Analytic Surface Detection in CAD Exported Models

Pavel Šigut[1], Petr Vaněček[1] and Libor Váša[2]
[1]Department of Computer Science and Engineering, University of West Bohemia, Technická 8, Plzeň, Czech Republic
[2]NTIS - New Technologies for the Information Society, University of West Bohemia, Technická 8, Plzeň, Czech Republic

Keywords: Analytic Surface, Fitting, Curvature, Estimation, Analysis, CAD.

Abstract: 3D models exported from CAD systems have certain specifics, that influence their subsequent processing. Typically, in contrast with scanned surface meshes, vertices of exported meshes lie almost exactly on analytic surfaces used in CAD modeling. On the other hand, the triangulation of exported models is usually dictated by the requirement of having the lowest possible number of primitives, which results in highly uneven sampling density and common appearance of extremely large and small triangle inner angles. For applications such as classification, categorization, automatic labeling or similarity based retrieval, it is often necessary to identify significant features of an exported model, such as planar, cylindrical, spherical or conical regions, and their properties. While this type of information is naturally available in the original CAD system, it is only rarely exported together with the surface model. In this paper, we discuss two means of identifying analytic regions in triangle meshes, taking into account the specifics of CAD-exported models, and provide a quantitative comparison of their performance.

1 INTRODUCTION

In certain applications, large sets of 3D models are exported from a CAD system in the form of triangle meshes and further processed in this format. Commonly, a very basic file format, such as .STL, is intentionally chosen in order to maximize compatibility. Apart from being rather inefficient at representing the mesh connectivity, such format does not include any additional information about the original CAD part. The subsequent processing, which might include classification, categorization, automatic labeling or similarity based retrieval, can greatly benefit from availability of information such as number, locations and radii of cylindrical holes in the model and similar high level properties. However, such information is lost during the export, and for reasons of compatibility with multiple CAD tools, it is generally not an option to define a custom export format that includes such data.

In particular, common CAD models, such as technical workpieces or models of buildings, generally consist of patches defined by simple geometric primitives, such as planes, cylinders, spheres, cones, less often also tori and more general surfaces. These patches are tessellated during the export, and thus turned into piecewise planar surfaces. Our specific objective is to identify these patches, together with the parameters of the original geometric primitives. For each resulting primitive, a set of triangles is to be found that represent it in the model. For CAD models, it is reasonable to expect that the vertices of the triangles assigned to a particular patch lie almost immediately on a surface defined by the primitive, only allowing for a numerical imprecision.

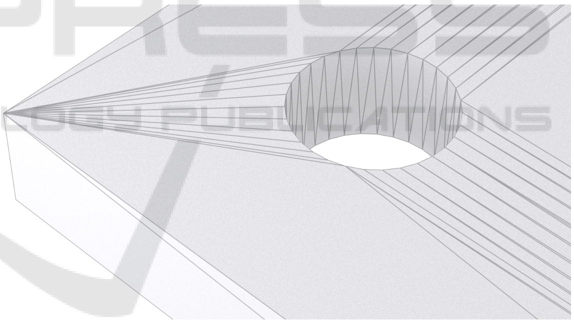

Figure 1: Example of a typical model exported from a CAD system. Notice the extremely small inner angles.

A similar problem has been addressed by many works concerning analysis, completion and refinement of scanned triangle meshes. We argue that although these problems are related, there are circumstances that make them substantially different. In particular, scanned 3D meshes usually have a relatively uniform sampling that is dictated by the resolution of

Šigut, P., Vaněček, P. and Váša, L.
Analytic Surface Detection in CAD Exported Models.
DOI: 10.5220/0007396702780285
In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2019)*, pages 278-285
ISBN: 978-989-758-354-4

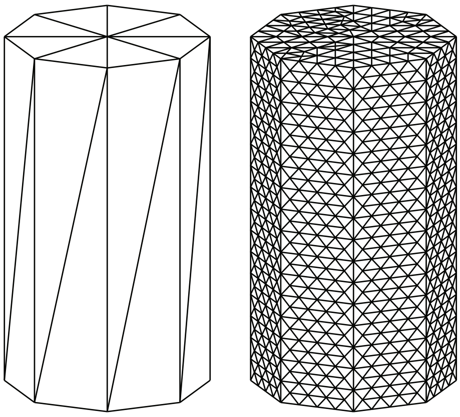

Figure 2: Cylinders are often represented by a low number of sides (left). However, subdividing them in order to improve triangle regularity is not possible, because the resulting shape (right) is rather interpreted as a flat sided prism than a cylinder.

the sampling device. A CAD exported model, on the other hand, usually has a tessellation that is minimal with respect to the original shape. Therefore there are many elongated triangles, and even important parts are often represented by only a *very few* triangles.

