Estimating Approximate Plane of Symmetry of 3D Triangle Meshes

Lukáš Hruda*
Jan Dvořák†
Supervised by: Ivana Kolingerová‡

Department of Computer Science and Engineering
University of West Bohemia
Pilsen / Czech Republic

Abstract

In many natural or artificial objects some form of reflectional symmetry can be found. For most objects there is no exact symmetry but still the object may look symmetrical to a person. In such cases an approximate plane of symmetry can be estimated. The plane of symmetry can be used for example in cases when we need to align a given object with coordinate axes. This paper presents a new simple algorithm which can be used to estimate the plane of symmetry of a 3D object represented by a triangle mesh. Unlike other algorithms proposed in this field, our algorithm does not use any advanced numerical calculations, is fairly easy to implement and configure.

Keywords: Plane of Symmetry, Triangle Mesh, 3D

1 Introduction

Symmetry information can be useful, e.g., in geometric modeling for object alignment or in computer vision for object recognition. This paper deals with reflection symmetry, specifically with finding a plane of symmetry of a given three dimensional object represented by a triangle mesh. The main motivation for our work was finding a symmetry plane of 3D scans of real human faces. This symmetry information was then intended to be used in anthropology. It is almost impossible for an exact reflection symmetry to exist in a 3D object but there is something very close to a symmetry in many real world objects, natural or artificial. If we look at such objects, we usually see the symmetry in them, even when it is only approximate. The human face is actually a good example, because there can never be an exact symmetry in it, but we usually see the symmetry between eyes, ears and between the whole right and left side of the face. When the object is represented by a triangle mesh and has no exact reflection symmetry in it, i.e. no plane of symmetry exists, we can still find its very good approximation. Our goal was to design an algorithm which could find a 'decent' approximation of a symmetry plane of a given 3D object represented by a triangle mesh, possibly the one a person would expect after looking at the object. Figure 1 shows an example of a 3D object where a person would expect the vertical symmetry plane (marked red).

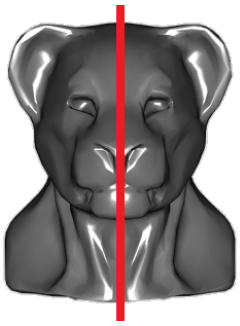


Figure 1: Symmetrical object represented by a triangle mesh

Figure 2 shows the same object with a missing part. Even in such a case a person would most likely still expect the symmetry plane the same as in Figure 1.

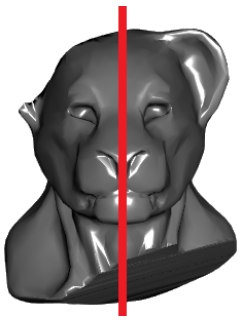


Figure 2: Symmetrical object with missing parts

There are many methods for symmetry detection but most of them use complicated calculations and are difficult to implement, sometimes even to understand with-

*hrudalu@students.zcu.cz
†jdvorak@students.zcu.cz
‡kolinger@kiv.zcu.cz

out rather deep mathematical knowledge. We propose an algorithm for the estimation of an approximate symmetry plane which gives good results, often even for objects where some part is missing. The proposed algorithm is quite simple, fairly easy to understand and implement even by someone new in the triangle mesh processing and differential geometry. The proposed solution was derived as a simplification of [11]. The advantage of our method is that it does not require calculation of the Laplace-Bertrami operator's eigenvalues and eigenvectors to find the significant points, instead we use quite a simple Gaussian curvature estimation. Our method also works well when used on a triangle mesh with a low number of vertices (thousands).

The structure of the paper is as folows. Section 2 describes related work, Section 3 explains the proposed method. Section 4 presents results and Section 5 concludes the paper.

2 Related Work

In the past, there has been a lot of research in estimation of many types of symmetry. The reason is that many of algorithms in computer graphics and computer vision can benefit from the knowledge about the symmetricity of an object.

