away from the label endpoints. When a corner that is neither *concave* or *parallel* has an angle close to 90 degrees, it is classified *strongly convex*; and *weakly convex* otherwise. Some examples are shown in Fig. 6b: blue lines denote *strongly convex* corners, cyan lines denote those that are *weakly convex*, and purple lines terminated with red squiggles denote *concave* or *parallel* 'non-corners'.



a) Corner estimation | b) Virtual corners

Figure 6: Classifying corner candidates.

Even when the corner is classified *strongly convex* based on these criteria, the surface that fills in the region between the label candidates and the corner could be geometrically complex (e.g. of high Gaussian curvature), like in the case of the lowermost corner in Fig. 6b. To detect such undesirable corners, see Fig. 7, we consider the trihedral angle formed by the surface normals at the endpoints (colored green and red), and the normal vector of the virtual corner (colored purple). If the solid angle (or equivalently the area) of the corresponding spherical triangle is very large, as in Fig. 7a, the corner should be degraded to *weakly convex*.



a) Bad corner trihedron | b) Good corner trihedron

Figure 7: Corner trihedron tests.

We can formalize these criteria by defining an energy that measures the strength of a convex corner:

$$E_{Cor} = (1 - w_{Tri})E_{Cvx} + w_{Tri}E_{Tri}. \tag{3.3}$$

The first term measures the quality of the corner, and is itself composed of two terms:

$$E_{Cvx} = (1 - w_{Dist})E_{Ang} + w_{Dist}E_{Dist}, \tag{3.4}$$

where $E_{Ang} = \frac{|\pi/2-\theta|}{\pi/2}$ measures the deviation of the corner angle θ from a right angle, and $E_{Dist} = \frac{A_{Tri}}{A_{Surf}}$ is the area of the triangle formed by the corner and the label endpoints, normalized by the trimmed

surface area. The second term of the energy $E_{Tri} = \frac{\Omega}{2\pi}$ is the (normalized) solid angle Ω spanned by the normal vectors. The weights w_{Tri}, w_{Dist} control the trade-off between the different criteria.

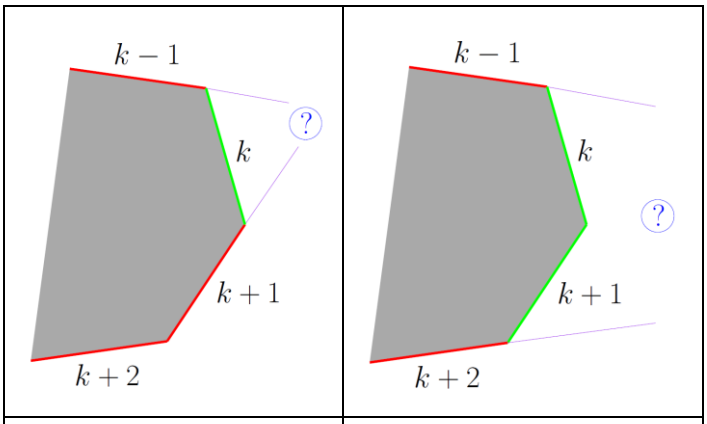
3.4 Concatenating Label Candidates

After executing the tests of Sec. 3.2., we have *L* remaining label candidates, and 'a priori' *C=L* real or virtual corners. In the next phase of our algorithm, we take pairs of neighboring label candidates, and decide whether they need to be concatenated into *multi-segment* labels or define a proper (virtual) corner. With the corners classified, we remove those that are *concave* or *parallel* (*C → C-1, L → L*), and concatenate label candidates through *weakly convex* corners with obtuse angles (*C → C-1, L → L-1*). Consecutive segments of multi-segment labels are smoothly connected with cubic Hermite spline curves, so that the label forms an unbroken polyline. This scenario occurs in the test examples of Section 4. Hereinafter, the retained *strongly convex* corners will be referred simply as 'corners'.

3.5 Removing Label Candidates

After excluding weak corners, the result might still not correspond to any of the admissible label configurations, so further removal of label candidates is required.

