University of West Bohemia
Faculty of Applied Sciences
Department of Computer Science and Engineering

Master Thesis

Symmetry Detection in Geometric Models

Plzeň 2018 Bc. Lukáš Hruda

Místo této strany bude zadání práce.

Declaration

I hereby declare that this master thesis is completely my own work and that I used only the cited sources.

Plzeň, 9th May 2018

Bc. Lukáš Hruda

Acknowledgement

I would like to thank Prof. Dr. Ing. Ivana Kolingerová for supervising my thesis, providing valuable council and for enabling me to work on this topic in the first place. I also have to thank Doc. Ing. Libor Váša, Ph.D. for providing me the *Mesh Processing framework* which was used in this work for data loading and processing and mainly for visualization of results. At last, I would like to thank my colleague Bc. Jan Dvořák for starting the whole symmetry detection research with me because, although he did not work with me on this thesis, I would probably have never pursued this topic without him.

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic, project SGS 2016-013 Advanced Graphical and Computing Systems and institutional research support (1311).

Abstract

Reflectional symmetry is a potentially very useful feature which many real world objects exhibit. Its knowledge can be used in variety of applications such as object alignment, compression, symmetrical editing or reconstruction of incomplete objects. To acquire the symmetry information usable in such applications, often a robust symmetry detection algorithm needs to be used since most objects are not perfectly symmetrical and exhibit only approximate symmetry. In this thesis a new method for detecting the plane of reflectional symmetry for 3D objects is proposed which works on perfectly as well as approximately symmetrical objects. Furthermore, the proposed method works on point clouds and therefore puts virtually no constraints on the input data.

Abstrakt

Zrcadlová symetrie je vlastnost, která se vyskytuje u mnoha reálných objektů. Její znalost může být velice užitečná v mnoha aplikacích, jako zarovnání objektů, komprese, symetrická editace nebo rekonstrukce neúplných objektů. K získání znalosti o symetrii, která je použitelná v podobných aplikacích, je často zapotřebí použití robustního algoritmu pro detekci symetrie, jelikož mnoho objektů nevykazuje perfektní symetrii, ale pouze přibližnou. V tomto textu bude popsána nová metoda pro detekci roviny zrcadlové symetrie pro 3D objekty, která je použitelná jak pro perfektně, tak pro přibližně symetrické objekty. Tato metoda navíc funguje na objektech reprezentovaných pouze množinou bodů, a tudíž neklade prakticky žádné požadavky na vstupní data.

Contents

1 Introduction

Many real world objects exhibit some kind of symmetry. There are many types of symmetry such as reflectional symmetry with respect to a plane, to a line or to a point, rotational symmetry or some more general symmetry. In this thesis we will mainly address the reflectional symmetry of a 3D object with respect to a plane. A 3D object X is reflectionally symmetrical with respect to a plane P if the object X stays the same when it is reflected over the plane P. In such a case we can call P the plane of symmetry, or the symmetry plane, of the object X. Some other types of symmetry will be briefly discussed in Chapter 5.

In some cases only parts of an object are symmetrical but the whole object is not, we call such symmetries local symmetries. When the whole object is symmetrical then we call such a symmetry a global symmetry. In this thesis we will only consider global symmetry, with the exception of a few of the existing symmetry detection methods described in Chapter 2 that can be used to detect local symmetries.

In most cases the reflectional symmetry is not perfect but only approximate, for example, see Figure 1.1a which depicts a real 3D-scanned human face. We can see that the face is symmetrical with respect to a plane that passes between the eyes and through the nose and the mouth (such a plane is also shown in the figure), but since it is a real human face it is certainly not perfectly symmetrical. Finding the plane that captures such approximate symmetry automatically is often not a simple task.

In computer graphics, the information about reflectional symmetry in 3D objects has various applications, such as object alignment [17], compression [20], symmetrical editing [14] or reconstruction of incomplete objects [24] [21] [19]. The last application is especially interesting because it requires the symmetry detection to work on incomplete objects, i.e. objects with some missing parts, such as the clipped face depicted in Figure 1.1b. Even though the face is incomplete, there is still some symmetry remaining in it which a human observer can see.

(a) 3D-scanned face (b) Incomplete 3D-scanned face

Figure 1.1: 3D-scanned human face - (a) and its incomplete version damaged by clipping - (b) with planes that capture their approximate reflectional symmetry.

Finding such a symmetry in objects with missing parts seems to be the major challenge in the field of symmetry detection. Therefore, the main goal of this work was to design a method that would be capable of detecting the plane of symmetry (global) of perfectly as well as approximately symmetrical 3D objects and possibly even objects with some missing parts.

In the following text, we will describe several existing methods usable for symmetry plane detection on 3D objects and discuss their advantages and disadvantages. After that, we will describe a new global symmetry plane detection method that was designed as part of this work and we will show its results. In the end, we will propose several ways how this method could be extended or generalized for detecting symmetries of different types.

2 Related Work

This chapter provides a brief description and subjective evaluation of number of available methods that can be used for detecting a plane of reflectional symmetry of 3D objects. Each section in this chapter corresponds with a single publication in which one or more symmetry plane detection methods are presented. The sections are named after the authors of the corresponding publications and also contain the publication year by which the sections are sorted. The publication titles and other information can be found in the bibliography at the end of the thesis.

All the methods described in this chapter can be divided into two groups according to whether they can be used on point clouds or whether they require a surface to work on. The methods described in Sections 2.3, 2.7, 2.8, 2.9, 2.10 and 2.15 seem to be usable on point clouds, meaning they do not require any information on the input other than the point positions. In addition, the method described in Section 2.2 seems to work with various object representations, point clouds included. All the other methods are designed to work on surfaces and all of them seem to be usable on triangle meshes, which is probably the most common surface representation. An exception is the method described in section 2.5 which works not just with a surface but also with volumetric data.

Not all the described methods were specifically designed for detecting a plane of reflectional symmetry, some are more general and were designed for detecting more types of symmetry, but all of them can, in some way, be used for symmetry plane detection or can be modified to do so. At the end of this chapter a brief summary and overall subjective evaluation of the described methods are provided.

2.1 Zabrodsky et al. (1995)

This article [28] presents a symmetry measure called Symmetry Distance (SD) and its uses in detection of reflectional or rotational symmetry. The measure is originally designed for 2D shapes represented by a sequence of points (basically polygons) and can be used for symmetry detection in 2D images (grayscale bitmap pictures) which are first transformed into contours and the contours are then sampled to create the sequences of points. The Symmetry Distance is a quantifier of the minimum effort required to transform a given shape into a symmetric shape and is based on squared distances

between the points of the two shapes. The authors also present a way to use this approach to detect reflectional symmetry in 3D shapes represented by a set of points. For a given candidate plane a perpendicular plane is created and sampled and each sampled point is projected onto the 3D object. For every projected point its elevation is calculated relatively to the sampled plane and the symmetry value of the candidate plane is evaluated using the projected sampling points. As candidate planes, only planes that pass through the object are taken.

The authors only show the result of the reflectional symmetry detection on one 3D object (see Figure 2.1) which seems to be perfectly or almost perfectly symmetrical. Furthermore the method requires the 3D object to be represented by a surface onto which points can be projected which excludes point clouds. Also since the symmetry measure uses squared distances between points, it would probably have problems with objects that exhibit missing parts or have outliers.

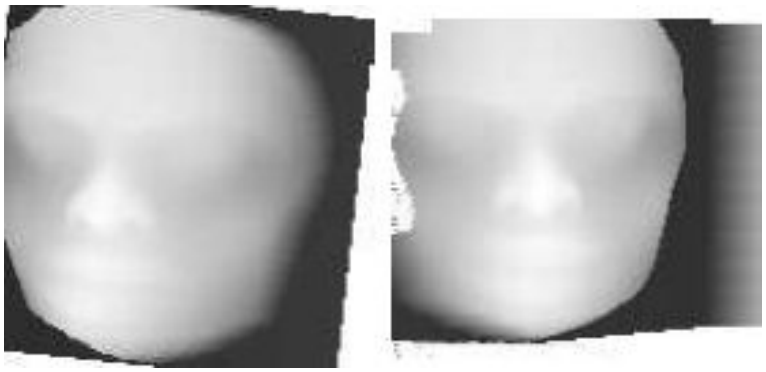


Figure 2.1: Result of the symmetry plane detection using the method by Zabrodsky et al. The symmetry plane was detected on the original object (left) and used for alignment. Specifically the object was rotated to a frontal vertical view (right). The symmetry plane itself is not shown in the figure. (Figure taken from [28].)

2.2 Sun, Sherrah (1997)

This article [23] presents a method which can be used to detect reflectional and rotational symmetry of 3D shapes. The method uses the discrete version of the extended Gaussian image called orientation histogram which can be obtained by dividing a unit sphere into hexagonal bins with values assigned according to the number of normal vectors facing in the given bin's direction (see Figure 2.2). Detection of the reflectional symmetry is then done by

choosing a few candidate planes passing through the center of the orientation histogram's sphere. For each candidate plane the histogram is reflected over the given plane and a correlation of the reflected histogram with the original one is calculated. The plane with the highest correlation is then declared the plane with the strongest reflectional symmetry.

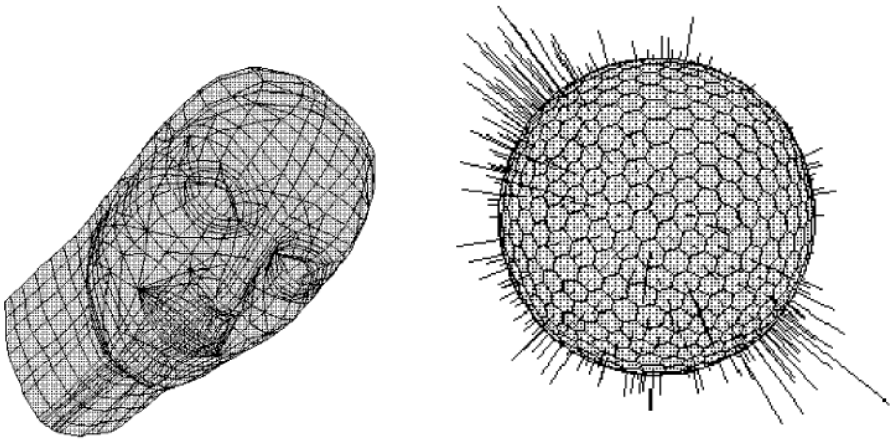


Figure 2.2: Model of a human head (left) and its orientation histogram (right). (Figure taken from [23].)

The authors state they only choose candidate planes with normal vectors facing in directions of the three principle axes and 5 or 6 of their neighbors which makes 8 or 9 candidate planes in total. This, together with the fact that the method expects the symmetry plane to pass through the center of the orientation histogram's sphere, implies that this method probably would not work very well with shapes that have some missing parts. Also the authors mostly show the results of their method on very simple objects which seem to be perfectly or nearly perfectly symmetrical (see Figure 2.3). An advantage of this approach is that it can be used to detect the rotational symmetry as well but the specific algorithm is a little different. Also it seems to work with various object representations such as a surface representation (e.g. triangle mesh), 2D range image, volumetric representation or a point cloud.

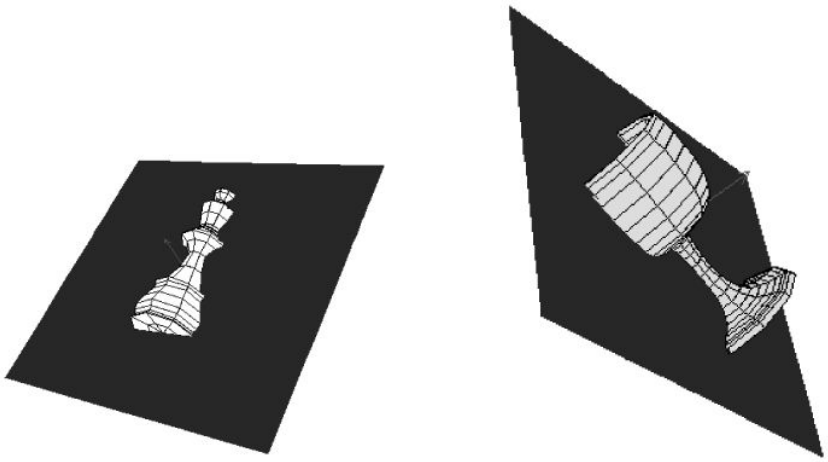


Figure 2.3: Results of the symmetry plane detection using the method by Sun and Sherrah. (Figure taken from [23].)

2.3 Thrun, Wegbreit (2005)

This paper [24] presents a method for detecting local symmetries of various types in partial point clouds and their use for completing partial 3D objects. Apart from the point cloud representing the object the method also needs to know the position from which the points were scanned to determine what places of the space are occluded. The authors propose a probabilistic model to score symmetries and they use this model to detect the correct symmetry type and the symmetry parameters. The algorithm consists of three nested loops where the outer loop searches for appropriate symmetry types, the middle loop identifies which points are actually symmetric with respect to the given symmetry type and the inner loop determines the parameters of the final symmetry.