Let us concentrate on the papers which focus on detection of imperfect or approximate symmetry of objects. There are some common approaches. The first approach uses some symmetry measurements. Zabrodsky et al. [14] used Symmetry Distance - a measure of how much different is the given object from a perfectly symmetric object. However, it required the processing of all points of the object to evaluate the function. Podolak et al. [10] suggested a Monte Carlo algorithm for sampling points which were used in the measurement. Martinet et al. [6] described Generalized moment functions. These functions have the same symmetries as the processed object plus a small number of extra candidate symmetries that can be filtered. The main disadvantage of all the methods described above is that they expect the plane of symmetry going through the center of mass of the object, which is not always true in the case of an object with missing parts.

Another frequently used approach is to employ voting. It is based on the assumption that the better the approximation of the plane of symmetry is, the more points their symmetric counterpart has. This approach was firstly described by Mitra et al. [8]. They use a clustering of symmetry transformations for determining partial symmetries of an object. The symmetry transformation candidates are generated by matching points with high Gaussian curvature with points that have similar principal curvatures. Another algorithm based on voting was introduced by Lipman et al. [5] - they search for correspondences in points and store them in a symmetry correspondence matrix. The best symmetric transformation is then found by spectral methods.

All methods mentioned so far were designed for objects without missing parts. However, many applications require working with incomplete data. For example Sipiran et al. [11] use symmetry to repair or complete the missing parts of scanned cultural heritage objects. The voting approach mentioned in the previous paragraph can be improved to work with incomplete data. Xu et al. [13] proposed a method which is similar to the [10], but only points that fulfill a certain criterion can contribute to (or we can say vote for) the given measure. Jiang et al. [4] use a curve skeleton to find the symmetry in the point cloud. Their method is similar to [5], however, only points of the skeleton are considered for creating the symmetry correspondence matrix. Most recently, Sipiran et al. [11] described an algorithm that selects points as local maxima of heat diffusion signatures and then generates candidate symmetry planes between each pair of points. All the pairs then vote for the best symmetry plane. This method is very robust to missing parts and noise in data, however, it is really complex.

For further information about all types of symmetry, other algorithms for its estimation and its use in computer science see [9].

3 Proposed Algorithm

Our algorithm works in two phases. During the first phase a given number of significant points are extracted from the mesh. In the second phase a voting process is deployed to find a pair of points which generates the best plane of symmetry. This is a similar approach to the one used in [11], but to find the significant points we use a quite simple Gaussian curvature estimation and the voting process is simplified too.

3.1 Extraction of Significant Points

Let P be an optional parameter giving the required number of significant points. As the significant points, P vertices with the highest Gaussian curvature are taken. Gaussian curvature is a term from differential geometry and is defined in a point on a continuous surface. Since a triangle mesh is not a continuous but discrete representation of a surface, there is no way how to calculate the exact Gaussian curvature in a vertex of a triangle mesh. It can only be estimated. There is more than one way to estimate the Gaussian curvature in a vertex of a 3D triangle mesh. The one we used in this algorithm is described as follows [7].

$$G(\mathbf{x}_i) = \frac{1}{A_i}(2\pi - \sum_{j=1}^{n} \varphi_j)$$

$G(\mathbf{x}_i)$ is the Gaussian curvature in a vertex $\mathbf{x}_i$, n is the total number of vertices neighboring to $\mathbf{x}_i$. The meaning of the angle φ_j is depicted in Figure 3. The value A_i is the surface area which belongs to the vertex $\mathbf{x}_i$ and it can be calculated

as one third of the sum of areas of the triangles incident to the vertex $\mathbf{x}_i$. We used a little different way to calculate the area. Specifically we calculate it as the sum of areas of Voronoi regions incident to $\mathbf{x}_i$, more details about this calculation can be found in [7], but it is not very complex.