One possible operation we perform is removing label candidates one by one. We assume that very long trim curves were removed earlier due to their geometric complexity, so we consider label candidates for removal in order of increasing length. To decide whether a label candidate is ought to be removed, we carry-out a hypothesis-test, as illustrated in Fig. 8a: assuming we wish to remove a label *k*, a virtual corner is determined by the extensions of labels *k-1* and *k+1*. If this hypothetical corner can be classified as *strongly convex* by having a low value of E_{Cor}, we remove the label under consideration (*L → L-1, C → C-1*). Otherwise, we move on to test the second shortest label candidate, and so on. If none of the hypothesis tests result in a *strongly convex* corner, we remove the label that creates the corner with the lowest value of E_{Cor}.



a) Label removal | b) Corner removal

Figure 8: Label candidates (red), segments to be removed (green).

Another kind of operation that we may perform is removing a corner together with its two neighboring label candidates, thus creating a *parallel* corner (*L → L-2, C → C-3*). This might be preferable, if such an operation would immediately result in an admissible label configuration, i.e. when we are in a situation with *L=5, C=5* (that would become *L=3, C=2*) or *L=4, C=3* (that would become *L=2, C=0*). In these cases, if a hypothesis test for removing the shortest label fails, a different sequence of hypothesis-tests is performed: we remove two adjacent labels *k*, *k+1*, and consider the virtual corner of labels *k-1* and *k*+2, as illustrated on Fig. 8b. If the resulting corner is *parallel*, both

labels are removed, along with their corresponding corner. The corners are tested in order of decreasing values of E_{Cor}. If no label candidate pair can be removed to create a *parallel* situation, we proceed by removing one of the labels, as before.

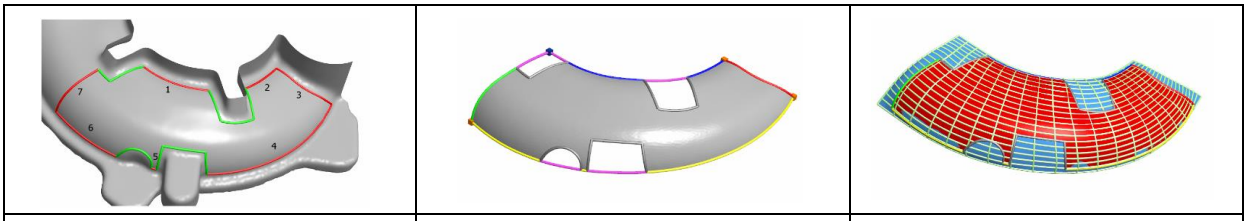
These steps are iterated until we arrive at one of the six admissible configurations, when all the boundary segments get labeled accordingly. Note, that for intermediate configurations *C<=L* always holds.

4 TEST EXAMPLES

In this section we demonstrate the algorithm by means of five examples.

4.1 Example 1

This is a relatively straightforward configuration (Fig. 9). The concave parts fall out leaving seven label candidates. 1-2 and 4-5-6 are concatenated, yielding two multiple-segment labels. 2-3, 3-4 and 6-7 define real corners. 7-1 is a strong virtual corner computed by extending and snapping the related label curves. Thus we have four labels and four corners, and the algorithm terminates.

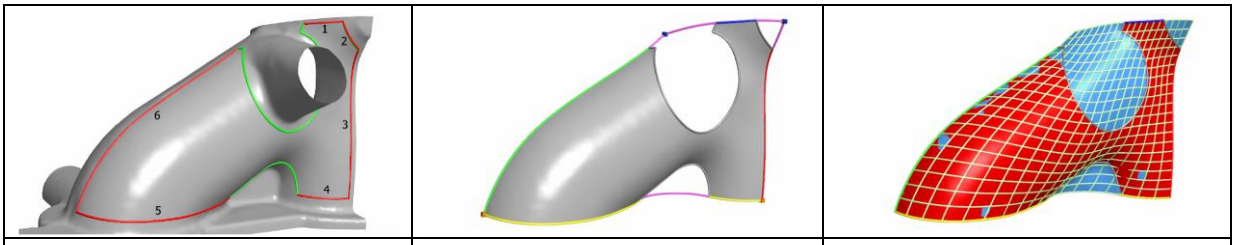


a) Mesh and boundary loop, label candidates - red	b) Label segments concatenated, virtual corners detected	c) Labeled surface fitted

Figure 9: Example 1.

4.2 Example 2

This configuration is more complex (Fig. 10). The concave parts fall out, leaving six label candidates. 4-5 are concatenated yielding a multi-segment label at the bottom. Then - in the first round - there are four real corners: 1-2, 2-3, 3-4 and 5-6, and a single virtual corner 6-1. Five labels and five corners do not form an admissible configuration, thus the algorithm attempts to remove one label and one corner. In the second round, the algorithm evaluates the cost of removing each label, and measures which newly computed virtual corner would be the strongest. Deleting label 1, 3, 4-5 or 6 would yield weak virtual corners, while deleting label 2 leads to a good final solution.