Resampling a CAD exported model regularly, although possible, is generally not a viable approach. A CAD-exported model usually has all vertices exactly (up to numerical error) lying on the analytic surfaces used in the design process. Any resampling always introduces additional vertices, whose position cannot be determined so that they also lie on the actual surface of the model (for illustration see Figure 2). On the other hand, the vertex positions in scanned data always contain noise that must be taken into account, which is an issue that rarely occurs with CAD exports.

We have experimented with two fundamentally different approaches to the problem at hand. First is based on estimating *principal curvatures* for each mesh vertex, which in turn allows us to classify vertices as either planar, cylindrical and possibly spherical. Next, continuous patches of vertices that lie on a common plane, cylinder or sphere are created using a region growing technique. Finally, the equation of each segment is refined.

The curvature based approach depends strongly on the quality of curvature estimation, which is in turn dictated by the sampling quality that is typically rather poor with CAD exported models. Therefore we have experimented with another, more direct approach to the problem at hand. Basic primitives, such as planes, cylinders, spheres and cones are *fitted* to local patches of triangles. If a good fit is found, the segment is then grown by traversing the surface. The parameters of the representing primitives are refined using a gradient descent based approach.

After reviewing related work in the following section, sections 3.1 and 3.2 will discuss both approaches in more detail. Note that although some of the design choices in the algorithm description may seem arbitrary, we have performed extensive testing of a plethora of variations, trying to achieve best possible results in experiments that will be described in section 4. The algorithm flow and parameters described here result from these experiments and provide the best results we were able to achieve by either method. Finally, section 5 will compare the two approaches and section 6 will conclude the paper.

2 RELATED WORK

Previous work mostly addresses the problem of finding analytic surfaces in dense scanned meshes, mostly in the context of reverse engineering (Várady et al., 1997)(Benkö et al., 2001)(Benkö et al., 2002). A bottom-up approach fitting planes, spheres and cylinders has been proposed by (Attene et al., 2006). Primitives are fitted to small patches first, and the patches are merged in a greedy fashion in order to obtain larger consistent areas.

An approach based on sampling the surface randomly and searching for fitting planes, cylinders, cones, spheres and tori has been proposed by (Schnabel et al., 2007). This approach has been later used for filling in missing parts of scanned man made objects (Schnabel et al., 2009).

Another approach based on extracting and optimizing boundary curves on the primitive shapes has been proposed by (Jenke et al., 2008). The method is primarily targeted at scanned indoor/outdoor scenes with noise and outliers.

Fitted surfaces were also used by (Li et al., 2011) in order to discover global relations, such as symmetries and parallel planes. This kind of global relations help adjusting the parameters of the fitted surfaces and improve the reconstruction quality.

Identifying local proxies, i.e. planar or other analytical surfaces, is also a key ingredient in Variational Shape Approximation(VSA) (Cohen-Steiner et al., 2004) and its more advanced variants (Wu and Kobbelt, 2005). While the original VSA algorithm only uses planes as proxies, the Hybrid Variational Surface Approximation also allows using spheres and cylinders as well.

The primitive fitting based approach we describe is closely related to algorithms discussed in the literature. On the other hand, using curvatures for identifying analytic patches in triangle meshes has to the best of our knowledge not been thoroughly studied before.

3 ALGORITHMS

3.1 Curvature based Detection

The curvature based detection algorithm consists of following steps:

1. **Preprocessing.** Mesh is cut along crease edges and extremely long triangles are subdivided into smaller ones.
2. **Curvature Estimation.** Principal curvatures are estimated for each vertex of the preprocessed mesh.
3. **Vertex Classification.** Each vertex is marked as either planar, cylindrical or other based on the curvatures.
4. **Patch Creation.** Continuous patches of vertices of the same type are identified.
5. **Primitive Fitting.** Parameters of the corresponding primitives are estimated for each patch.
6. **Vertex-triangle Mapping.** The primitives represented as vertex sets are transformed into sets of triangles of the input mesh (i.e. before subdivision).
7. **Postprocessing.** Large primitives with a good fit are grown to include possible neighbouring triangles that also fit their equation. Small patches are removed.