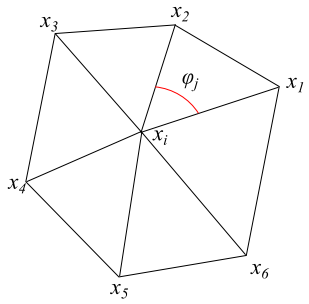


Figure 3: Vertex $\mathbf{x}_i$ and its neighborhood with angle φ_j shown

3.2 Selecting Candidate Planes

Let X denote the set of vertices chosen in the first phase. In the second phase only vertices in X are processed. In the next step we choose all pairs of vertices $\{\mathbf{x}_i, \mathbf{x}_j\}, \mathbf{x}_i, \mathbf{x}_j \in X$, $i \neq j$ which satisfy the following two criteria:

Curvature similarity criterion

$$S \leq \frac{G(\mathbf{x}_i)}{G(\mathbf{x}_j)} \leq \frac{1}{S},\ S \in \langle 0,1 \rangle,$$

where S is an optional parameter, which denotes how similar the Gaussian curvatures in the two vertices must be for this criterion to be satisfied. The higher the S value is the more similar they must be.

Normal angle criterion

$$|cos(\alpha_{ij}^{ij})| \geq C_{minNorm},\ C_{minNorm} \in \langle 0,1 \rangle,$$

where α_{ij}^{ij} is the angle between vectors $(\mathbf{x}_i - \mathbf{x}_j)$ and $(\mathbf{n}_i - \mathbf{n}_j)$, where $\mathbf{n}_i, \mathbf{n}_j$ are unit normal vectors in vertices $\mathbf{x}_i, \mathbf{x}_j$, respectively, and $C_{minNorm}$ is an optional parameter which denotes how large the angle α_{ij}^{ij} can be for this criterion to be satisfied.

Then a set R of candidate planes ρ_{ij} for all pairs of points chosen in the previous step is created. The plane ρ_{ij} is the plane of symmetry of the vertices $\mathbf{x}_i$ and $\mathbf{x}_j$. It is defined by the implicit equation $a_{ij}x + b_{ij}y + c_{ij}z + d_{ij} = 0$, where $\mathbf{n}_{ij} = [a_{ij}, b_{ij}, c_{ij}]^T$ is the plane normal vector calculated as $\mathbf{n}_{ij} = \mathbf{x}_i - \mathbf{x}_j$. The d_{ij} coefficient is calculated by substituting $\frac{\mathbf{x}_i + \mathbf{x}_j}{2}$ for $[x, y, z]^T$ in the equation.

3.3 Voting

The next step is to choose a plane from R which best approximates the symmetry plane of the mesh. In order to do that we start a simple voting process. For each plane $\rho_{ij} \in R$ we iterate through all pairs of points $\{\mathbf{x}_k, \mathbf{x}_l\}, \mathbf{x}_k, \mathbf{x}_l \in X, k \neq l$. The plane ρ_{ij} gets a vote from the pair $\{\mathbf{x}_k, \mathbf{x}_l\}$ if and only if the next four criteria are satisfied:

Curvature similarity criterion

$$S \leq \frac{G(\mathbf{x}_k)}{G(\mathbf{x}_l)} \leq \frac{1}{S},\ S \in \langle 0,1 \rangle$$

Angle criterion

$$|cos(\beta_{kl}^{ij})| \geq C_{min},\ C_{min} \in \langle 0,1 \rangle,$$

where β_{kl}^{ij} is the angle between the vector $(\mathbf{x}_k - \mathbf{x}_l)$ and the normal vector $\mathbf{n}_{ij}$ of the plane ρ_{ij} (which is the vector $(\mathbf{x}_i - \mathbf{x}_j)$) and C_{min} is an optional parameter which denotes how large the angle β_{kl}^{ij} can be for this criterion to be satisfied.