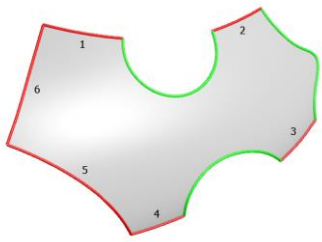


a) Mesh and boundary loop, label candidates - red	b) Labels concatenated, virtual corners detected	c) Labeled surface fitted

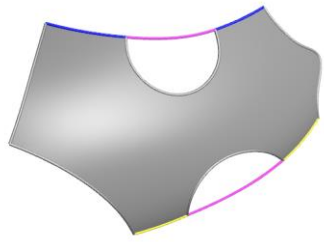
Figure 10: Example 2.

4.3 Example 3

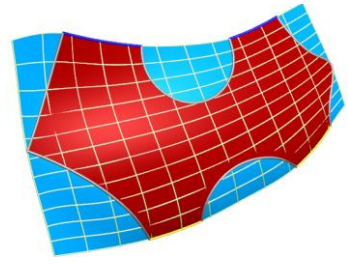
In this example (Fig. 11), after deleting segments with large moving frame rotation we obtain six label candidates. 1-2 and 3-4 are concatenated, resulting in four labels and three real corners – a non-admissible configuration. Deleting either label 5 or 6 would result in a weak corner, while removing both results in parallel labels, i.e. a configuration with two labels and no corners.



a) Mesh and boundary loop, label candidates - red



b) Labels concatenated, no virtual corner detected

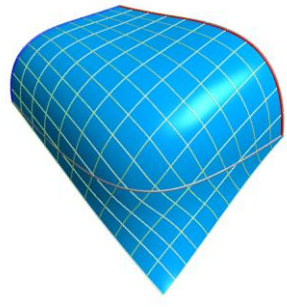


c) Labeled surface fitted

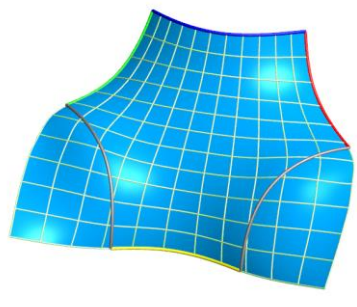
Figure 11: Example 3.

4.4 Example 4 – Ambiguous Cases

There exist ambiguous cases, where it is impossible to select a best labeling configuration, and the algorithm must recommend one of the possible solutions in an 'ad hoc' manner. Typical examples include highly symmetrical configurations, such as those shown in Fig. 12. The first surface has three labels and three right-angled corners, also the endpoint tangents are close to parallel, thus deleting either label would result in undefined, distant corners. In Fig. 12a the bottom label was chosen to be removed in order to obtain an admissible configuration *(L=2, C=1).* In the case of the hyperbolic saddle shown in Fig. 12b, each of the six boundary segments are approximately geodesic with every corner close to 90 degrees; very weak corners would be created whichever label is removed. One possible labeling is shown in Fig. 12b, however, its symmetric counterparts would also be appropriate. In case of ambiguous configurations either all labeling cases are acceptable for surface fitting or the user has to select explicitly the most favored configuration.



a) Three-sided region

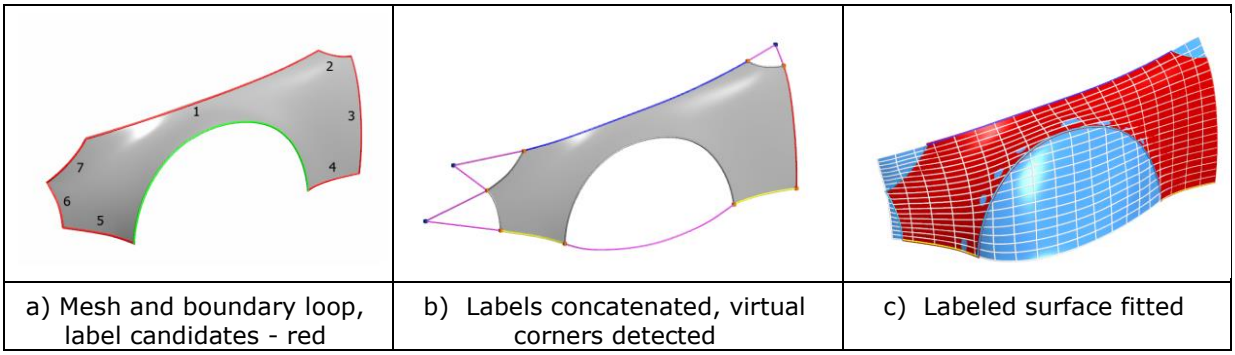


b) Six-sided region

Figure 12: Example 4.