Judging by the results shown in the paper the algorithm seems to work only on quite simple 3D objects, such as a ball, torus or a box. An advantage of this method is that it works on point clouds representing partial objects and it seems to be able to detect the symmetries even on point clouds where most of the original object is missing (see Figure 2.4). The authors also show how these objects are reconstructed using the symmetry information. A disadvantage of this method is the fact that it needs to know the scanner position.

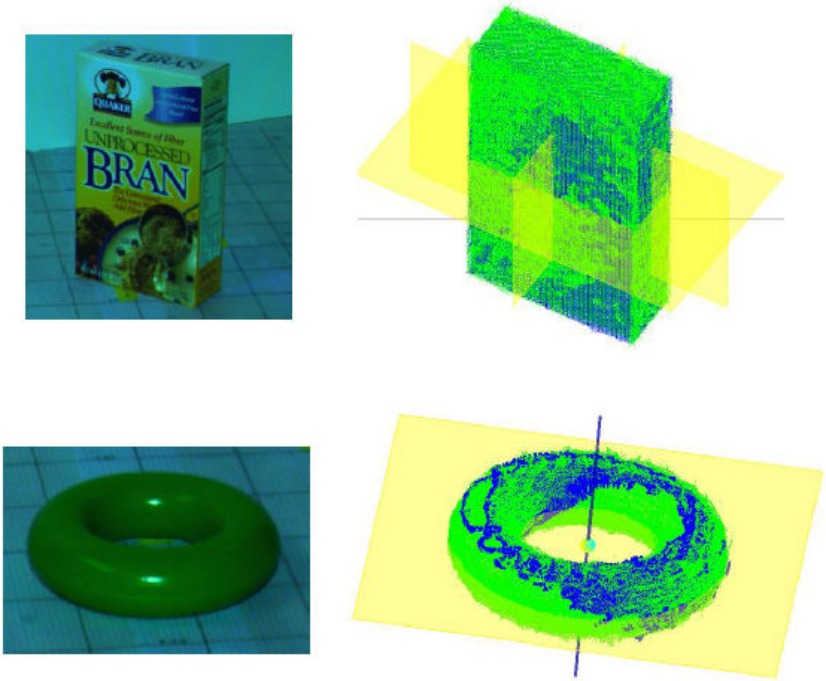


Figure 2.4: Results of the symmetry detection and object reconstruction using the method by Thrun and Wegbreit. Blue points are the original points and green points are the reconstructed points. (Figure taken from [24].)

2.4 Simari et al. (2006)

This paper [20] presents an algorithm for detecting local reflectional symmetries of 3D triangle meshes but seems to be usable for global reflectional symmetry detection as well. The authors propose a distance function which for a vertex $\boldsymbol{v}_i$ gives a distance of this vertex reflected over a given plane from the triangle mesh, this distance is denoted d_i. They use weighted covariance matrix $\boldsymbol{C}$ calculated as follows:

$$\boldsymbol{C} = \frac{1}{s}\sum_{i=1}^{n} w_i(\boldsymbol{v}_i - \boldsymbol{m})(\boldsymbol{v}_i - \boldsymbol{m})^T$$

where n is the vertex count of the mesh, $\boldsymbol{v}_i$ is the i-th vertex of the mesh, $\boldsymbol{m}$ is the center of mass of the mesh, w_i is the weight of i-th vertex and s is the sum of all weights. The weights are calculated as

$$w_i = \frac{2\sigma}{(\sigma^2 + d_i^2)^2}$$

where σ is an optional parameter. Also for each vertex $\boldsymbol{v}_i$ its cost ρ_i is defined as

$$\rho_i = \frac{d_i^2}{\sigma^2 + d_i^2}.$$

The weights are actually derived from the costs so that

$$w_i = \frac{1}{d_i} \frac{\partial \rho_i}{\partial d_i}.$$

Eigenvectors of the matrix $\boldsymbol{C}$ are computed and used to create three planes so that these vectors are the normal vectors of the planes and the planes pass through $\boldsymbol{m}$. For these planes the sum of costs of all vertices is computed and only the plane with the lowest cost sum is kept. The authors define a support face as the mesh face whose all vertices $\boldsymbol{v}_i$ reflected over the given plane have $d_i \leq 2\sigma$. They also define a support region as the largest connected region of support faces. For the plane acquired in the previous step the support region is computed and weights of all vertices outside the support region are set to 0. These plane creation and region finding steps are iterated until convergence is achieved.

When the final support region is removed from the mesh the algorithm can be used on the remaining components to find more local symmetries. This way a tree structure can be created called a folding tree. The authors propose to use this structure for mesh compression.

Results of the symmetry plane detection are shown on several objects which mostly seem to be quite simple and perfectly or almost perfectly symmetrical. For these objects also the reconstructed meshes from their folding trees are shown (see Figure 2.5). Few results of the symmetry detection are also shown on objects which are not perfectly symmetrical but none of these objects seem to miss parts and they also mostly exhibit only local symmetries (see Figure 2.6). The biggest disadvantage of this algorithm is the fact that it only works on triangle meshes. Also, since the algorithm seems to be more designed for local symmetry detection, it would most likely fail to detect a global reflectional symmetry of an object with significant missing parts.

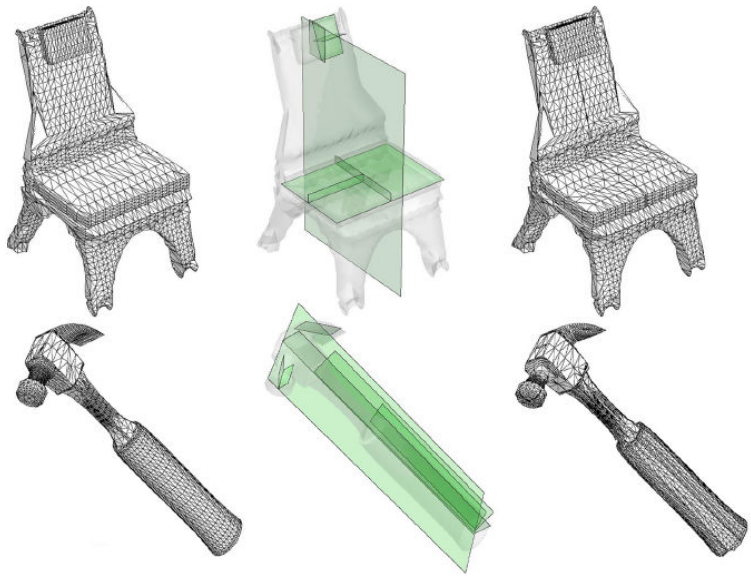


Figure 2.5: Results of the local symmetry plane detection using the method by Simari et al. The figure shows the original objects (left), their detected symmetry planes (center) and the objects reconstructed from their folding trees (right). (Figure taken from [20].)

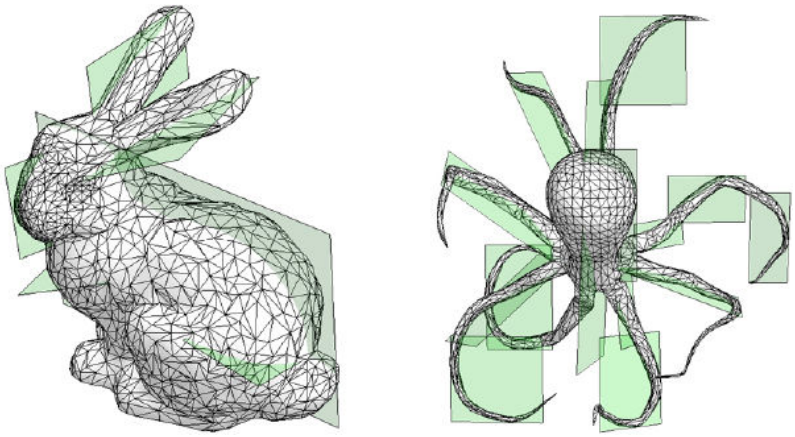


Figure 2.6: Results of the local symmetry plane detection using the method by Simari et al. on imperfectly symmetrical objects. (Figure taken from [20].)

2.5 Podolak et al. (2006)

This article [17] presents a method for detecting symmetry planes of volumetric functions. The authors define a *Planar Reflective Symmetry Transform* ($PRST$) which represents a measure of how symmetrical a given volumetric function is with respect to a given plane and they use its square ($PRST^2$)

to find the best symmetry plane. The brute force approach to find such a plane has time complexity $O(n^6)$ on the $n \times n \times n$ grid of voxels but the authors also propose a method to find the plane with the highest $PRST^2$ which seems to have time complexity $O(n^4 log(n))$.

In order to use this approach on 3D surfaces the authors propose using the Gaussian Euclidean Distance Transform to convert a surface to a volumetric function. Since such a function is usually very sparse, the authors propose a Monte Carlo algorithm which seems to be able to find its symmetry plane with time complexity only $O(n^4)$. The authors also propose an iterative approach to find local maxima of the $PRST$ to detect local symmetries.

Results of the symmetry detection on 3D surfaces are only shown on one 3D object which seems quite asymmetrical (see Figure 2.7). Specifically four strongest local symmetries are shown on it but none of them seems to capture the whole object's symmetry very well. Also it seems doubtful whether this method is able to detect a global reflectional symmetry of an object with significant missing parts. The authors also show how their method can be used for segmentation of 3D objects the results of which are shown and seem to be good.

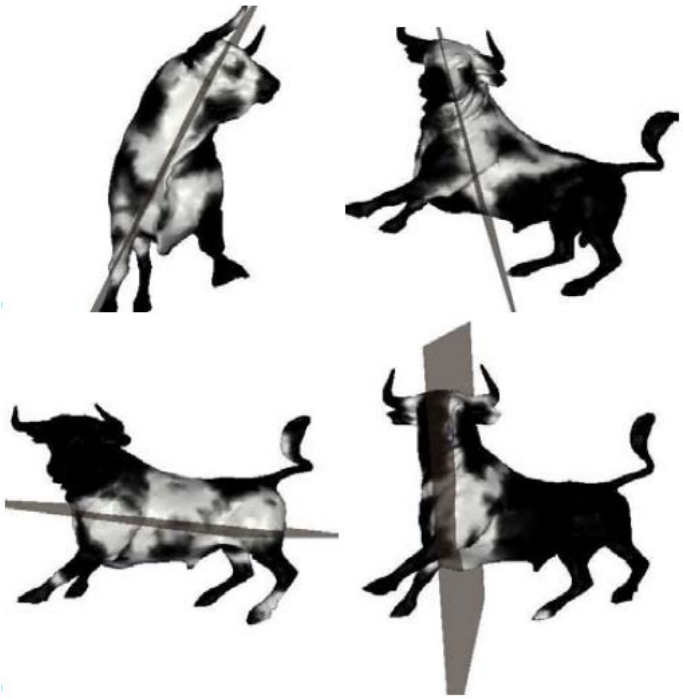


Figure 2.7: Several symmetry planes detected using the method by Podolak et al. on a single object. (Figure taken from [17].)

2.6 Martinet et al. (2006)

This article [14] presents a method for detection of rotational and reflectional symmetries of a 3D surface. The symmetry is defined by a symmetry axis, rotation angle α and a value λ which equals either 1 or -1 and determines whether the symmetry contains reflection. The authors propose functions called generalized moments of a shape and claim that these functions have the same symmetries as the shape itself. They also show that finding the symmetries comes down to determining where the generalized moments have zero gradient. Several candidate axes are created by refining a sphere from an icosahedron and the steepest descent minimization is performed on the norm of the moment function's gradient from these candidates. Finding the α and λ is done deterministically by examining the moment function's spherical harmonics.

Results of the symmetry detection are shown on objects which mostly exhibit quite strong symmetries (see Figure 2.8). The main disadvantage of this method is probably the fact that the detected symmetry axes must pass through the object's center of mass. This implies that it most likely would not work on objects exhibiting missing parts or imperfect symmetries. Also it does not work on point clouds since it needs a surface to work on.

Figure 2.8: Results of the symmetry plane detection using the method by Martinet et al. (Figure taken from [14].)

2.7 Mitra et al. (2006)

This paper [15] presents a method for detecting various types of local symmetries of 3D shapes. The detected symmetry transformations can contain reflection, rotation, translation and even scaling. In each sampled point of the input shape, principal curvatures and principle directions are computed. Pairs of the sampled points are then selected and used to create candidate

transformations by transforming one point of the given pair to align its position, its principal directions and its normal direction with the second point in the pair. The scale component of the transformation is estimated from the ratio of principle curvatures in the two points. The candidate transformations represent points in the transformation space. Clustering is performed on these points to detect the final symmetries.

The proposed method, as it is, does not serve the purpose of detecting a global plane of symmetry of a 3D shape but could very likely be modified to do so. The great advantage of this method is that it seems to be very general in the symmetry types it is able to detect.

2.8 Combès et al. (2008)

This paper [7] presents an ICP-like [6] iterative approach for finding a symmetry plane of a 3D point cloud. The first step is to choose an initial plane P. The second step is to reflect each point $\boldsymbol{x}_i$ in the point cloud over the plane P and for each reflected point $\boldsymbol{S}_P(\boldsymbol{x}_i)$ find the closest point $\boldsymbol{y}_i$ in the point cloud where the function $\boldsymbol{S}_P(\boldsymbol{x})$ reflects a point $\boldsymbol{x}$ over a plane P. The third step is to minimize the following function:

$$f(P) = \sum_{i=1}^{n} ||\boldsymbol{y}_i - \boldsymbol{S}_P(\boldsymbol{x}_i)||^2$$

where n is the number of points in the point cloud. The authors claim a closed form solution exists for the minimization problem. The fourth step is to determine whether the new plane P differs from the previous one and if so the algorithm returns to the second step.