Each of these steps will be discussed in the following paragraphs in more detail.

3.1.1 Preprocessing

First, any non-manifold edges and vertices of the mesh are removed. Next, the continuous mesh is split along edges where the dihedral angle is larger than 42°. This threshold is chosen experimentally, based on the knowledge that regular 8-sided prisms (45° dihedral angles) commonly appear in CAD models, and their sides therefore should be separated, while higher side count usually indicates the presence of a cylinder, where the mantle should remain continuous.

As a second step, elongated triangles are subdivided. A threshold is determined based on the median edge length, and on each edge longer than the threshold new vertices are generated. Finally, all triangles incident with subdivided edges are split into smaller triangles filling the area of the original triangle. This step is necessary for the following curvature estimation to succeed, however, as argued previously, global remeshing of the surface is not a viable option. In the experiments we have performed, the above described process provided better results than both working with original geometry and full resampling.

3.1.2 Curvature Estimation

Many approaches to estimating curvature on sampled surface have been proposed in the literature (for a recent review, see (Váša et al., 2016)). We have tested multiple choices, obtaining best results using the generalized shape operator approach by Hildebrandt et al. (Hildebrandt and Polthier, 2011). In particular, we have used the $\hat{S}$ operator with radius equal to $1.6 \times a$, where a is the average edge length in the triangle mesh. This method allows us to estimate both principal curvatures at each mesh vertex with reasonable accuracy and robustness.

3.1.3 Vertex Classification

Next, each vertex is classified as either planar, cylindrical or unknown. A vertex is classified as planar, when both its curvatures are smaller than 5% of the average curvature over all vertices. Otherwise, a vertex may be classified as cylindrical, if the ratio of the magnitudes of the estimated principal curvatures is larger than 10:1. Finally, if this condition is not met either, the vertex is classified as unknown. Note that it would be fairly easy to classify vertices as spherical by principal curvature equality, however, we did not use this classification, since spherical segments are rather rare in the data we have experimented with. In contrast with that, detecting a conic vertex is generally not possible by looking only at a single vertex and its estimated principal curvatures.

3.1.4 Patch Creation

Having a vertex classification, continuous surface patches are generated in the next step. From each vertex that is not classified as unknown, a breadth first search is started using the mesh connectivity. Vertices are added to the patch if they are of the same type, and their estimated curvature is not too different from that of the initial vertex. In order to evaluate the similarity, we relate the difference in curvature to the average curvature over all vertices of the mesh. If the magnitude of the difference is larger than 10% of the average curvature in either of the principal curvatures, then the candidate vertex is deemed too different and is not added to the growing patch. Otherwise, it is conquered and marked as such, so that it is neither later used in any other patch nor as initial vertex for another search.

3.1.5 Primitive Fitting

For each patch, a corresponding primitive is fitted. Plane fitting is done in the least squares sense. Cylinders, on the other hand, are fitted by a two stage process. First, a unit direction is found that has the smallest dot product with all vertex normals in the patch, by solving an eigenvalue problem resulting from the constrained minimization. This direction is the likely orientation of the axis of the cylinder. Next, all vertices are projected along this direction, forming a set of 2D points, which are fitted by a circle. We start with a set of equations in the form

$$(x_i - x_c)^2 + (y_i - y_c)^2 = r^2, \tag{1}$$

where x_i, y_i are the projected coordinates of i-th point, x_c, y_c are the unknown coordinates of the center and r is the unknown circle radius. Expanding the equation yields

$$x_i^2 - 2x_i x_c + x_c^2 + y_i^2 - 2y_i y_c + y_c^2 = r^2. \tag{2}$$

Subtracting an arbitrary j-th equation from all others results in a set of *linear* equations of form

$$x_i^2 - 2x_i x_c + y_i^2 - 2y_i y_c - x_j^2 + 2x_j x_c - y_j^2 + 2y_j y_c = 0, \tag{3}$$

which is solved in the least squares sense. The resulting center point corresponds to the cylinder axis, and the cylinder radius is computed as the average distance of all vertices in the segment to this axis.