Normal angle criterion

$$|cos(\alpha_{kl}^{ij})| \geq C_{minNorm},\ C_{minNorm} \in \langle 0,1 \rangle,$$

where α_{kl}^{ij} is the angle between the vector $(\mathbf{n}_k - \mathbf{n}_l)$ and the normal vector $\mathbf{n}_{ij}$ of the plane ρ_{ij} (which is the vector $(\mathbf{x}_i - \mathbf{x}_j)$), where $\mathbf{n}_k, \mathbf{n}_l$ are unit normal vectors in the vertices $\mathbf{x}_k, \mathbf{x}_l$, respectively.

Distance criterion

$$dist(\frac{\mathbf{x}_k + \mathbf{x}_l}{2}, \rho_{ij}) \leq D_{max},$$

where $dist(\mathbf{x}, \rho)$ is the distance of point $\mathbf{x}$ from the plane ρ and $D_{max} = D_{diag} \cdot D_{maxRel}$, where D_{diag} is the length of the diagonal of the triangle mesh bounding box and $D_{maxRel} \geq 0$ is an optional parameter which denotes how far (relatively to the length of the bounding box diagonal) the middle point of the two vertices can be from the plane in order to satisfy this criterion.

You can see that the **curvature similarity criterion** and the **normal angle criterion** are the same as for the candidate plane selection. In the end the plane which was given the highest number of votes is declared the resulting approximate symmetry plane.

3.4 Averaging

If there are more planes with the same highest number of votes, we can simply choose one at random or we can average them all together. Before the averaging is done, we have to ensure that all the planes have the same orientation. Let R_{win} denote the set of winning planes, i.e. the

set of planes with the highest number of votes. It is obvious that $R_{win} \subseteq R$, in most cases $R_{win} \subset R$. We choose any default plane $\rho_0 \in R_{win}$ and for all the other planes $\rho_g \in R_{win}$, $g \neq 0$, we do the following: if $\mathbf{n}_0^T \mathbf{n}_g < 0$, then we switch the direction of $\mathbf{n}_g$ by multiplying it by -1, if $\mathbf{n}_0^T \mathbf{n}_g \geq 0$, then nothing is done, the vectors $\mathbf{n}_0, \mathbf{n}_g$ are the normal vectors of the planes ρ_0, ρ_g respectively.

The averaging itself is done by averaging the a, b, c, d coefficients of the given planes after their normalization (dividing the plane equation by the length of the plane normal vector). Let R'_{win} denote the set of winning planes after the orientation adjustment done in the previous step and let $r(\rho_g) = [a_g, b_g, c_g, d_g]^T$ denote the vector of coefficients of the plane $\rho_g \in R'_{win}$. The vector $r(\rho_{avrg})$ of coefficients of the averaged plane ρ_{avrg} is calculated as follows.

$$r(\rho_{avrg}) = \frac{1}{|R'_{win}|} \sum_g \frac{1}{||\mathbf{n}_g||} r(\rho_g),\ \rho_g \in R'_{win},$$

where $\mathbf{n}_g$ is the normal vector of plane ρ_g. After this calculation ρ_{avrg} is declared the resulting approximate symmetry plane.

4 Experimental Results

During the implementation and testing of our method we have set the default configuration of the algorithm parameters as follows: $P = 200$, $S = 0.5$, $C_{min} = 0.985$, $C_{minNorm} = 0.985$, $D_{maxRel} = 0.01$. All of the results shown in this section were accomplished with this exact configuration unless stated otherwise. We have also used the winning plane averaging method described in Section 3.4 in all our experiments, but let us note that in most cases there was only one winning plane.

We have tested our algorithm on several objects. Most importantly we used four triangle meshes created by scanning real human faces [2]. Figure 4 shows the four face scans and the approximate symmetry planes which our algorithm estimated for them. The planes are shown as dark rectangles in the images. You can see that the estimated planes really capture the reflectional symmetry in the faces, even when the faces clearly are not perfectly symmetrical.