As the authors themselves point out, this approach is not robust to outliers and often converges to only a local optimum. Therefore, the authors also propose a probabilistic approach to find the symmetry plane which is just a generalized version of the above described algorithm. In this approach the point cloud is considered to be noised with the noise being isotropic Gaussian with variance σ. Convergence of this method seems to depend hardly on the σ value so the authors use a multiscale scheme where they start with a large value of σ and they let it decrease gradually. Also they decimate the point cloud for larger values of σ and refine it progressively as σ decreases. In the end the authors propose a simple method for outlier rejection for the cases when the closest point to $\boldsymbol{S}_P(\boldsymbol{x}_i)$ is too far from it.

The authors show the results of their symmetry plane detection method on several objects containing scanned human faces, the Stanford bunny [5] and a chair with a missing leg (see Figure 2.9). None of the tested objects

seems to exhibit perfect symmetries and apart from the Stanford bunny all of them have some missing parts although not very significant. Overall, the results seem to be good. The greatest advantage of this method is probably the fact that it works on point clouds so no other information besides the point positions is needed. The biggest disadvantage is probably the fact that the result of the method seems to depend on quite many parameters (6 in total) but the authors proposed their default values. Whether these values are universally optimal is unclear. Also each step of this iterative method seems to have time complexity $O(n^2)$ which raises a question whether this method is computable for large point clouds (with hundreds of thousands of points or more).

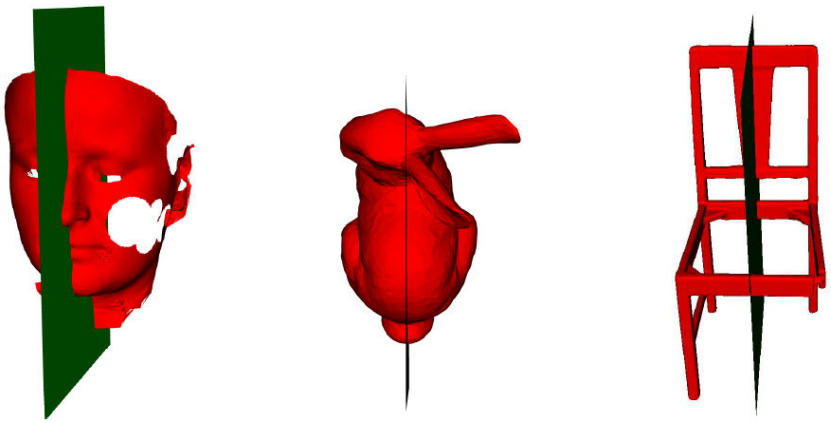


Figure 2.9: Results of the symmetry plane detection using the method by Combès et al. (Figure taken from [7].)

2.9 Lipman et al. (2010)

This article [13] presents a method for detecting various types of symmetries in 3D objects represented by sets of points. It uses symmetry correspondence matrix and its spectral analysis. The symmetry correspondence matrix is derived from a dissimilarity matrix $\boldsymbol{S}$ which is built in a way that value S_{ij} should represent a minimal distance between the original shape and the shape which is transformed by such a transform g that the point $\boldsymbol{x}_i$ ends up in the point $\boldsymbol{x}_j$. The transform g is from a given group of transformations such as rigid transformations.

The method seems to be able to detect the global plane of symmetry of a 3D shape as well but it does not seem to be its main purpose. The symmetry plane detection is shown on a few objects which mostly are not

perfectly symmetrical but also do not exhibit any missing parts (see Figure 2.10). An advantage of this method is the fact that it works on point clouds and it also seems to be quite general. Overall the presented results are good, but the method's running time seems to be rather high. The results shown in the paper reveal that the running times on objects consisting of 1000 points are in matters of minutes.

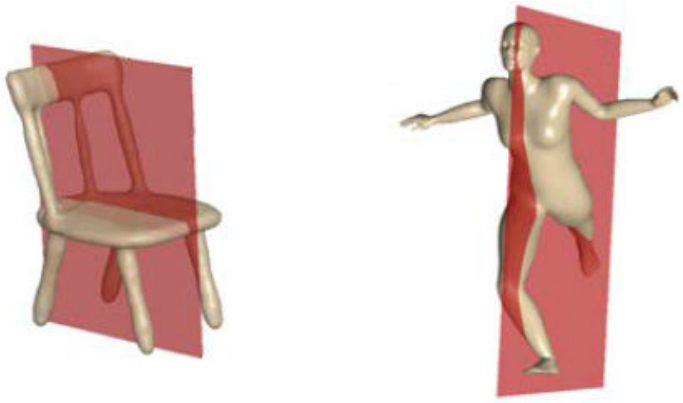


Figure 2.10: Results of the symmetry plane detection using the method by Lipman et al. (Figure taken from [13].)

2.10 Kakarala et al. (2013)

This paper [9] presents a method for detecting a plane of symmetry of objects represented by a triangle mesh but it seems usable for objects represented by a point cloud as well. In this method the shape is approximated with spherical harmonics creating a star-shaped surface and the symmetry estimation is then done on this shape. The authors mention an observation that if a real-valued function has symmetry across the origin then its Fourier transform is real-valued. They apply this observation on the spherical harmonics to derive an error function whose optimization leads to the symmetry plane. The authors also mention that the input surface should be uniformly sampled before applying their method because it helps the symmetry estimation.

The paper presents several results of the proposed method but all of them seem to be on objects which are perfectly or nearly perfectly symmetrical. Also all of these objects seem to be very simple having quite a low vertex count (see Figure 2.11). The fact that this method is theoretically usable on point clouds as well as on triangle meshes can be considered its advantage. Its disadvantage is the fact that only symmetry planes passing through the origin seem to be detected since the detection is done on a shape created

by the spherical harmonics. Also it is quite doubtful whether this approach would succeed on an object with missing parts.

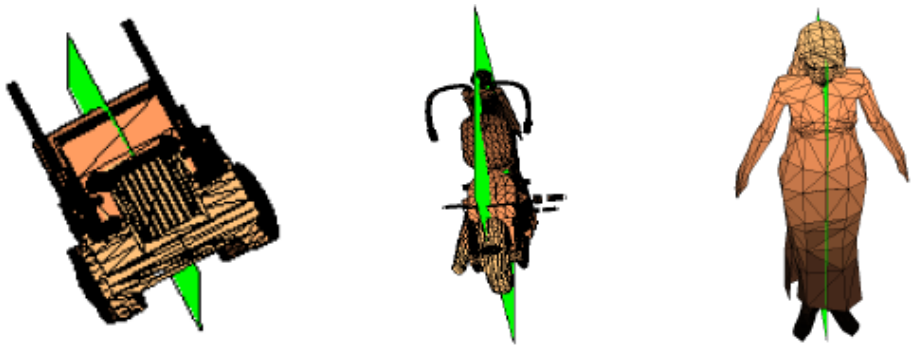


Figure 2.11: Results of the symmetry plane detection using the method by Kakarala et al. (Figure taken from [9].)

2.11 Sipiran et al. (2014)

The method presented by this paper [21] aims to detect a reflectional symmetry of 3D shapes exhibiting missing parts which are represented by triangle meshes. First step of this method is to detect local features of the given 3D shape. This is done using the theory of heat diffusion on manifolds. A function is defined which associates the accumulation of heat up to time t to each point on the surface $\boldsymbol{x}$. This function is calculated using the eigenvalues and eigenvectors of the Laplace-Beltrami operator. As the feature points, local maxima of the heat accumulation function are taken (see Figure 2.12). Pairs of the feature points then generate the candidate symmetry planes but only pairs of points which have a similar value of the heat accumulation function are considered. The last stage of this algorithm is a voting process where other pairs of points are tested against the plane and the more pairs of points are considered to be symmetrical with respect to the given plane the more votes the plane gets. There are several criteria designed to decide whether or not a plane is considered a plane of symmetry of a given pair of points. Also only points in which the mean curvature is higher than some threshold are used in the voting process. In the end the plane with the highest vote count can be declared the resulting plane of symmetry. In some cases a set of more than one plane is taken as a result.

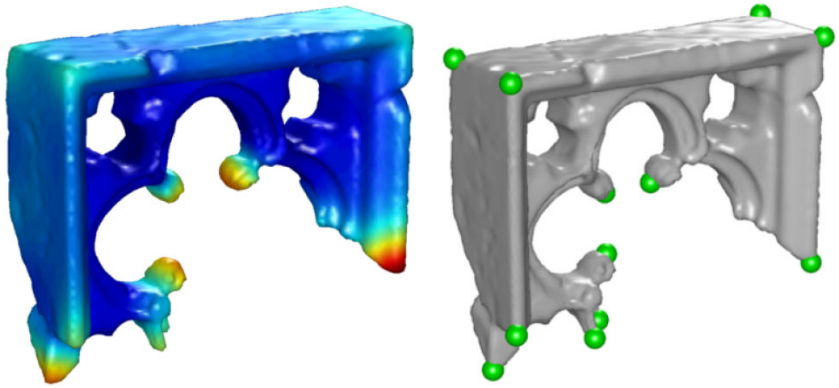


Figure 2.12: The heat accumulation function (left) (the function values increase as the color goes from blue to yellow and to red) and the feature points identified as local maxima of this function (right). (Figure taken from [21].)

This method seems to give very good results even when used on objects with very high level of missing parts (see Figure 2.13). It also seems to be able to detect more than one plane of symmetry in case of objects which are symmetrical with respect to more planes. One of the disadvantages of this method is the fact that it only works on manifold triangle meshes. Also it uses local features to create the candidate planes which seems to be a little limiting since there can be objects which do not have any feature points. Apart from this the feature points are located using eigenvectors and eigenvalues of the Laplace-Beltrami operator which is a matrix of size $v \times v$ where v is the number of the mesh vertices. This raises a question whether this method is computable for larger meshes (with tens of thousands of vertices or more).

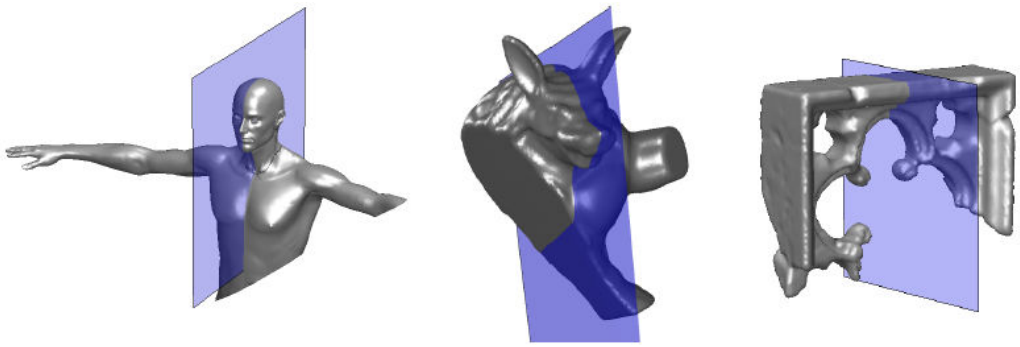


Figure 2.13: Results of the symmetry plane detection using the method by Sipiran et al. (Figure taken from [21].)

2.12 Korman et al. (2015)

This article [10] presents a method for detecting rigid symmetries (combinations of rotational and reflectional symmetries) of 3D shapes. The shape is represented by a binary function which has a value of 1 for all points inside the shape and 0 for all points outside the shape. The authors propose a distortion measure which is the total amount of mismatched volume between the original shape and the transformed shape. They also propose a sampling of the transformation space and a quick approximation of the distortion which gives approximately the correct distortion value with overwhelmingly high probability. The distortion and transformation space sampling are used to detect all approximate symmetries of the shape. How approximate the symmetries should be is defined by the method's parameter. The authors also propose smoothing the shape in advance to decrease their method's running time.

The proposed method seems to be usable for detection of a symmetry plane of a given 3D shape. The results are mostly presented on perfectly or almost perfectly symmetrical shapes, most likely represented by polygonal meshes, but there are a few results shown on imperfectly symmetrical shapes where the detected symmetry is really approximate (see Figure 2.14). No results on objects with missing parts are shown. Disadvantage of this method is the fact that it does not consider translation in the symmetry transformations, only rotation and reflection, which means the detected plane or axis of symmetry must pass through the origin. Also it does not seem to work on point clouds since in order to decide which points are outside and which are inside a given shape, surface representation is needed.

Figure 2.14: Results of the symmetry plane detection using the method by Korman et al. (Figure taken from [10].)

2.13 Stephenson et al. (2015)

This paper [22] proposes two methods for detection of symmetry plane of 3D triangle meshes. The first step is common for both methods and it is the candidate plane creation. Only three candidate planes are created using PCA [27], specifically using the PCA's two eigenvectors and their cross product.

The first method uses a variation of Hausdorff distance which is applied to measure distance between the original mesh and the mesh created by reflecting the original mesh over the candidate plane. This measure is applied on each of the three candidate planes to determine whether or not it is the plane of symmetry.