4.5 Example 5

Finally, let us take the surface shown in Section 2 (Fig. 13). Seven label candidates are determined, where 4-5 define a weak virtual corner and gets concatenated, while the other label pairs define real corners yielding six labels with six corners in the first round. The algorithm defines label 2 as to be removed, since labels 1 and 3 define a strong virtual corner. Retaining label 6 and deleting 7, or retaining 7 and deleting 6 would yield poor configurations with weak virtual corners, so the algorithm deletes both and terminates with a 'three labels - two corners' configuration.



a) Mesh and boundary loop, label candidates - red	b) Labels concatenated, virtual corners detected	c) Labeled surface fitted

Figure 13: Example 5.

5 CONCLUSIONS AND FUTURE WORK

An algorithm to determine a preferred set of labels for facilitating trimmed surface fitting by tensor product parametric surfaces has been presented. Automatic labeling and the corresponding parameterization helps to fit a collection of high-quality surfaces without manual interaction, often requested in applications, such as reverse engineering or the conversion of non-standard surface representations.

We have dealt only with boundary curves and their close vicinity, however considering interior feature curves for labeling is a challenging problem for future research. Another area is to deal with a collection of trimmed surface elements. This may help to better distinguish between feature curves and trimming curves, and may produce labeling not by local geometric tests, but by some higher-level structural analysis. Besides least-squares TP fitting, other surface reconstruction techniques might benefit from the existence of boundary labels. For example, they may be used for the detection and reconstruction of profile or spine curves of sweeps [4,14,23], or serve as guiding constraints in the "active surface" approach [20].

ACKNOWLEDGEMENTS

This project has been supported by the Hungarian Scientific Research Fund (OTKA, No.124727). We acknowledge several thought-provoking technical discussions with our colleague, Péter Salvi. All images were generated by the Sketches prototype system (ShapEx Ltd, Budapest); special thanks are due to György Karikó for his exceptional development contribution to this project.

Márton Vaitkus, http://orcid.org/0000-0003-2064-763X
Tamás Várady, http://orcid.org/0000-0001-9547-6498

REFERENCES

[1] Autodesk PowerShape Manual. https://knowledge.autodesk.com/support/powershape/learn-explore/caas/CloudHelp/cloudhelp/2018/ENU/PWRS-ReferenceHelp/files/GUID-A1165E64-B7E5-4BAC-A17B-5EBE4792AA60-htm.html

[2] Autodesk PowerShape Tutorial. https://www.youtube.com/watch?v=vaiU0FMRsSM

[3] Azariadis, P. N.: Parameterization of clouds of unorganized points using dynamic base surfaces, Computer-Aided Design, 36(7), 2004, 607-623. https://doi.org/10.1016/S0010-4485(03)00138-6

[4] Bartoň, M.; Pottmann, H.; Wallner, J.: Detection and reconstruction of freeform sweeps, Computer Graphics Forum, 33, 2014, 23-32. https://doi.org/10.1111/cgf.12287

[5] Bommes, D.; Vossemer, T.; Kobbelt, L.: Quadrangular parameterization for reverse engineering, Proc. 7th International Conference on Mathematical Methods for Curves and Surfaces, Tønsberg, Norway, 2008. https://doi.org/10.1007/978-3-642-11620-9_5

[6] Buonamici, F.; Carfagni, M.; Furferi, R; Governi, L.; Lapini, A.; Volpe, Y.: Reverse engineering modeling methods and tools: a survey, Computer-Aided Design and Applications, 15(3), 2018, 443-464. https://doi.org/10.1080/16864360.2017.1397894

[7] Campen, M.; Ibing, M.; Ebke, H.-C.; Zorin, D.; Kobbelt, L.: Scale-Invariant Directional Alignment of Surface Parametrizations, Computer Graphics Forum, 35(5), 2016, 1-10. https://doi.org/10.1111/cgf.12958