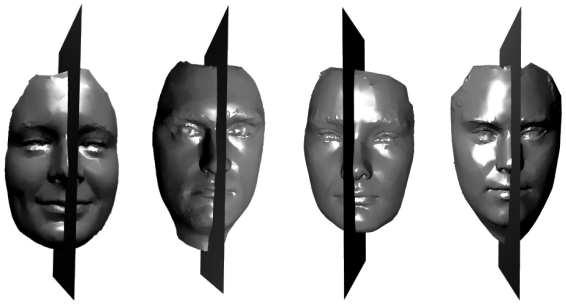


Figure 4: Four scans of real human faces with the estimated symmetry planes shown

We also used 3D models of faces artificially generated by *FaceGen 3D Print* software [3]. The software also provides control over how the face will be deformed. The result of the symmetry estimation for those models is shown in Figures 5 and 6. It can be clearly seen that the proposed algorithm has no problems with finding what seems to be a correct symmetry plane even for faces with higher level of deformation.

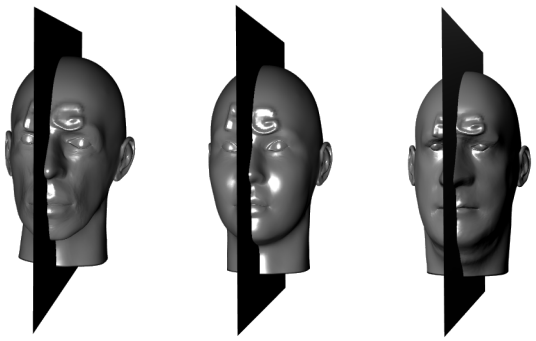


Figure 5: Artificially created faces with the estimated symmetry planes shown

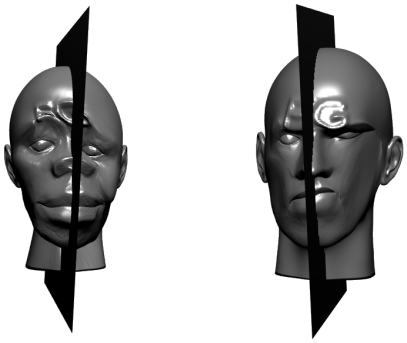
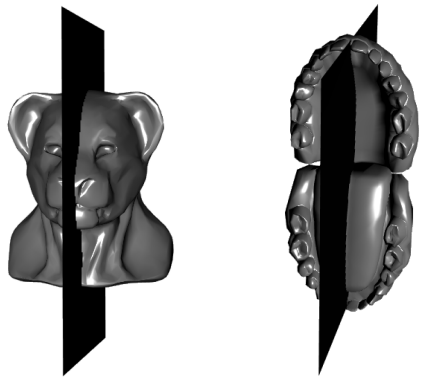


Figure 6: Symmetry estimation for artificially created faces with higher level of deformation

The algorithm also works well with different objects than faces. For example, for the lion head shown in Figure 1 it estimated the plane of symmetry exactly where it was expected (see Figure 7a). Another object for which the algorithm can find a perfectly correct plane of symmetry is the Teeth model from *Microsoft 3D Builder* sample model library [1] (see Figure 7b). Such perfect results were expected as both models are perfectly symmetrical.

4.1 Models with Missing Parts

The algorithm was also tested on models with missing parts. One of the face scans and two artificially generated faces were clipped using *Microsoft 3D Builder* software. Applied on those models, the algorithm still performs well and can still find what seems to be a good approximation of the plane of symmetry (see Figure 8).