The second method uses ray casting. Number of rays perpendicular to the candidate plane are casted at the triangle mesh and the intersections of each ray with the mesh are recorded. The distances between the intersections on one side of the candidate plane and intersections on the other side of the plane are used to form a symmetry measure of the given plane. This measure is then used in the same way as the Hausdorff distance measure in the first method.

The authors also suggest simplifying the triangle mesh to decrease the runtime of their methods and using a k-d tree to decrease the runtime of the ray casting method.

There are no visual results shown in the paper. The authors only state that both methods demonstrated 100% accuracy when used on perfectly symmetrical models and that the accuracy for approximate symmetry detection is more difficult to quantify. There are two quite obvious disadvantages of both proposed methods. The first one is the fact that they both only work on triangle meshes, although the first one could probably be modified to work on point clouds as well. The second disadvantage is the fact that the resulting plane has to pass through the origin since the candidate planes are created using PCA. Also it is questionable whether for models with significant missing parts the PCA-based candidate plane creation would detect the correct candidate planes at all.

2.14 Li et al. (2016)

This article [12] proposes a method for detection of symmetry plane of 3D triangle meshes. It first uses CPCA (continuous PCA) to align the model and then the model is translated so that the center of its bounding sphere is at the origin. Also the model is scaled so that the bounding sphere's radius

is 1. Next a set of viewpoints is sampled on a sphere around the model and in each viewpoint a camera is set to look at the origin. Using these cameras the model is rendered from each viewpoint using an orthogonal projection. The authors use a viewpoint entropy which depends on the areas of the rendered faces. This entropy is computed for each viewpoint which creates a viewpoint entropy distribution sphere. Since the symmetry planes of the model and of the viewpoint entropy distribution sphere are the same the symmetry plane detection is performed on the viewpoint entropy distribution sphere. Candidate planes are created using pairs of viewpoints with matching values of the viewpoint entropy and for each plane the rest of all pairs of viewpoints with matching entropy are verified to see whether they are symmetric with respect to the given candidate plane. If the number of symmetric pairs is great enough the given candidate plane is declared a symmetry plane.

The article shows results of the symmetry plane detection on quite many objects but all of them are perfectly or almost perfectly symmetrical and most of them seem to be quite simple (see Figure 2.15). The greatest disadvantage of this method is probably the fact that it only detects symmetry planes passing through the center of the model's bounding sphere. Also the method only works on triangle meshes and it is very unlikely that it would work on models with missing parts.

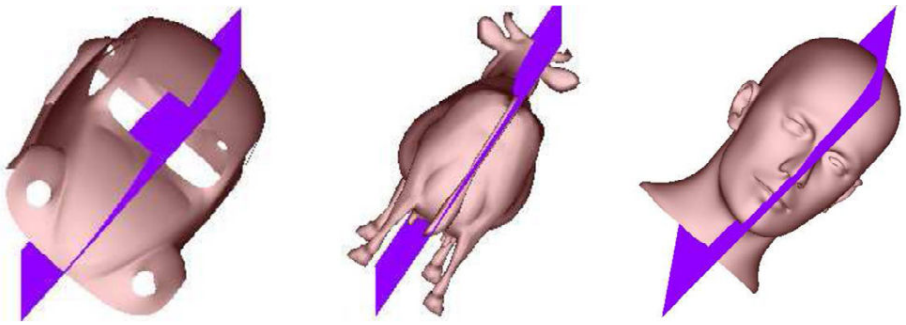


Figure 2.15: Results of the symmetry plane detection using the method by Li et al. (Figure taken from [12].)

2.15 Schiebener et al. (2016)

This paper [19] presents a method for detecting the plane of symmetry of an object represented by a point cloud and its use for incomplete object completion. Apart from the point cloud representing the input object, the method also needs a point cloud representing the object's surrounding and

position from where the points were scanned. It relies on the fact that a 3D object usually stands on some kind of supporting structure, such as the ground, and the first step is detecting the supporting plane which represents such a structure. Candidates for the supporting planes are generated from the object's surrounding using the RANSAC algorithm and only 10 best candidates are kept. Which supporting plane candidates are the best is determined using a simple rating defined by the authors. For each supporting plane candidate several symmetry plane candidates are created by sampling the space of planes orthogonal to the given supporting plane. The authors define rating of the candidate symmetry planes which uses the supporting plane and the scanner position to evaluate whether a given point reflected over a given plane ends up in a plausible location. The plane with the highest rating is declared the object's plane of symmetry.

Results of the symmetry detection and symmetry-based object completion are shown on a few quite simple objects (see Figure 2.16). An advantage of this method is the fact that it works on point clouds representing incomplete objects. Its main disadvantage is that it needs the object's surrounding and the scanner position to work. Also it only detects symmetry planes which are orthogonal to the supporting surface.

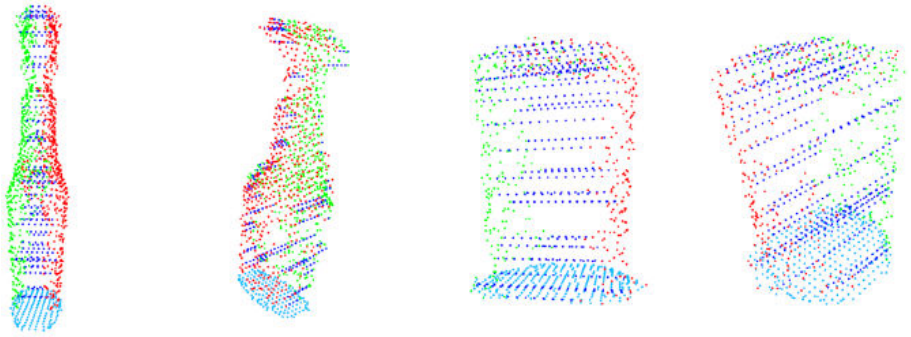


Figure 2.16: Results of the symmetry-based object completion using the method by Schiebener et al. The original points are red, the points generated by reflection over the detected symmetry plane are green, dark blue are the points on the sides and light blue are the bottom points. (Figure taken from [19].)

2.16 Hruda, Dvořák (2017)

In this paper [8] a method for detecting a symmetry plane of 3D triangle meshes is presented which is in many ways a simplified version of the method

described in Section 2.11. In the first step a given number of vertices with the highest Gaussian curvature are extracted as feature points. Pairs of these points which satisfy certain criteria are then used to create the candidate symmetry planes. Next a voting process is deployed to find the best symmetry plane. For each candidate plane all pairs of the feature points are tested against the given plane and if the given pair of points together with the plane satisfy certain criteria, the plane gets a vote. In the end the plane with the highest vote count is declared the resulting symmetry plane.

Results of this method are shown on several objects, also on some objects exhibiting missing parts and imperfect symmetries (see Figure 2.17). Most of these objects represent human faces or human heads but there are a few others as well. Overall the results seem to be good. An advantage of this method is that it appears to be quite simple. One of this method's disadvantages is the fact that it only works on triangle meshes. Also it seemingly does not work well with triangle meshes with a high number of vertices (tens of thousands or more) but this problem can be solved with mesh simplification. On the other side it seems to work well with meshes with very low vertex count which can be considered an advantage. The greatest disadvantage of this method is that it relies on quite many parameters (5 in total). Default values of these parameters are proposed but it is shown in the paper that these values are not universally optimal and that for some objects they do not ensure satisfying results.

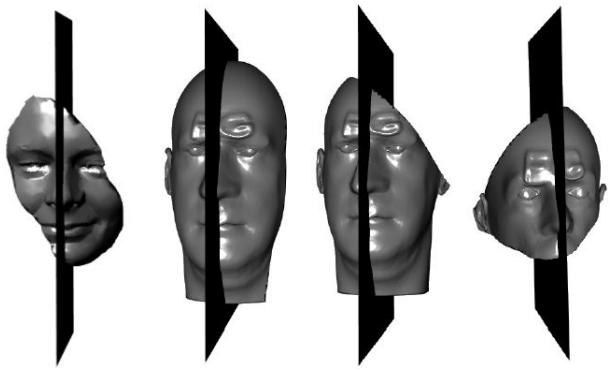


Figure 2.17: Results of the symmetry plane detection using the method by Hruda and Dvořák. (Figure taken from [8].)

2.17 Summary

Several methods for symmetry plane detection were described in this section and a subjective evaluation of each one of them was provided by the author

of this thesis. One important aspect of each method is what type of input data it works on, overall evaluation of this aspect was already provided at the beginning of this chapter.

Another important aspect is whether the given method puts any constrains on the planes it detects. Some of the described methods (specifically those described in Sections 2.2, 2.6, 2.10, 2.12, 2.13 and 2.14) only detect planes which pass through some reference point such as the origin, centroid or the center of mass.

These constrains are also related to whether the given method is able to detect approximate symmetries and symmetries on objects with missing parts. The methods described in Sections 2.3, 2.7, 2.8, 2.9, 2.12 and 2.15 seem to be usable (at least to some level) for approximate symmetry detection and some of them even seem to work on objects which exhibit some missing parts, although usually the missing parts are not so significant. Also two of these methods, specifically those from Sections 2.3 and 2.15, need some additional information to work, such as the scanner position. The only method which seems to be specifically designed for global symmetry plane detection on objects with significant missing parts is the one described in Section 2.11. The method described in Section 2.16 seems usable for detecting symmetry planes of objects with quite significant missing parts as well but this method is not of such importance since it is in many ways just a simplified version of the method from Section 2.11 and also its results seem to depend quite strongly on its parameter configuration.

3 Proposed Method

The goal of this work was to design a method which could be used to detect the plane of global reflectional symmetry of a given 3D object and would possibly be able to detect also an approximate symmetry and symmetry of an object with missing parts. In this chapter a new method is proposed which seems to, more or less, fulfill the above mentioned requirements. The general method can be used on objects represented by only a set of points (a point cloud) but when more information about the object is available the method can be further extended to use this information in its favor.

The proposed method is based on maximizing a specific symmetry measure which is continuous and even differentiable. The input point cloud is simplified to a very low number of points and pairs of points of this simplified point cloud are used to create a number of candidate planes. From these candidate planes the one with the highest symmetry measure is chosen. For time reasons, the symmetry measure is computed on another simplified version of the input point cloud. In the end a local optimization is performed to find the final plane of symmetry. The symmetry measure also contains weights which allow using some additional information about the input object.

The detailed description of the proposed symmetry detection method is given in the rest of this chapter.

3.1 Background

In order to describe the method itself, a few terms have to be defined first. The goal of the proposed method is to detect the plane which best captures the reflectional symmetry of a given 3D object. Let us define a general plane P by its implicit equation $P : ax + by + cz + d = 0$ or in the vector notation $[a, b, c, d][x, y, z, 1]^T = 0$ where $[a, b, c, d]^T = \boldsymbol{p}$ is a four-dimensional vector of the plane coefficients. Therefore we are searching for such a vector $\boldsymbol{p}$ which represents the plane of symmetry of the given object. We also need to define a function $\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{x}) = [r_x(\boldsymbol{p}, \boldsymbol{x}), r_y(\boldsymbol{p}, \boldsymbol{x}), r_z(\boldsymbol{p}, \boldsymbol{x})]^T \in E^3$ which reflects a point $\boldsymbol{x} = [x, y, z]^T \in E^3$ over a plane P represented by $\boldsymbol{p}$. This function

can be defined as shown in Equation 3.1.

$$\begin{gathered}\boldsymbol{r}(\boldsymbol{p},\boldsymbol{x}) = \boldsymbol{r}([a,b,c,d]^T,[x,y,z]^T) = \\ [x,y,z]^T - 2\frac{ax+by+cz+d}{\sqrt{a^2+b^2+c^2}}\cdot\frac{[a,b,c]^T}{\sqrt{a^2+b^2+c^2}}\end{gathered} \tag{3.1}$$

The expression $\frac{ax+by+cz+d}{\sqrt{a^2+b^2+c^2}}$ represents the signed distance of $\boldsymbol{x}$ from P and $\frac{[a,b,c]^T}{\sqrt{a^2+b^2+c^2}}$ is the normalized normal vector of P. The function can be further simplified as shown in Equation 3.2 and rewritten as shown in Equation 3.3.

$$\boldsymbol{r}(\boldsymbol{p},\boldsymbol{x}) = [x,y,z]^T - 2\frac{ax+by+cz+d}{a^2+b^2+c^2}[a,b,c]^T \tag{3.2}$$

$$\begin{gathered}\boldsymbol{r}(\boldsymbol{p},\boldsymbol{x}) = \\ [x-2a\frac{ax+by+cz+d}{a^2+b^2+c^2}, y-2b\frac{ax+by+cz+d}{a^2+b^2+c^2}, z-2c\frac{ax+by+cz+d}{a^2+b^2+c^2}]^T\end{gathered} \tag{3.3}$$

The $ax+by+cz$ expression can be replaced with $[a,b,c]\boldsymbol{x}$ and since $[a,b,c]^T$ is the normal vector of the plane P, we can let $\boldsymbol{n_p}$ denote it and use $\boldsymbol{n_p}^T\boldsymbol{x}$ instead. It can also be noticed that $a^2+b^2+c^2 = \boldsymbol{n_p}^T\boldsymbol{n_p}$. This gives us the form shown in Equation 3.4.

$$\begin{gathered}\boldsymbol{r}(\boldsymbol{p},\boldsymbol{x}) = [x-2a\frac{\boldsymbol{n_p}^T\boldsymbol{x}+d}{\boldsymbol{n_p}^T\boldsymbol{n_p}}, y-2b\frac{\boldsymbol{n_p}^T\boldsymbol{x}+d}{\boldsymbol{n_p}^T\boldsymbol{n_p}}, z-2c\frac{\boldsymbol{n_p}^T\boldsymbol{x}+d}{\boldsymbol{n_p}^T\boldsymbol{n_p}}]^T = \\ \boldsymbol{x}-2\frac{\boldsymbol{n_p}^T\boldsymbol{x}+d}{\boldsymbol{n_p}^T\boldsymbol{n_p}}\boldsymbol{n_p}\end{gathered} \tag{3.4}$$

It should be noted that from Equation 3.3 it is obvious that when $\boldsymbol{x}$ is constant, all three function r_x, r_y and r_z, which represent the components of $\boldsymbol{r}$, are continuous and differentiable except for $\boldsymbol{p} = [0,0,0,d]^T$ which does not represent a valid plane.