3.1.6 Vertex-triangle Mapping

So far, patches of vertices are identified, our goal is, however, to obtain triangle patches. In the mapping step, a conversion is performed. If a vertex is assigned to a patch, all of its incident triangles are marked as claimed by the given patch. These triangles may be the result of triangle splitting performed in preprocessing, for each such triangle we keep the index of the original triangle it comes from, and this triangle is then claimed by the patch. This way, each original triangle may be claimed three times by the same patch, or possibly by several different patches different number of times. Conflicts are resolved based on the number of claims: the patch that claims a triangle most times gets it assigned. This way, some minor patches disappear completely.

3.1.7 Postprocessing

Since some vertices may fit into multiple patches, some triangles may be assigned to a wrong patch at this point, especially those on the border between patches. As a final step, patches are sorted in order of descending area, and grown. For each patch, neighbouring triangles are considered, and the tip vertex of each neighbouring triangle is tested for fit to the given patch. If it is close enough to the analytic surface represented by the patch, it is reassigned, unless it is already assigned to a larger patch. If a triangle is reassigned to a patch, its neighbours are considered for reassignment as well.

3.2 Gradient Descent

The above described method suffers from issues related to the key component of vertex curvature estimation. Therefore we have implemented a different approach as well, building on different concepts. In particular, we use analytic expression for squared distance from each fitted primitive. This allows us to fit primitives to vertex sets using a gradient descent approach. The algorithm works in following steps:

1. **Plane Fitting.** Parameters of a plane equation are computed for each local neighbourhood and planar regions are grown using a traversal technique.
2. **Cylinder Fitting.** Parameters of a cylinder equation are estimated for each local neighbourhood and refined using gradient descent. Additional triangles are added to the patch using a traversal technique.
3. **Cone Fitting.** Parameters of a cone equation are estimated for each local neighbourhood and refined using gradient descent. Additional triangles are added to the patch using a traversal technique.
4. **Judging.** Triangles that are part of multiple patches are assigned to a single patch using a decision algorithm.

Note that this approach does not use triangle subdivision as the curvature based algorithm did, since it is not necessary. On the other hand, splitting edges with large dihedral angle is still performed.

3.2.1 Plane Fitting

Each triangle of the triangle mesh defines a plane. During plane fitting, plane equation parameters are computed for each triangle that is not yet assigned to a plane patch. Next, using breadth first search, neighbouring triangles are tested whether or not they fit into the equation, and fitting triangles are added to the growing patch. Patches consisting of a single triangle are discarded, all others are preserved for the final judging.

3.2.2 Cylinder Fitting

Next, a similar process is done for fitting cylinders. Tetriamonds, i.e. quadruples of neighbouring triangles, are generated starting from triangles that are not assigned to any cylinder patch yet. For each tetriamond, an initial cylinder is found using the same procedure as described in section 3.1.5, only this time using triangle normals and triangle centroids instead of vertices and their normals to estimate cylinder orientation, since the triangle normals are more representative of the surface. Next, we build an analytic equation for squared distance from the cylinder surface. Partial derivatives of this equation are computed with respect to all cylinder parameters. Using the derivatives, gradient descent is used to optimize the parameters of the cylinder so that it fits the vertices of the tetriamond as closely as possible. If the final fit is tight enough, the cylinder patch is preserved and additional triangles are assigned to it using breadth first search.

3.2.3 Cone Fitting

Cones are fitted in a way similar to cylinder fitting. Normals of points lying on a cone all lie on a plane that is orthogonal to the orientation of the cylinder (see Fig. 3). Once again, we use triangle normals and triangle centroids of a tetriamond consisting of triangles not yet assigned to any cone. Fitting a plane to the normals in least squares sense allows us to estimate the direction of a potential cone. Average angle between the normals and the estimated cone direction allows us to estimate the cone angle, and combined with the cone direction it is possible to estimate the cone apex position.