(a) (b)

Figure 7: Example of symmetry estimation of perfectly symmetrical objects (a) Head of lion, (b) Teeth

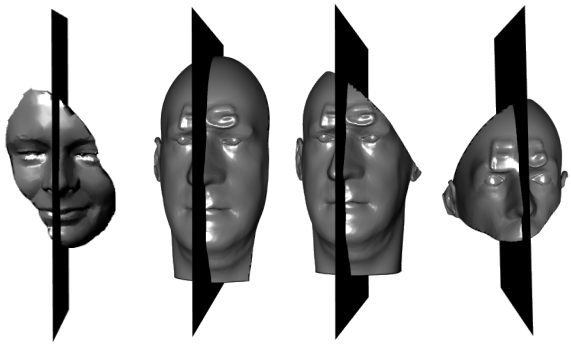


Figure 8: Symmetry estimation of face models with missing parts

Another model with missing parts was created by clipping the head of the lion. The expected plane of symmetry can be seen in Figure 2. The plane estimated by our algorithm is shown in Figure 9.

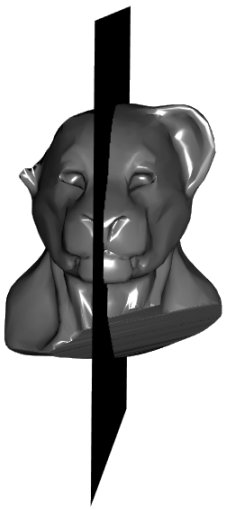
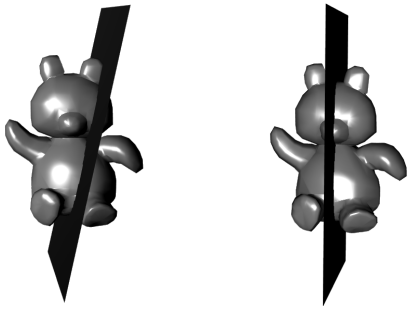


Figure 9: Symmetry estimation of the model with missing parts

4.2 Influence of Parameter Configuration

The default parameter configuration described at the beginning of this section is not universally optimal. There are many triangle meshes for which a different configuration ensures much better results than the default one. Let us consider the triangle mesh depicted in Figure 10. Figure 10a shows the estimated symmetry plane found by our algorithm with the default configuration.

You can see that the plane does not capture the symmetry very well. The reason for this might be the fact that, although a person probably perceives some symmetry, the object is actually quite far from symmetrical, i.e. there is little symmetry between the arms and between the legs. This means that for this particular triangle mesh the default configuration might be too constraining (i.e. the voting criteria are too difficult to satisfy).



(a) (b)

Figure 10: Teddy bear and its symmetry plane estimated by our algorithm with: (a) the default parameter configuration, (b) the adjusted parameter configuration where $D_{maxRel} = 0.02$

If we just change the D_{maxRel} parameter from 0.01 to 0.02, which makes the configuration less constraining, we can get much better approximation of the teddy bear's symmetry plane. The symmetry plane which was estimated by our algorithm with this adjusted configuration is depicted in Figure 10b.

You can see that by adjusting the algorithm parameters to correspond with the object's properties we can achieve results which are much better than results achieved using the default configuration. The less symmetrical the given object is the less strict symmetry plane we are looking for and the less constraining the parameters should be.

4.3 Influence of Vertex Count

During our experiments we have discovered that our algorithm works best when used on triangle meshes with a low number of vertices, specifically when the number of vertices is in the order of thousands. Figure 11 shows a triangle mesh [12] with 117535 vertices and the symmetry plane which our algorithm estimated for it. You can see that this result is very far from the plane of symmetry.

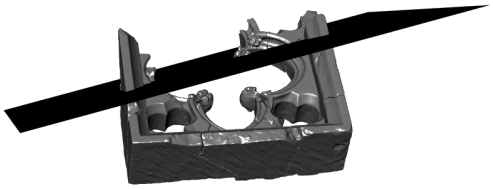


Figure 11: Triangle mesh with 117535 vertices and its symmetry plane estimated by our algorithm

We have used *Microsoft 3D Builder* [1] to simplify the mesh and by this we have lowered the number of its vertices to 15652. The simplified triangle mesh and its plane of symmetry estimated by our algorithm are depicted in Figure 12. It is obvious that this result is much better than the previous one, but it certainly cannot be considered a good result.