3.2 Symmetry Measure

Let us consider a perfectly reflectionally symmetrical 3D object sampled with n points, which form a point cloud $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$, in such a way that the point cloud is perfectly symmetrical as well. This means that such a plane P represented by $\boldsymbol{p} = [a,b,c,d]^T$ exists where for any $\boldsymbol{x}_i \in X$ there is $\boldsymbol{x}_j \in X$ such that $\boldsymbol{r}(\boldsymbol{p},\boldsymbol{x}_i) = \boldsymbol{x}_j$. In other words any point in the point cloud X, after it gets reflected over the plane P, ends up in another point of the point cloud X or in itself. Detection of the symmetry plane P of such

an object could probably be done by computing distances between $\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{x}_i)$ and $\boldsymbol{x}_j$ for all possible pairs of points $\boldsymbol{x}_i$, $\boldsymbol{x}_j \in X$ and minimizing their sum. Formally this can be expressed as minimizing the error function shown in Equation 3.5 for $\boldsymbol{p}$ or the one shown in Equation 3.6 when the minimization is considered in the least square sense.

$$e^1_X(\boldsymbol{p}) = \sum_{i=1}^{n}\sum_{j=1}^{n} ||\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{x}_i) - \boldsymbol{x}_j|| \tag{3.5}$$

$$e^2_X(\boldsymbol{p}) = \sum_{i=1}^{n}\sum_{j=1}^{n} ||\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{x}_i) - \boldsymbol{x}_j||^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} (\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{x}_i) - \boldsymbol{x}_j)^T(\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{x}_i) - \boldsymbol{x}_j) \tag{3.6}$$

The problem with such an error function is that it considers the point cloud to be perfectly or almost perfectly symmetrical. If the point cloud exhibits imperfect symmetry, has outliers or represents an object with significant missing parts, minimizing this error function would fail to detect the desired plane of symmetry, since such a plane does not represent its minimum. This is because, even if the plane visually correctly captures the object's symmetry, some points in the point cloud, when they get reflected over this plane, can end up in locations where there are no other close points of the point cloud which results in a higher value of the error function. It can also be explained in the way that minimizing the error function for $\boldsymbol{p}$ does not force the points in the point cloud to end up close to other points of the point cloud when reflected over the plane, it only minimizes the overall error sum. This behavior is very undesirable since the method should be able to detect not just perfect, but also imperfect and approximate symmetries and possibly symmetries on objects with significant missing parts.

This problem can be solved by using the opposite approach. Instead of computing and summing distances of reflected points from other points we can compute their similarities. For each pair of points $\boldsymbol{x}_i$, $\boldsymbol{x}_j \in X$ we will compute the distance between $\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{x}_i)$ and $\boldsymbol{x}_j$ but instead of using the distance directly we will transform it into similarity using a similarity function which will have a maximum value in the case $\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{x}_i) = \boldsymbol{x}_j$ and its value will approach zero with the increasing distance between $\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{x}_i)$ and $\boldsymbol{x}_j$. We can sum these similarities of all possible pairs $\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{x}_i)$, $\boldsymbol{x}_j$, where $\boldsymbol{x}_i$, $\boldsymbol{x}_j \in X$, and try to find a plane that maximizes this sum which can be called a symmetry measure. Maximizing such measure will basically force the maximum of the points in the point cloud to reflect as close as possible to other points of the point cloud. This can formally be expressed

as maximizing the symmetry measure function shown in Equation 3.7.

$$s_X(\boldsymbol{p}) = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\varphi(||\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{x}_i) - \boldsymbol{x}_j||) \tag{3.7}$$

The similarity function $\varphi(l)$ is some radial function which equals 1 for $l = 0$ and its value is decreasing and approaching 0 as l increases, w_{ij} are weights of given pairs of points and will be discussed later, for now we will consider all the weights to have the value of 1.

3.2.1 Similarity Function

As φ, for example the Gaussian function or one of the Wendland's functions [26] can be used. Whether the symmetry measure $s_X(\boldsymbol{p})$ is continuous and differentiable depends on the choice of the φ function, specifically $s_X(\boldsymbol{p})$ is continuous and differentiable (except for $\boldsymbol{p} = [0, 0, 0, d]^T$) when $\varphi(l)$ is continuous and differentiable for $l \in \langle 0; \infty)$ and $\frac{d}{dl}\varphi(0) = 0$. This holds, of course, for the Gaussian function and also for most of the Wendland's functions. The continuity and differentiability of s_X will be useful in the last step of our method. Although the Gaussian function is simple, easy to implement and could be chosen as φ without significant problems, we used a modified Wendland's function shown in Equation 3.8 instead. The reason will be clear from the following text.

$$\varphi(l) = \begin{cases} (1 - \frac{1}{2.6}\alpha l)^5(8(\frac{1}{2.6}\alpha l)^2 + 5\frac{1}{2.6}\alpha l + 1) & \alpha l \leq 2.6 \\ 0 & \alpha l > 2.6 \end{cases} \tag{3.8}$$

The value α is the shape parameter of the function. The multiplier $\frac{1}{2.6}$ is our modification which ensures that the function is similar to the Gaussian function ($e^{-(\alpha l)^2}$), this can be useful because this way both our function and the Gaussian function will give similar results for the same value of α. The main difference between the Gaussian function and our Wendland's function is that the Wendland's function is equal to 0 for $\alpha l > 2.6$. Why this is useful will be discussed later. Figures 3.1 and 3.2 provide a visual comparison of the Gaussian function and our modified Wendland's function for $\alpha = 1$.

The shape parameter α can be set appropriately using the size of the input point cloud. We set α as shown in Equation 3.9.

$$\alpha = \frac{15}{l_{avrg}} \tag{3.9}$$

The value l_{avrg} is the average distance of the point cloud points from its centroid and the value 15 was chosen experimentally.

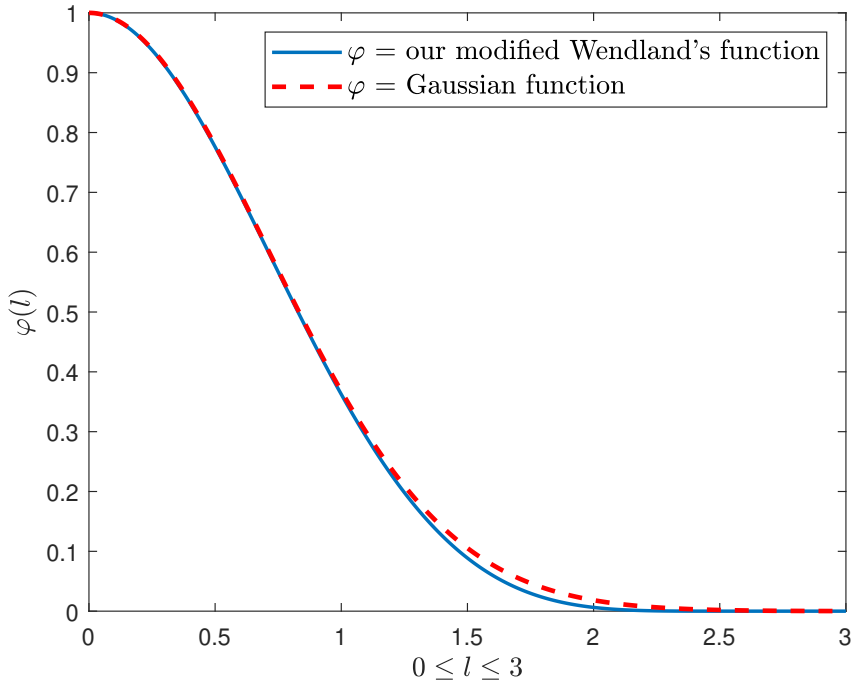


Figure 3.1: Visual comparison of the Gaussian function (red) and our modified Wendland's function (blue) for $\alpha = 1$ and $0 \leq l \leq 3$.

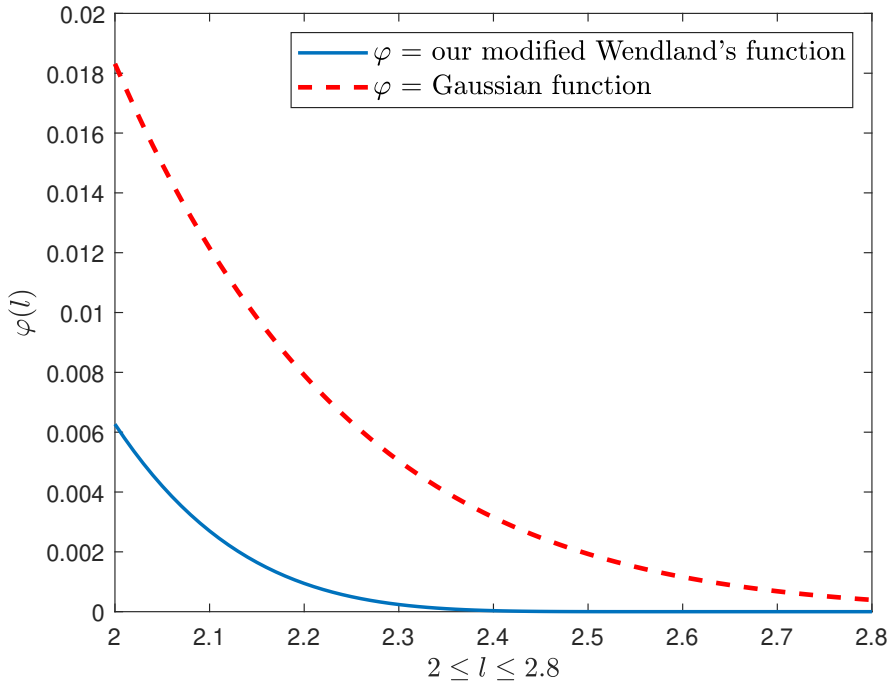


Figure 3.2: Visual comparison of the Gaussian function (red) and our modified Wendland's function (blue) for $\alpha = 1$ and $2 \leq l \leq 2.8$.

3.2.2 Efficient Computation

It is obvious that the brute force computation of $s_X(\boldsymbol{p})$ for a given plane P represented by $\boldsymbol{p}$ has time complexity of $O(n^2)$ but it can be noticed that for many pairs $\boldsymbol{x}_i$, $\boldsymbol{x}_j \in X$ the similarity $\varphi(||\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{x}_i) - \boldsymbol{x}_j||)$ is 0. Specifically we only need to compute the similarities for such pairs $\boldsymbol{x}_i$, $\boldsymbol{x}_j \in X$ for which $||\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{x}_i) - \boldsymbol{x}_j|| \leq \frac{2.6}{\alpha}$. Therefore, some auxiliary data structure can be used to filter those pairs for which the similarity has to be computed. We use a uniform grid with the cell size $\frac{2.6}{\alpha} \times \frac{2.6}{\alpha} \times \frac{2.6}{\alpha}$ which is implemented as a hash table where the key is a triplet of the cell coordinates and the value corresponding with a given key is a list of the point cloud points contained in the given cell. During the computation of $s_X(\boldsymbol{p})$, after a point $\boldsymbol{x}_i$ is reflected over the given plane and ends up in a cell c, only points in c and cells adjacent to c are considered for the symmetry measure computation. Points in any cells which are farther from c have zero similarity with the reflected point and do not have to be considered.

If the Gaussian function was used as φ instead of the modified Wendland's function, even points in farther cells would have non-zero similarity with the reflected point which would result in $s_X(\boldsymbol{p})$ being computed with a small error. Although this error is quite insignificant when the symmetry measure is computed for a given plane, it slightly disrupts the continuity and differentiability of the function s_X.

3.3 Point Cloud Simplification

The input point cloud can be simplified in order to compute the symmetry measure faster. The acceleration is especially noticeable when the number of points in the input point cloud is very high. The simplification is also used during the candidate plane creation which will be described later.