Next, we improve the cone equation in the same way cylinder equation has been improved. Since distance to a cone is not as straightforward as distance to a cylinder, we describe the used formula in more detail. Squared distance d of a point $\mathbf{x}$ from a cone with apex $\mathbf{o}$, unit axis direction $\mathbf{v}$ and tip angle α (see figure 4) can be expressed as

$$d(\mathbf{x},\mathbf{o},\mathbf{v},\alpha) = sin^2(\beta-\alpha) * (\mathbf{x}-\mathbf{o}) \cdot (\mathbf{x}-\mathbf{o}), \quad (4)$$

where

$$cos\beta = \frac{(\mathbf{x}-\mathbf{o}) \cdot \mathbf{v}}{\|\mathbf{x}-\mathbf{o}\|}. \quad (5)$$

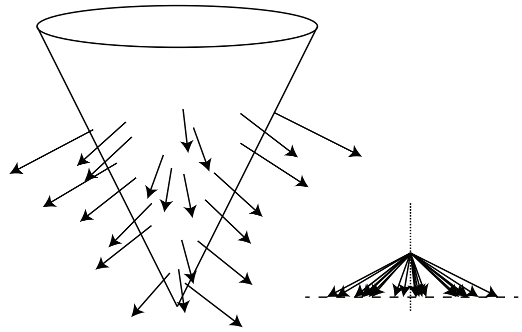

Figure 3: A cone with a few random surface points and their normals (left). When the normals are brought to a common origin (right), they all lie in a plane (dashed) that is orthogonal to the cone axis direction (dotted).

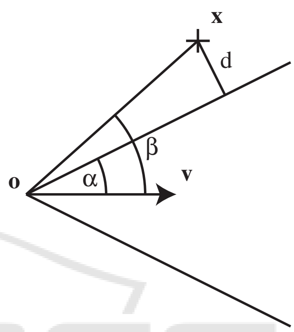

Figure 4: Variables in the distance from cone computation.

For the purposes of efficient computation, the first term in equation 4 can be written as

$$sin^2(\beta-\alpha) = (sin\beta cos\alpha - cos\beta sin\alpha)^2, \quad (6)$$

where $cos\beta$ can be obtained from equation 5 and $sin\beta = (1-cos^2\beta)^{\frac{1}{2}}$. Note that this expression is correct for both points inside and outside of the cone.

Equation 4 can be differentiated with respect to all cone parameters, which allows us to use gradient descent in order to reduce the squared distances of vertices from the cone. Cones that fit tetriamond vertices are grown again by a breadth first search. In contrast with cylinders, whenever we reach a vertex that does *not fit* the cone, we refine the cone parameters using the gradient descent procedure once more, taking into account all previously added vertices. This is necessary, since the cone fitting is more prone to getting trapped in a local minimum. Once the parameters are refined, the previously unfit vertex is tested again and if it fits, it is added to the patch and the search procedure continues.

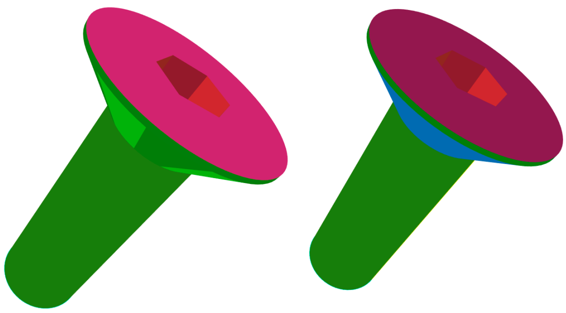

Figure 5: A typical result of classification. Different planes are marked in different shades of red, cylinders in green, cones in blue. Curvature is unable to find the conic surface and fits several cylinders instead (left), while gradient correctly fits a single cone (right).

3.2.4 Postprocessing

At this point, each triangle can be claimed by up to three patches: a plane, a cylinder and a cone. A decision algorithm assigns each triangle to a single patch, selecting the largest patch in terms of total number of claimed triangles. After this step, each triangle is assigned to a single patch. Finally, assignment is refined by the same procedure as described in section 3.1.7, i.e. patches are processed from largest to smallest and possible reassignments of neighboring vertices are considered and performed.