[8] CATIA V5 Manual. http://catiadoc.free.fr/online/qsrug_C2/qsrugbt0602.htm

[9] Crane, K.; Wardetzky, M.: A Glimpse of Discrete Differential Geometry, Notices of the AMS, 64(10), 2017. https://doi.org/10.1090/noti1578

[10] Eck, M.; Hoppe, H.: Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Type, Proc. 23rd annual conference on Computer graphics and interactive techniques, 1996, 325-334. https://doi.org/10.1145/237170.237271

[11] Geomagic Design X Tutorial. https://www.youtube.com/watch?v=9D6yhPrk4Ys

[12] Hormann, K.; Lévy, B.; Sheffer, A.: Mesh Parameterization: Theory and Practice, ACM SIGGRAPH 2007 Courses, 2007. https://doi.org/10.1145/1281500.1281510

[13] Knöppel, F.; Crane, K.; Pinkall, U.; Schröder, P.: Globally optimal direction fields, ACM Transactions on Graphics (ToG), 32(4), 2013. https://doi.org/10.1145/2461912.2462005

[14] Kovács, I.; Várady, T.: Reconstructing swept surfaces from measured data, Proc. 14th IMA Conference on Mathematics of Surfaces, Birmingham, UK, 2013.

[15] Lai, Y. K.; Hu, S. M.; Pottmann, H.: Surface fitting based on a feature sensitive parametrization, Computer-Aided Design, 38(7), 2006, 800-807. https://doi.org/10.1016/j.cad.2006.04.007

[16] Liu, L.; Zhang, L.; Xu, Y.; Gotsman, C.; Gortler, S. J.: A Local/Global Approach to Mesh Parameterization, Computer Graphics Forum, 27(5), 2008, 1495-1504. https://doi.org/10.1111/j.1467-8659.2008.01290.x

[17] Ma, W.; Kruth J. P.: Parameterization of randomly measured points for least squares fitting of B-spline curves and surfaces, Computer-Aided Design, 27(9), 1995, 663-675. https://doi.org/10.1016/0010-4485(94)00018-9

[18] Marussig, B.; Hughes, T. J.: A Review of Trimming in Isogeometric Analysis: Challenges, Data Exchange and Simulation Aspects, Archives of Computational Methods in Engineering, 2017, 1-69. https://doi.org/10.1007/s11831-017-9220-9

[19] Piegl, L. A.; Tiller, W.: Parametrization for surface fitting in reverse engineering, Computer-Aided Design, 33(8), 2001, 593-603. https://doi.org/10.1016/S0010-4485(00)00103-2

[20] Pottmann, H.; Leopoldseder, S.: A concept of parametric surface fitting that avoids the parametrization problem, Computer Aided Geometric Design, 20(6), 2003,343-362. https://doi.org/10.1016/S0167-8396(03)00078-5

[21] Rabinovich, M.; Hoffmann, T.; Sorkine-Hornung, O.: Discrete Geodesic Nets for Modeling Developable Surfaces, ACM Transactions on Graphics (ToG), 37(2), 2017. https://doi.org/10.1145/3180494

[22] Sawhney, R.; Crane, K.: Boundary First Flattening, ACM Transactions on Graphics (ToG), 37(1), 2017. https://doi.org/10.1145/3132705

[23] Tsuchie, S.: Reconstruction of underlying surfaces from scanned data using lines of curvature, Computers & Graphics, 68, 2017, 108-118. https://doi.org/10.1016/j.cag.2017.08.015
[24] Vaitkus, M.; Várady, T.: Parameterizing and extending trimmed regions for tensor-product surface fitting, Computer-Aided Design, 2017. https://doi.org/10.1016/j.cad.2017.11.008
[25] Várady, T.; Rockwood, A.; Salvi, P.: Transfinite surface interpolation over irregular n-sided domains, Computer-Aided Design, 43(11), 2011, 1330-1340. https://doi.org/10.1016/j.cad.2011.08.028
[26] Várady, T.; Salvi, P.; Karikó, G.: A Multi-sided Bézier Patch with a Simple Control Structure, Computer Graphics Forum, 35(2), 2016, 307-317. https://doi.org/10.1111/cgf.12833
[27] Weiss, V.; Andor, L.; Renner, G.; Várady, T.: Advanced surface fitting techniques, Computer Aided Geometric Design, 19(1), 2002, 19-42. https://doi.org/10.1016/S0167-8396(01)00086-3