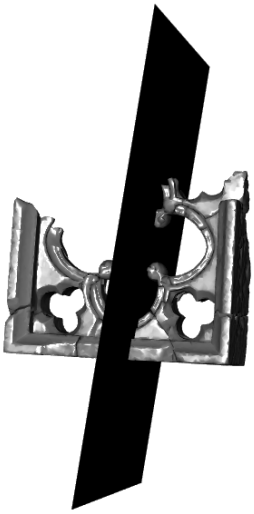


Figure 12: Simplified triangle mesh with 15652 vertices and its symmetry plane estimated by our algorithm

The number of vertices of the simplified mesh is still quite large for our algorithm when it is using the default parameter configuration. It would be very convenient to adjust the configuration to correspond with this fact and we can do that by increasing the number of extracted significant points. So we change the value of parameter P from 200 to 500. With higher number of significant points we can also afford to make the other parameters more constraining, so we change S from 0.5 to 0.8 and we also change C_{min} and $C_{minNorm}$ from 0.985 to 0.99. After these changes in the configuration our algorithm estimates a considerably good approximation of the symmetry plane depicted in Figure 13.

It can be seen that by adjusting the parameter configuration we were again able to improve the results quality, but the greatest impact on the quality had the simplification of the triangle mesh which lowered the number of its vertices from 117535 to 15652.

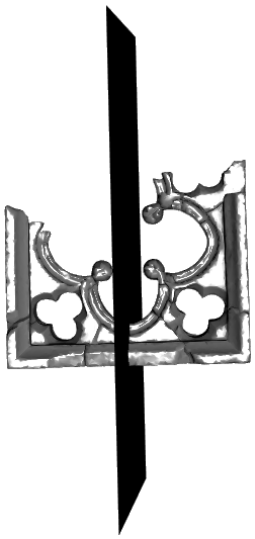


Figure 13: Simplified triangle mesh with 15652 vertices and its symmetry plane estimated by our algorithm with the adjusted parameter configuration ($P = 500, S = 0.8, C_{min} = C_{minNorm} = 0.99$)

4.4 Limitations of our Algorithm

There are triangle meshes for which our algorithm does not work very well, such as meshes where a person sees a symmetry but there actually is none. For example let us consider the triangle mesh depicted in Figure 14. The figure also shows the symmetry plane estimated by our algorithm. You can see that there is some form of reflectional symmetry in the object's image but the estimated plane does not capture it very well (the plane should be more centered and less tilted to the right). The reason why this triangle mesh is problematic is that although its 2D projection looks very symmetrical when observed from certain points of view, the 3D object itself actually is quite asymmetrical.

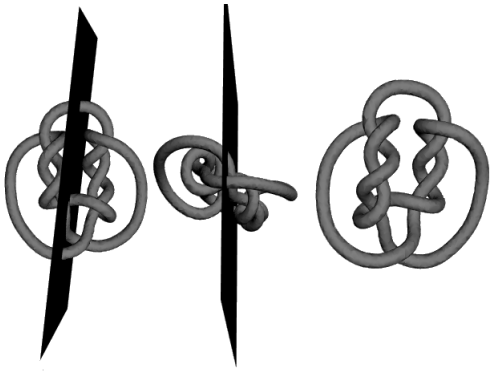


Figure 14: Triangle mesh representing a knot and its symmetry plane estimated by our algorithm

Another limitation is the vertex count. Our algorithm works very poorly when used on meshes with large number of vertices (tens of thousands or more) but this prob-

lem can be resolved with mesh simplification (see Section 4.3). Our method also probably would not work very well with triangle meshes representing rough surfaces or meshes with high level of detail where the Gaussian curvature is high in all vertices. This is probably also one of the reasons why it works poorly with meshes with large vertex count, because such meshes are usually quite detailed. This problem could most likely be resolved with using some more advanced method for Gaussian curvature estimation, but the simplicity of our algorithm would suffer.