4 EXPERIMENTS

In order to test the performance of the two described methods (abbreviated as *curvature* and *gradient*), we have designed an experiment that compares the results with manually tagged data. A person that was otherwise not involved with the development of either method was given a set of exported CAD models. They were requested to select groups of triangles on each model and mark them as plane, cylinder or cone. On each model, multiple groups could be selected, and each group should always contain triangles that belong to a single analytic surface. They were not required to mark all the triangles of each model, and in areas where the classification was uncertain they were specifically instructed not to perform marking.

In total, 95 models were marked, ranging from 12 to 80 000 triangles. In total, there were 81597 out of 225068 triangles marked in 1919 groups, out of which 1136 were plane groups, 636 were cylinder groups and 147 were cone groups.

This data allows us to evaluate several accuracy criteria. First, for each marked triangle, we can check whether or not each method has assigned it to the correct primitive type. Second, for each group, we can check whether all triangles belonging to the group were also assigned to a single patch by each method, and no classified patch falls into more than one marked group.

Since the principle of the curvature based method makes it unable to detect cones, although they appear quite commonly in our data, we must decide how to handle their evaluation and avoid bias towards either method (see figure 5). There are two approaches possible, both of which we have tested:

1. for *curvature*, consider all triangles that were marked as cone and classified as anything but *unknown* as an error (denoted *cones on*).
2. turn off cone classification in *gradient* and ignore all triangles marked as cone (denoted *cones off*).

Table 1: Percentages of incorrectly classified triangles. *Average* values are listed with corresponding *standard deviations* in parentheses.

method	**cones on**	**cones off**
curvature	9.06% (15.22%)	3.44% (10.52%)
gradient	4.67% (12.90%)	2.85% (11.61%)

5 RESULTS

Table 1 lists the percentages of *incorrectly* classified triangles for both methods. It shows that both methods present a viable approach to the problem at hand. The gradient based method provides slightly better results. The considerably large values of standard deviation indicate that with most models most triangles were classified correctly (in fact, median of all percentages in the table is 0.0%), while there are a few problematic cases where large error percentages occur.

Most interestingly, the Pearson correlation coefficient of the error percentages of the curvature/gradient based methods is -0.025 for the *cones on* scenario and -0.058 for the *cones off* scenario. This indicates, that rather than some models being more difficult than others, it is more likely that different models cause problems with either method (see figure 6). A potential hybrid method that combines the strengths of both discussed approaches could therefore achieve a considerable improvement of classification accuracy.

Sometimes, it might happen that a small area of a model is densely sampled, and if it gets incorrectly classified, then it may skew the overall results. In table 2 we list the percentages of marked areas that were classified incorrectly. Interestingly, in this crite-

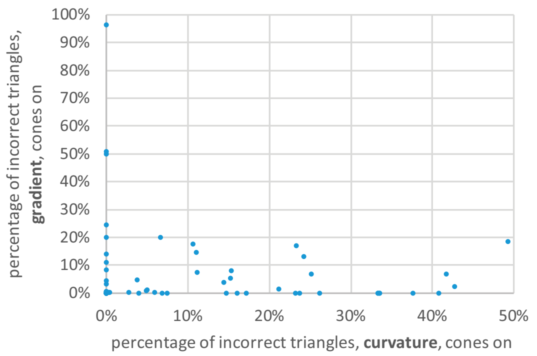

Figure 6: Correlation of the percentage of incorrectly classified triangles by both methods.

Table 2: Percentages of incorrectly classified surface areas. *Average* values are listed with corresponding *standard deviations* in parentheses.

method	cones on	cones off
curvature	3.29% (9.60%)	1.66% (7.61%)
gradient	4.63% (15.63%)	4.01% (15.61%)

rion, the curvature based approach provides better results regardless of whether or not cone classification is considered.

Additionally, we can also evaluate the consistence of whole marked groups rather than individual triangles. For each marked group, we check two fail cases:

1. Triangles of the group are part of two or more classified patches.
2. Another marked group contains triangles classified to the same patch.

Note that the two cases are not equivalent: if a classifier puts all cylinder triangles into a single patch, then the first case does not occur, since for each marked group, all of its triangles fall into a single classified patch. If either of these cases occurs, we mark the group as incorrectly identified. Table 3 lists the percentages of incorrectly identified groups for both tested methods in both tested scenarios. Curvature provides here substantially better performance than gradient in terms of both average values and standard deviation.