Our simplification algorithm is very simple and considerably fast. It uses the same kind of grid as used for the efficient computation of the symmetry measure (see Section 3.2.2). The grid is created for the input point cloud with the cell size $\frac{4 \cdot l_{avrg}}{k} \times \frac{4 \cdot l_{avrg}}{k} \times \frac{4 \cdot l_{avrg}}{k}$ and each occupied cell gives one point of the simplified point cloud by averaging all points contained in the cell. The constant 4 in the cell size was chosen so that for $k = 1$ the whole point cloud could approximately fit into one cell. We want to simplify the input point cloud to approximately a given number of points, let m denote this number, so we first set $k = 1$ and we repeat the simplification of the original point cloud with increasing value of k until the number of points of the simplified point cloud reaches at least m. We increase k by 1 each time.

3.4 Candidate Plane Creation

In order to find the symmetry plane of the given point cloud X as the plane which maximizes the symmetry measure s_X, we first have to create a set of candidate symmetry planes where the best candidate for the symmetry plane will be the one for which $s_X(\boldsymbol{p})$ is maximal. It would be convenient to create only such candidate planes that pass through X and for which there is at least some chance that they actually are the symmetry planes of X. We can create the candidate planes by taking each pair of points $\boldsymbol{x}_i$, $\boldsymbol{x}_j \in X$, $i \neq j$ and create the plane of symmetry of these two points. Such a plane is created by computing its normal vector as $\boldsymbol{n_p} = \boldsymbol{x}_i - \boldsymbol{x}_j$ and its d coefficient as $d = -\boldsymbol{n_p}^T(\frac{\boldsymbol{x}_i+\boldsymbol{x}_j}{2})$.

Problem is that creating the candidate planes this way directly on X would result in an overwhelming number of planes, at least in such a case when X consists of more than a few tens of points, which would be uncomputable. Therefore, we first simplify X using the simplification algorithm described in Section 3.3 with $m = 100$, creating a new point cloud X_{cand} consisting of approximately 100 to 110 points. The candidate plane creation is then performed on X_{cand} creating approximately 5000 to 6000 candidate planes. The value of $m = 100$ was chosen experimentally so that the simplification results in a reasonable number of candidate planes and X_{cand} is still a sufficient, although very rough, approximation of X.

3.5 Selecting the Best Candidate

Once we have the candidate planes, we have to select the one with the highest symmetry measure. If the input point cloud X consists of a large number of points (tens of thousands or more), the symmetry measure computation for all the candidate planes on X takes quite a lot of time. Therefore, we first simplify X, using the simplification algorithm described in Section 3.3, creating a new point cloud X_{simp} consisting of approximately 1000 points (we use $m = 1000$ for the simplification). The number 1000 was experimentally chosen because with such a number of points X_{simp} seems to represent the input point cloud X sufficiently and the computational time is acceptable.

Now we compute the symmetry measure $s_{X_{simp}}(\boldsymbol{p})$ for all candidate planes P represented by $\boldsymbol{p}$ and we select the one for which the symmetry measure is maximal. Such a plane now represents the best candidate for the symmetry plane of X_{simp} and, therefore, of the input point cloud X as well.

3.6 Final Optimization

It is obvious that the best symmetry plane candidate selected in the previous step is not in general the global maximum of $s_{X_{simp}}(\boldsymbol{p})$, it is just very likely close to it. So we can use some numerical optimization method, which starts from the best candidate plane, in order to get from it to the actual global maximum of $s_{X_{simp}}(\boldsymbol{p})$. Since the numerical optimization methods usually operate best on functions which are continuous and differentiable, this is where the continuity and differentiability of $s_{X_{simp}}(\boldsymbol{p})$ can be useful, it can, for example, make the numerical method converge more quickly. We use the Nelder-Mead optimization method [16] implemented in *Microsoft Solver Foundation* [3].

Before the numerical optimization is run, the point cloud should be translated, together with the best candidate plane, so that the average of all points in the point cloud is at the origin. This translation can as well be done at the beginning of the whole algorithm with the input point cloud X, and the whole algorithm can be performed on the translated point cloud, which is what we do. We can define this translation as a translation by a vector $\boldsymbol{t}$ which is defined as shown in Equation 3.10.

$$\boldsymbol{t} = -\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i \tag{3.10}$$

The translation is important because if the point cloud was very far from the origin then the best candidate plane would be very far from it as well. Then even a slight change of its normal vector could cause quite a noticeable change of the plane position which would be bad for the convergence of the numerical method. After the numerical optimization is over we take the plane which it converged to, we translate it by $-\boldsymbol{t}$ and we declare the resulting plane the plane of symmetry of the point cloud X.

3.7 Weights

Until now we have been ignoring the weights w_{ij} in the symmetry measure s_X (see Equation 3.7) and considered all of them to equal to 1. Using the weights the algorithm can be made to work even better or to work on objects on which it does not work without the weights. The weight w_{ij} can be written as $w_{ij} = w_{ij}^s w_{ij}^d(\boldsymbol{p})$ where w_{ij}^s is a static weight and $w_{ij}^d(\boldsymbol{p})$ is a dynamic weight. It can be noticed that the dynamic weight depends on the plane P represented by $\boldsymbol{p}$. Now the symmetry measure s_X can be rewritten

as shown in Equation 3.11.

$$s_X(\boldsymbol{p}) = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}^s w_{ij}^d(\boldsymbol{p})\varphi(||\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{x}_i) - \boldsymbol{x}_j||) \tag{3.11}$$

The dynamic weights can, for example, represent the symmetry of normal vectors or directions of principle curvatures in corresponding pairs of points with respect to a given plane. The static weights can be set to represent the importance of given pairs of points. In other words, the more it is desired for the point $\boldsymbol{x}_i$ to end up in or near the point $\boldsymbol{x}_j$ after reflecting it over the plane of symmetry the higher the static weight w_{ij}^s should be. The importance can be set as a similarity of some kind of feature function values in the given two points. As this feature function, for example, some type of curvature can be used, since it can be expected that in two symmetrical points there are similar curvature values.

In the rest of this section we give one possible way how the static and dynamic weights can be set. To set the dynamic weights we use the symmetry of normal vectors and for the static weights we use the similarity of the values of Gaussian curvature. It should be noted that the Gaussian curvature is not a universally optimal feature function to use but it has already proven to be usable for quantifying point similarity in many cases [8]. How such weighting can be useful will be shown in Chapter 4 and how both dynamic and static weights are set exactly will be described in the following text.

Since a point cloud representation of an object does not implicitly give enough information to set the weights (the normal vectors and the values of Gaussian curvature), we will now consider the object to be represented by a manifold triangle mesh. On such mesh the normal vectors and the values of Gaussian curvature in all its vertices can be estimated in various ways. We estimate the normal vectors of all vertices by summing the normal vectors of triangles adjacent to the given vertex and normalizing the resulting vector. The values of Gaussian curvature are estimated the same way as in [8].

Apart from the set of points $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ we now also have a set of unit normal vectors $N = \{\boldsymbol{n}_1, \boldsymbol{n}_2, ..., \boldsymbol{n}_n\}$ and a set of Gaussian curvatures $G = \{g_1, g_2, ..., g_n\}$ where $\boldsymbol{n}_i$ is the unit normal vector in the point $\boldsymbol{x}_i$ and g_i is the Gaussian curvature in the point $\boldsymbol{x}_i$.

We also need to define how the normal vectors and the values of Gaussian curvature will be determined when the simplification algorithm described in Section 3.3 is applied. When a new point of the simplified point cloud is created by averaging the points in a given cell of the simplification grid, its normal vector is determined by averaging the normal vectors in all points in the cell and normalizing the resulting vector. Its Gaussian curvature is

taken from the point in the cell for which the absolute value of its Gaussian curvature is the highest.

3.7.1 Computing the Dynamic Weights

In order to measure the reflectional symmetry of two unit normal vectors in two points, we first have to define a function $\boldsymbol{r_n}(\boldsymbol{p}, \boldsymbol{n})$ which reflects a unit normal vector $\boldsymbol{n}$ over the plane P represented by $\boldsymbol{p}$. This function is almost the same as the function $\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{x})$ (see Equation 3.4) that reflects a point $\boldsymbol{x}$ over P, the only difference is that, when reflecting the normal vector, the plane must be translated so that it passes through the origin. This can be done simply by leaving out the d coefficient of the plane, therefore, the function $\boldsymbol{r_n}$ can be defined as shown in Equation 3.12.

$$\boldsymbol{r_n}(\boldsymbol{p}, \boldsymbol{n}) = \boldsymbol{n} - 2\frac{\boldsymbol{n_p}^T\boldsymbol{n}}{\boldsymbol{n_p}^T\boldsymbol{n_p}}\boldsymbol{n_p} \tag{3.12}$$

The symmetry of two normals $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ is defined as the similarity of $\boldsymbol{r_n}(\boldsymbol{p}, \boldsymbol{n}_i)$ and $\boldsymbol{n}_j$. To quantify such similarity we apply the similarity function $\varphi(l)$ on the angle between $\boldsymbol{r_n}(\boldsymbol{p}, \boldsymbol{n}_i)$ and $\boldsymbol{n}_j$. As φ we use our Wendland's function (see Equation 3.8) with $\alpha = 4$. This value of α was chosen so that for angle $\frac{\pi}{16}$ (that is 11.25°) the similarity is approximately 0.5 because it seems reasonable that only for such low angle the similarity is significant (close to 1). The dynamic weights $w_{ij}^d(\boldsymbol{p})$ are therefore defined as shown in Equation 3.13.

$$w_{ij}^d(\boldsymbol{p}) = \varphi(arccos(\boldsymbol{r_n}(\boldsymbol{p}, \boldsymbol{n}_i)^T\boldsymbol{n}_j)), \quad with \quad \alpha = 4 \tag{3.13}$$

Since $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ are normalized, the expression $\boldsymbol{r_n}(\boldsymbol{p}, \boldsymbol{n}_i)^T\boldsymbol{n}_j$ is the cosine of the angle between $\boldsymbol{r_n}(\boldsymbol{p}, \boldsymbol{n}_i)$ and $\boldsymbol{n}_j$. It should be noted that the symmetry measure remains continuous when such dynamic weighting is used. For the symmetry measure to also stay differentiable we now need to fulfill another condition which is $\frac{d}{dl}\varphi(\pi) = 0$. Fortunately, this holds for $\alpha = 4$ because $4\pi > 2.6$ and for any $\alpha l > 2.6$ it is that $\frac{d}{dl}\varphi(l) = 0$.

3.7.2 Setting the Static Weights

As mentioned before, we use the similarity of Gaussian curvatures to set the static weights, see Equation 3.14.

$$w_{ij}^s = \begin{cases} \frac{min(|g_i|,|g_j|)}{max(|g_i|,|g_j|)} & |g_i| \geq \frac{g_{avrg}}{h} \wedge |g_j| \geq \frac{g_{avrg}}{h} \wedge g_i g_j > 0 \\ 0 & otherwise \end{cases} \tag{3.14}$$

The weight is non-zero only when both curvatures g_i and g_j have the same sign and their absolute values are both greater than the threshold $\frac{g_{avrg}}{h}$ where g_{avrg} is the average of absolute values of Gaussian curvatures in all points and h is a constant which we set as $h = 100$. This ensures that Gaussian curvatures with very small absolute values are not considered. The value g_{avrg} is computed as shown in Equation 3.15.

$$g_{avrg} = \frac{1}{n} \sum_{i=1}^{n} |g_i| \tag{3.15}$$

3.7.3 Filtering the Candidate Planes

The weights do not have to be used only when computing the symmetry measure, they can also be used to lower the number of candidate planes. Suppose we have the input object defined by the point cloud X, the set of normal vectors N and the set of Gaussian curvatures G and we simplify it for the candidate plane creation (see Section 3.4) creating the point cloud X_{cand}, a new set of normal vectors N_{cand} and a new set of Gaussian curvatures G_{cand}. When creating a candidate plane P, represented by $\boldsymbol{p}$, as a symmetry plane of $\boldsymbol{x}_i \in X_{cand}$ and $\boldsymbol{x}_j \in X_{cand}$, we can test whether the weights w_{ij}^s and $w_{ij}^d(\boldsymbol{p})$, computed on N_{cand} and G_{cand}, are high enough and if not, we will not consider the given plane a candidate plane anymore and will not compute its symmetry measure. It should be noted that the normal vector information and Gaussian curvature information are quite damaged by the simplification process and therefore we should not set the conditions for the weights too constraining. Specifically we consider the plane P, created as a symmetry plane of $\boldsymbol{x}_i \in X_{cand}$ and $\boldsymbol{x}_j \in X_{cand}$, a candidate plane if $w_{ij}^s > 0$ and $w_{ij}^d(\boldsymbol{p}) > 0.25$. These thresholds were chosen experimentally.

3.7.4 Additional Notes

There are numerous ways how both the static and dynamic weights can be set and each of them can be suitable for different types of input data. In addition, using the weights can also have influence on the numerical optimization which we use as the final step of our algorithm (see Section 3.6). When no weights are used, the symmetry measure s_X has non-negligible value for almost any plane which passes through the object. This is because, even when the given plane is not the plane of symmetry of the point cloud, at least some points, when they get reflected over this plane, will end up somewhere near other points of the point cloud increasing the symmetry

measure. This probably also causes the symmetry measure to have plenty of local extrema.