5 Conclusion

We have proposed a new algorithm for approximate symmetry plane estimation for 3D triangle meshes which was developed as a significant simplification of [11] and we have shown that it still works considerably well, even when used on meshes with missing parts. We have simplified both the significant point selection and the voting process.

Our algorithm is very well configurable, it has a few optional parameters and the result quality often depends on the configuration of these parameters. Although we have set a default configuration which works well with many objects, there are also many other objects for which the default configuration does not work that well. By setting the configuration to correspond with the particular mesh properties the algorithm can be made to give much better results. In the future we would like to find a way to estimate the optimal values of the parameters automatically for a given triangle mesh.

Our algorithm also has some limitations, specifically it works very poorly with triangle meshes which have a large number of vertices. On the other side it works well with meshes with very low number of vertices (thousands) which can be considered an advantage, because in our opinion, simplifying a triangle mesh to lower its vertex count is much less problematic than subdividing it to make the vertex count larger.

Acknowledgements. This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic, the project SGS-2016-013 Advanced Graphical and Computing Systems. We would like to thank to Prof. J. Sochor from the Masaryk University in Brno, Czech Republic for inspiring this work and supplying us the data for the experiments.

References

[1] Microsoft 3d builder. `https://www.microsoft.com/cs-cz/store/p/3d-builder/9wzdncrfj3t6`. Accessed: 2017-02-03.

[2] Fidentis project. `https://www.fidentis.cz`, 2012.

[3] Singular Inversions. Facegen 3d print 1.9[computer software]. `https://facegen.com/3dprint_demo.htm`.

[4] Wei Jiang, Kai Xu, Zhi-Quan Cheng, and Hao Zhang. Skeleton-based intrinsic symmetry detection on point clouds. *Graphical Models*, 75(4):177–188, 2013.

[5] Yaron Lipman, Xiaobai Chen, Ingrid Daubechies, and Thomas Funkhouser. Symmetry factored embedding and distance. In *ACM Transactions on Graphics (TOG)*, volume 29, page 103. ACM, 2010.

[6] Aurélien Martinet, Cyril Soler, Nicolas Holzschuch, and François X Sillion. Accurate detection of symmetries in 3d shapes. *ACM Transactions on Graphics (TOG)*, 25(2):439–464, 2006.

[7] Mark Meyer, Mathieu Desbrun, Peter Schröder, Alan H Barr, et al. Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and mathematics*, 3(2):52–58, 2002.

[8] Niloy J Mitra, Leonidas J Guibas, and Mark Pauly. Partial and approximate symmetry detection for 3d geometry. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 560–568. ACM, 2006.

[9] Niloy J Mitra, Mark Pauly, Michael Wand, and Duygu Ceylan. Symmetry in 3d geometry: Extraction and applications. In *Computer Graphics Forum*, volume 32, pages 1–23. Wiley Online Library, 2013.

[10] Joshua Podolak, Philip Shilane, Aleksey Golovinskiy, Szymon Rusinkiewicz, and Thomas Funkhouser. A planar-reflective symmetry transform for 3d shapes. *ACM Transactions on Graphics (TOG)*, 25(3):549–559, 2006.

[11] Ivan Sipiran, Robert Gregor, and Tobias Schreck. Approximate symmetry detection in partial 3d meshes. In *Computer Graphics Forum*, volume 33, pages 131–140. Wiley Online Library, 2014.

[12] Theoharis T. and G. Papaioannou. PRESIOUS 3d cultural heritage fragments, 2013.

[13] Kai Xu, Hao Zhang, Andrea Tagliasacchi, Ligang Liu, Guo Li, Min Meng, and Yueshan Xiong. Partial intrinsic reflectional symmetry of 3d shapes. In *ACM Transactions on Graphics (TOG)*, volume 28, page 138. ACM, 2009.

[14] Hagit Zabrodsky, Shmuel Peleg, and David Avnir. Symmetry as a continuous feature. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1154–1166, 1995.