Finally, we have also measured the processing speed of both methods. Both were implemented in C# and executed on a common PC with Intel Core i7 920 @2.67GHz processor and 12GB RAM memory. Table 4 lists the mean and median times needed per triangle. It shows that the processing times are all of roughly the same order of magnitude. The curvature based method is faster on average than the gradient based method with cone fitting turned on, note however, that this is caused by a few extreme cases when cone fit is recomputed many times. In more than one half of the cases, gradient based method is in fact faster than the curvature based method, as documented by the median. Finally, when cone fitting is turned off, the gradient based method is considerably faster than the two alternatives.

Table 3: Percentages of incorrectly identified groups. *Average* values are listed with corresponding *standard deviations* in parentheses.

method	cones on	cones off
curvature	8.45% (8.54%)	4.49% (4.54%)
gradient	10.43% (18.21%)	9.16% (17.40%)

Table 4: Per triangle processing times in milliseconds.

method	mean	st. dev.	median
curvature	0.44	0.46	0.34
gradient	1.21	6.64	0.28
gradient cones off	0.086	0.13	0.048

6 CONCLUSIONS

We have presented two approaches to identification of analytic surfaces in CAD exported triangle meshes. Both were thoroughly tested and tuned in order to provide the best result in an extensive experiment. One approach is based on the commonly used gradient descent paradigm, while the other uses a less common principle of analyzing estimated principal curvatures.

The experiment results show that both methods work roughly on par, with slight inclinations towards either depending on how exactly one specifies the target criterion. Most interestingly, the correlation of the results of the two methods is very weak, indicating a certain complementarity. A hybrid method combining the strengths of both approaches could therefore provide a considerable improvement of the performance. Design and implementation of such method is, however, difficult, because deciding which of the methods is more accurate in each case is a task that is in certain sense equivalent to the overall problem itself. Therefore we leave this question open for future research.

ACKNOWLEDGEMENTS

The authors would like to thank Jakub Vašta for carefully marking models for our research. This work was partially supported by Ministry of Education, Youth and Sports of the Czech Republic, project PUNTIS (LO1506) under the program NPU I, and University specific research project SGS-2016-013 Advanced Graphical and Computing Systems.

REFERENCES

Attene, M., Falcidieno, B., and Spagnuolo, M. (2006). Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22(3):181–193.

Benkö, P., Kós, G., Várady, T., Andor, L., and Martin, R. R. (2002). Constrained fitting in reverse engineering. *Computer Aided Geometric Design*, 19:173–205.

Benkö, P., Martin, R. R., and Várady, T. (2001). Algorithms for reverse engineering boundary representation models. *Computer-Aided Design*, 33:839–851.

Cohen-Steiner, D., Alliez, P., and Desbrun, M. (2004). Variational shape approximation. *ACM Trans. Graph.*, 23(3):905–914.

Hildebrandt, K. and Polthier, K. (2011). Generalized shape operators on polyhedral surfaces. *Comput. Aided Geom. Des.*, 28(5):321–343.

Jenke, P., Krückeberg, B., and Straßer, W. (2008). Surface reconstruction from fitted shape primitives. In Deussen, O., Keim, D. A., and Saupe, D., editors, *VMV*, pages 31–40. Aka GmbH.

Li, Y., Wu, X., Chrysathou, Y., Sharf, A., Cohen-Or, D., and Mitra, N. J. (2011). Globfit: Consistently fitting primitives by discovering global relations. *ACM Trans. Graph.*, 30(4):52:1–52:12.

Schnabel, R., Degener, P., and Klein, R. (2009). Completion and reconstruction with primitive shapes. *Computer Graphics Forum (Proc. of Eurographics)*, 28(2):503–512.

Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226.

Várady, T., Martin, R. R., and Cox, J. (1997). Reverse engineering of geometric models - an introduction. *Computer-Aided Design*, 29(4):255–268.

Váša, L., Vaněček, P., Prantl, M., Skorkovská, V., Martínek, P., and Kolingerová, I. (2016). Mesh statistics for robust curvature estimation. *Computer Graphics Forum*, 35(5):271–280.

Wu, J. and Kobbelt, L. (2005). Structure recovery via hybrid variational surface approximation. *Comput. Graph. Forum*, 24(3):277–284.