When the weights are used, they set more constrains on the symmetry measure. For example, for a given random plane passing through the point cloud, which is not its symmetry plane, it is quite likely that there will be some pairs of points which are symmetrical with respect to this plane, but it is not that likely that all these pairs will also have symmetrical normal vectors and it is even less likely that the points in all these pairs will also have similar Gaussian curvatures. It can be expected that this causes the symmetry measure to have non-negligible or even non-zero values only for planes which are considerably close to the symmetry plane being more smooth and having less significant local extrema. These observations could be useful not just for deciding how to set the weights but also for choosing the right numerical optimization method for the last step.

It should be noted that these are just theoretical notes and that we neither used these observations in practice nor deeper tested their validity.

4 Results

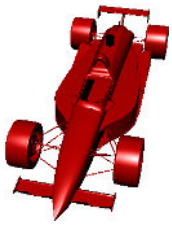
The designed method was tested on a computer with CPU *Intel® Pentium® Processor N3540* (clock rate 2.16 GHz, 4 cores, L1 *cache* 224 kB, L2 *cache* 2 MB) and 4GB of memory with clock rate of 666MHz and it was implemented in *C#*. We used the mesh processing framework provided by Doc. Ing. Libor Váša, Ph.D. (which is the same framework as used for [25]) for input data loading and processing and for visualization of the results. Also, some parts of the testing application, mainly related to the visualization, were implemented previously by, or in cooperation with, Bc. Jan Dvořák when working on [8]. The implementation is available on the DVD.

For testing we used several artificial objects which are strongly symmetrical and several real 3D-scanned objects. The artificial objects are depicted in Figure 4.1. The Ant, the Starship and the Formula (Figures 4.1b, 4.1c and 4.1d) are part of *The Princeton Shape Benchmark* [4].

(a) Lion (2213 points)

(b) Ant (3495 points)

(c) Starship (3099 points)



(d) Formula (10969 points)

Figure 4.1: Strongly symmetrical artificial 3D objects.

Figure 4.2 shows four 3D-scanned human faces provided by the authors of the *Fidentis Project* [2]. Other real 3D-scanned objects are depicted in Figure 4.3. The Armadillo (Figure 4.3a) was acquired from *The Stanford 3D Scanning Repository* [5], the Embrasure and Column base (Figures 4.3b and 4.3c) are 3D-scanned pieces of architecture taken from *PRESIOUS 3D Data Sets* [1]. These three objects (Armadillo, Embrasure and Column base) were specifically selected for comparison of our method with the method by Sipiran et al. [21] (see Section 2.11) because the authors of this method used

these three objects for testing as well. The reason why we chose the method by Sipiran et al. for comparison will be given in Section 4.1.4.

(a) Face 1 (2500 points)

(b) Face 2 (2500 points)

(c) Face 3 (2500 points)

(d) Face 4 (2500 points)

Figure 4.2: 3D-scanned human faces.

(a) Armadillo (172974 points)

(b) Embrasure (117535 points)

(c) Column base (69512 points)

Figure 4.3: 3D-scanned objects selected for comparison with Sipiran et al.

We also artificially created several perfectly symmetrical objects. We took some of the previously shown objects and we used our symmetry detection method to find the best symmetry plane of each one of them. After that, for each object, we removed all the points on one side of the plane and we reflected all the remaining points over the plane creating a perfectly symmetrical object with known symmetry plane - the ground truth symmetry plane. Each object was randomly rotated to prevent the symmetry plane

from being axis aligned. These perfectly symmetrical objects, together with their ground truth symmetry planes, are depicted in Figure 4.4.

(a) Lion PS (2376 points)

(b) Ant PS (3808 points)

(c) Armadillo PS (108266 points)

(d) Embrasure PS (138998 points)

Figure 4.4: Perfectly symmetrical objects with their ground truth symmetry planes. (The letters PS stand for *perfectly symmetrical*.)

In the following text we show the results achieved using the basic method, which uses neither weights ($w_{ij} = 1$) nor any candidate plane filtering, and at the end of this chapter we also show several results of the modified version which uses the weighting and candidate plane filtering described in Section 3.7.

For the sake of visualization simplicity all the test objects shown above are represented by triangle meshes but it should be noted that for the basic version of our method only vertex positions were used.

4.1 Results of the Basic Method

The basic method, which does not use the weights, was tested on all the objects shown above and also on several damaged versions of these objects. We also tried adding noise to some of these objects to test how our method performs on noisy shapes. This section presents the results of all these tests

including a comparison with another existing method, the method by Sipiran et al. [21] (see Section 2.11).

For the perfectly symmetrical objects we provide a numerical quantification of our method's error, for the other objects we provide visual results.

4.1.1 Perfectly Symmetrical Objects

Evaluation of the symmetry detection results on approximately symmetrical objects can only be done visually, not numerically, because for such objects there is no correct symmetry plane and therefore no ground truth to compare the detected plane with. On the other side any perfectly reflectionally symmetrical object has at least one correct symmetry plane. If we take the artificially created perfectly symmetrical objects from Figure 4.4, it is quite obvious that all these objects are precisely symmetrical with respect to only a single plane which is known. This gives us the ground truth with which the symmetry detection results can be compared in order to quantify the precision of our method.

In order to quantify the precision or the error of a given plane we need a way to quantify the difference between two planes. We implemented and used three difference measures denoted γ, D_1 and D_2, how they are computed will be described in the following text.

Let us define two general planes $P_1 : a_1x + b_1y + c_1z + d_1 = 0$ and $P_2 : a_2x + b_2y + c_2z + d_2 = 0$ represented by their coefficients vectors $\boldsymbol{p}_1 = [a_1, b_1, c_1, d_1]^T$, $\boldsymbol{p}_2 = [a_2, b_2, c_2, d_2]^T$. We also denote $\boldsymbol{n}_{\boldsymbol{p}_1} = [a_1, b_1, c_1]^T$ the normal vector of the plane P_1 and $\boldsymbol{n}_{\boldsymbol{p}_2} = [a_2, b_2, c_2]^T$ the normal vector of the plane P_2.

Plane difference measure γ

The first plane difference measure, denoted $\gamma(\boldsymbol{p}_1, \boldsymbol{p}_2)$, is just an angle between the normal vectors of the two planes. The angle in degrees is computed as shown in Equation 4.1.

$$\gamma(\boldsymbol{p}_1, \boldsymbol{p}_2) = \frac{180}{\pi} arccos\left(\frac{|\boldsymbol{n}_{\boldsymbol{p}_1}^T \boldsymbol{n}_{\boldsymbol{p}_2}|}{||\boldsymbol{n}_{\boldsymbol{p}_1}||||\boldsymbol{n}_{\boldsymbol{p}_2}||}\right) \tag{4.1}$$

It can be noticed that this difference measure does not use the d coefficients of the planes, ignoring their mutual position completely.

Plane difference measure D_1

The second plane difference measure, denoted $D_1(\boldsymbol{p}_1, \boldsymbol{p}_2)$, is defined as the distance between the coefficient vectors of the normalized planes. By nor-

malized plane we understand a plane with a unit normal vector. The plane difference measure $D_1(\boldsymbol{p}_1, \boldsymbol{p}_2)$ is computed as shown in Equation 4.2. The d coefficients of the planes are normalized according to the object size determined by l_{avrg} which is the average distance of the input point cloud points from its centroid.

$$D_1(\boldsymbol{p}_1, \boldsymbol{p}_2) = \sqrt{(a_1 - a_2)^2 + (b_1 - b_2)^2 + (c_1 - c_2)^2 + (\frac{d_1 - d_2}{l_{avrg}})^2} \tag{4.2}$$

The two planes must have the same orientation before this computation is applied, i.e. $\boldsymbol{n}_{p_1}^T \boldsymbol{n}_{p_2} \geq 0$. Otherwise, one of the plane vectors ($\boldsymbol{p}_1$ or $\boldsymbol{p}_2$) must be multiplied by -1 before the computation. Also, as mentioned above, the planes must both be normalized so that $||\boldsymbol{n}_{p_1}|| = ||\boldsymbol{n}_{p_2}|| = 1$.

Plane difference measure D_2

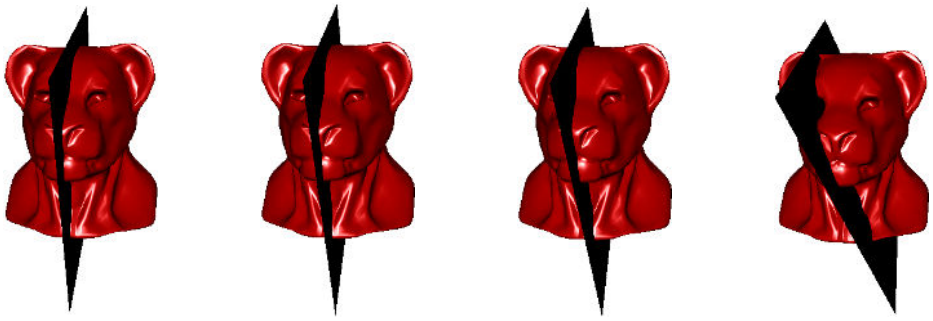
As the third plane difference measure, denoted $D_2(\boldsymbol{p}_1, \boldsymbol{p}_2)$, we use a normalized version of the distance metric for transformations proposed in [18]. Since reflection of an object over a given plane can be understood as a transformation, this distance metric is well adoptable for our case. The difference measure $D_2(\boldsymbol{p}_1, \boldsymbol{p}_2)$ is computed as shown in Equation 4.3.

$$D_2(\boldsymbol{p}_1, \boldsymbol{p}_2) = \frac{1}{l_{avrg}} \sqrt{\frac{1}{n} \sum_{i=1}^{n} ||\boldsymbol{r}(\boldsymbol{p}_1, \boldsymbol{x}_i) - \boldsymbol{r}(\boldsymbol{p}_2, \boldsymbol{x}_i)||^2} \tag{4.3}$$

This measure expresses the distance of the point cloud $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ reflected over the first plane from the same point cloud reflected over the second plane.

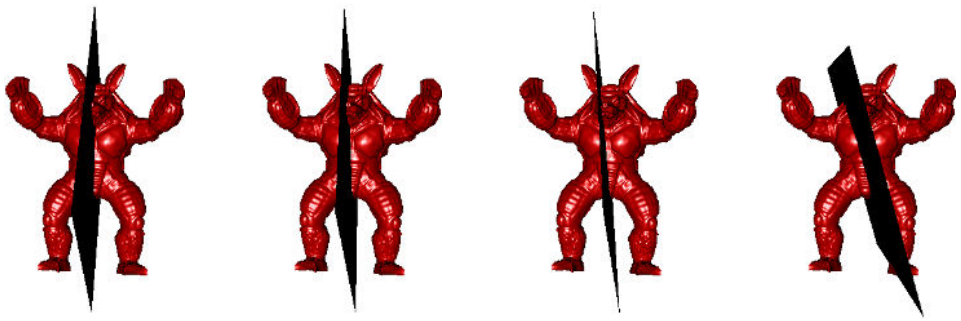
The minimal possible value of all the three plane difference measures $\gamma(\boldsymbol{p}_1, \boldsymbol{p}_2)$, $D_1(\boldsymbol{p}_1, \boldsymbol{p}_2)$ and $D_2(\boldsymbol{p}_1, \boldsymbol{p}_2)$ is 0, which only occurs when the planes P_1 and P_2, represented by $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, are the same. The maximum value of $\gamma(\boldsymbol{p}_1, \boldsymbol{p}_2)$ is 90°. The maximum value of $D_1(\boldsymbol{p}_1, \boldsymbol{p}_2)$ and $D_2(\boldsymbol{p}_1, \boldsymbol{p}_2)$ is technically unlimited but for planes that pass through the object, their maximum value is approximately 1, due to the normalization according to the object size.

For better understanding how the difference measures γ, D_1 and D_2 behave, Figure 4.5 shows differerent planes represented by $\boldsymbol{p}$ with the Lion PS object and for each of these planes the differences $\gamma(\boldsymbol{p}, \boldsymbol{p}_c)$, $D_1(\boldsymbol{p}, \boldsymbol{p}_c)$ and $D_2(\boldsymbol{p}, \boldsymbol{p}_c)$ are shown where $\boldsymbol{p}_c$ represents the correct symmetry plane of the Lion PS. Figure 4.6 shows the same differences for other planes with the Armadillo PS object.



(a)	(b)	(c)	(d)
$\gamma(\boldsymbol{p}, \boldsymbol{p}_c) = 0°$	$\gamma(\boldsymbol{p}, \boldsymbol{p}_c) = 2.882°$	$\gamma(\boldsymbol{p}, \boldsymbol{p}_c) = 6.566°$	$\gamma(\boldsymbol{p}, \boldsymbol{p}_c) = 21.270°$
$D_1(\boldsymbol{p}, \boldsymbol{p}_c) = 0$	$D_1(\boldsymbol{p}, \boldsymbol{p}_c) = 0.0597$	$D_1(\boldsymbol{p}, \boldsymbol{p}_c) = 0.1218$	$D_1(\boldsymbol{p}, \boldsymbol{p}_c) = 0.3813$
$D_2(\boldsymbol{p}, \boldsymbol{p}_c) = 0$	$D_2(\boldsymbol{p}, \boldsymbol{p}_c) = 0.0859$	$D_2(\boldsymbol{p}, \boldsymbol{p}_c) = 0.1937$	$D_2(\boldsymbol{p}, \boldsymbol{p}_c) = 0.6071$

Figure 4.5: Different planes with the Lion PS object and their differences γ, D_1 and D_2 from the correct symmetry plane of the Lion PS. The vector $\boldsymbol{p}$ represents the plane depicted in the given figure and $\boldsymbol{p}_c$ represents the correct symmetry plane of the Lion PS. The correct symmetry plane is also depicted in (a) for better visual comparison.



(a)	(b)	(c)	(d)
$\gamma(\boldsymbol{p}, \boldsymbol{p}_c) = 0°$	$\gamma(\boldsymbol{p}, \boldsymbol{p}_c) = 3.215°$	$\gamma(\boldsymbol{p}, \boldsymbol{p}_c) = 7.023°$	$\gamma(\boldsymbol{p}, \boldsymbol{p}_c) = 21.523°$
$D_1(\boldsymbol{p}, \boldsymbol{p}_c) = 0$	$D_1(\boldsymbol{p}, \boldsymbol{p}_c) = 0.0597$	$D_1(\boldsymbol{p}, \boldsymbol{p}_c) = 0.1305$	$D_1(\boldsymbol{p}, \boldsymbol{p}_c) = 0.4093$
$D_2(\boldsymbol{p}, \boldsymbol{p}_c) = 0$	$D_2(\boldsymbol{p}, \boldsymbol{p}_c) = 0.1068$	$D_2(\boldsymbol{p}, \boldsymbol{p}_c) = 0.2226$	$D_2(\boldsymbol{p}, \boldsymbol{p}_c) = 0.6490$

Figure 4.6: Different planes with the Armadillo PS object and their differences γ, D_1 and D_2 from the correct symmetry plane of the Armadillo PS. The vector $\boldsymbol{p}$ represents the plane depicted in the given figure and $\boldsymbol{p}_c$ represents the corrects symmetry plane of the Armadillo PS. The correct symmetry plane is also depicted in (a) for better visual comparison.

It can be seen that the plane difference measures D_1 and D_2 give considerably similar values when applied on the same two planes with the same object. This implies that neither one of these measures is more important than the other and that they both give us very similar information. But in

the rest of this chapter we will use both these difference measures for completeness. The difference measure γ, on the other hand, gives very different information than D_1 and D_2 because it measures the angle difference of the planes and does not consider their position.

Table 4.1 shows the a, b, c and d coefficients of the symmetry planes detected by our method on the perfectly symmetrical objects from Figure 4.4. For each object the correct symmetry plane is shown for comparison. All the planes are normalized (their normal vectors have unit lengths). The table also shows the differences between the corresponding coefficients of the correct and the detected plane denoted Δa, Δb, Δc and Δd. If we denote the normalized correct plane P_c, represented by $\boldsymbol{p}_c = [a_c, b_c, c_c, d_c]^T$, and the normalized detected plane P_d, represented by $\boldsymbol{p}_d = [a_d, b_d, c_d, d_d]^T$, the coefficient differences are computed as $\Delta a = |a_c - a_d|$, $\Delta b = |b_c - b_d|$, $\Delta c = |c_c - c_d|$ and $\Delta d = \frac{|d_c - d_c|}{l_{avrg}}$. The planes must have the same orientation, i.e. if $\boldsymbol{n}_{\boldsymbol{p}_c}^T \boldsymbol{n}_{\boldsymbol{p}_d} < 0$ then one of the plane vectors ($\boldsymbol{p}_c$ or $\boldsymbol{p}_d$) is multiplied by -1 before computing the coefficient differences.

It can be seen that the differences between the correct planes and the detected planes are very small. We do not show the detected planes in any figure because they differ from the correct planes so little that the difference is visually unnoticeable and the detected planes appear the same as the planes shown in Figure 4.4.

Object		a	b	c	d
Lion PS	Correct plane	0.19559	−0.71076	0.67569	17.60907
	Detected plane	0.19552	−0.71015	0.67635	17.63777
	Difference Δ	0.00007	0.00061	0.00066	0.00083
Ant PS	Correct plane	0.35990	0.60887	−0.70692	−15.63888
	Detected plane	0.35951	0.60896	−0.70704	−15.64079
	Difference Δ	0.00039	0.00009	0.00012	0.00028
Armadillo PS	Correct plane	−0.33494	−0.75954	0.55759	21.99788
	Detected plane	−0.33318	−0.76143	0.55606	22.00152
	Difference Δ	0.00176	0.00189	0.00153	0.00013
Embrasure PS	Correct plane	0.84358	0.22679	−0.48674	−165.15484
	Detected plane	0.84253	0.22662	−0.48865	−165.58402
	Difference Δ	0.00105	0.00017	0.00191	0.00154

Table 4.1: The a, b, c and d coefficients of the detected symmetry planes and the correct symmetry planes of the perfectly symmetrical objects. Also the differences Δa, Δb, Δc and Δd between the corresponding coefficients of the correct and the detected planes are shown. The planes are normalized.

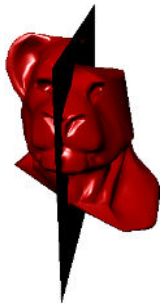
Table 4.2 shows the differences γ, D_1 and D_2 of the detected planes (represented by $\boldsymbol{p}_d$) from the correct symmetry planes (represented by $\boldsymbol{p}_c$) and the computation times of the detected planes for all the perfectly symmetrical objects from Figure 4.4. All three difference measures γ, D_1 and D_2 seem to be very small for all four objects, although for the Ant PS and the Lion PS they seem to be slightly smaller than for the Armadillo PS and the Embrasure PS. This is most likely caused by the larger point count of the Armadillo PS and the Embrasure PS because they suffer more damage by the simplification process performed at the beginning of our method. Since for the symmetry measure computation we simplify the input object to approximately 1000 points (see Section 3.5), the point count reduction is approximately 1:4 for the Ant PS, 1:2 for the Lion PS and over 1:100 for the Armadillo PS and the Embrasure PS. But even with such significant point count reduction the precision of the symmetry detection seems to be very high and does not differ so significantly from the precision achieved for the objects with a lower point count.

On the other hand, the simplification ensures that the computation time is very similar for all four objects even when their point counts differ significantly. This means that the computation time of our method is not very dependent on the input object's point count. The only part of our method whose computation time can be influenced by the point count is the simplification itself which is considerably fast.

Object	Points	$\gamma(\boldsymbol{p}_d, \boldsymbol{p}_c)$ [°]	$D_1(\boldsymbol{p}_d, \boldsymbol{p}_c)$	$D_2(\boldsymbol{p}_d, \boldsymbol{p}_c)$	Time [s]
Lion PS	2376	0.0516	0.00123	0.00142	16.9
Ant PS	3808	0.0240	0.00050	0.00102	21.9
Armadillo PS	108266	0.1718	0.00300	0.00404	19.9
Embrasure PS	138998	0.1251	0.00267	0.00402	21.9

Table 4.2: The differences $\gamma(\boldsymbol{p}_d, \boldsymbol{p}_c)$, $D_1(\boldsymbol{p}_d, \boldsymbol{p}_c)$ and $D_2(\boldsymbol{p}_d, \boldsymbol{p}_c)$ where $\boldsymbol{p}_d$ represents the detected symmetry plane and $\boldsymbol{p}_c$ represents the correct symmetry plane of a given perfectly symmetrical object. Also the computation time of each detected plane and the point count of each object are shown.

In order to quantify the precision of our method on objects with missing parts, we damaged the Lion PS and the Armadillo PS by removing some of their points. These two damaged objects are depicted in Figure 4.7 together with their correct symmetry planes which are exactly the same as the correct symmetry planes of their non-damaged versions.



(a) Lion PS damaged
(1577 points)

(b) Armadillo PS damaged
(73712 points)

Figure 4.7: Damaged versions of two perfectly symmetrical objects with their ground truth symmetry planes.

Table 4.3 shows the a, b, c and d coefficients of the symmetry planes detected by our method on the two damaged perfectly symmetrical objects from Figure 4.7 and on their non-damaged versions for comparison. For each object also the correct symmetry plane and the differences Δa, Δb, Δc and Δd between the correct and the detected planes are shown. All the planes are normalized. It can be seen that even on the damaged objects the detected symmetry planes differ very little from the correct symmetry planes. Even in this case the visual difference between the detected and the correct planes is so insignificant that showing the detected planes in a figure has no meaning since they appear the same as the planes depicted in Figure 4.7.

Table 4.4 shows the differences γ, D_1 and D_2 of the detected planes (represented by $\boldsymbol{p}_d$) from the correct symmetry planes (represented by $\boldsymbol{p}_c$) of the two damaged perfectly symmetrical objects from Figure 4.7 and their non-damaged versions for comparison. The table also shows the computation times of the detected planes. All three difference measures γ, D_1 and D_2 again seem to be very small for both objects. The differences γ and D_1 for the damaged version of the Armadillo PS are even lower than for its non-damaged version which may be caused by the lower point count of the damaged version. For the Lion PS all three difference measures are higher, but not very much, for its damaged version than for its non-damaged version. Overall the precision of our method seems to be very good even when the method is used on the damaged versions of the perfectly symmetrical objects which exhibit missing parts.

Object		a	b	c	d
Lion PS damaged	Correct pl.	0.19559	−0.71076	0.67569	17.60907
	Detected pl.	0.19708	−0.71018	0.67586	17.55938
	Difference Δ	0.00149	0.00058	0.00017	0.00154
Lion PS	Correct pl.	0.19559	−0.71076	0.67569	17.60907
	Detected pl.	0.19552	−0.71015	0.67635	17.63777
	Difference Δ	0.00007	0.00061	0.00066	0.00083
Armadillo PS damaged	Correct pl.	−0.33494	−0.75954	0.55759	21.99788
	Detected pl.	−0.33577	−0.76033	0.55600	21.96320
	Difference Δ	0.00083	0.00079	0.00159	0.00143
Armadillo PS	Correct pl.	−0.33494	−0.75954	0.55759	21.99788
	Detected pl.	−0.33318	−0.76143	0.55606	22.00152
	Difference Δ	0.00176	0.00189	0.00153	0.00013

Table 4.3: The a, b, c and d coefficients of the detected symmetry planes and the correct symmetry planes of both the damaged and non-damaged versions of Lion PS and Armadillo PS. Also the differences Δa, Δb, Δc and Δd between the corresponding coefficients of the correct and the detected plane are shown. The planes are normalized.

Object	Points	$\gamma(\boldsymbol{p}_d, \boldsymbol{p}_c)$ [°]	$D_1(\boldsymbol{p}_d, \boldsymbol{p}_c)$	$D_2(\boldsymbol{p}_d, \boldsymbol{p}_c)$	Time [s]
Lion PS damaged	1577	0.0918	0.00222	0.00476	17.5
Lion PS	2376	0.0516	0.00123	0.00142	16.9
Armadillo PS damaged	73712	0.1120	0.00242	0.00417	20.2
Armadillo PS	108266	0.1718	0.00300	0.00404	19.9

Table 4.4: The differences $\gamma(\boldsymbol{p}_d, \boldsymbol{p}_c)$, $D_1(\boldsymbol{p}_d, \boldsymbol{p}_c)$ and $D_2(\boldsymbol{p}_d, \boldsymbol{p}_c)$ where $\boldsymbol{p}_d$ represents the detected symmetry plane and $\boldsymbol{p}_c$ represents the correct symmetry plane of a given damaged or non-damaged version of a perfectly symmetrical object. Also the computation time of each detected plane and the point count of each object are shown.

We already mentioned that we chose the method by Sipiran et al. for comparison with our method. Unfortunately, the paper which presents this method [21] does not give any numerical evaluation of its precision. Therefore, we cannot compare our method with the method by Sipiran et al. numerically.

4.1.2 Approximately Symmetrical Objects

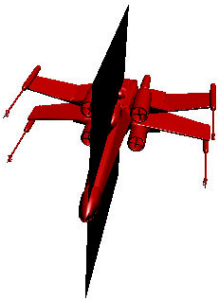
In this section we show results of our method on several objects which are not perfectly symmetrical and even exhibit missing parts. As already men-

tioned, there is no absolutely correct plane of symmetry of an approximately symmetrical object which means there is no ground truth symmetry plane to compare the detected plane with. Therefore, the results of the symmetry detection on such objects are presented visually showing that the detected planes truly capture the reflectional symmetry of the given objects. Most objects were randomly rotated before the symmetry detection to make sure our method also works on non-aligned objects.

First, we show the detected symmetry planes of the four strongly (but not perfectly) symmetrical artificial objects Lion, Ant, Starship and Formula (from Figure 4.1). The symmetry planes of these four objects detected by our method are depicted in Figure 4.8. It can be seen that the detected planes really capture the reflectional symmetry of these objects.

(a) Computation time: 15.7 s

(b) Computation time: 24.0 s



(c) Computation time: 19.3 s

(d) Computation time: 33.2 s

Figure 4.8: Symmetry planes detected by our method on the four strongly symmetrical objects (Lion - (a), Ant - (b), Starship - (c) and Formula - (d)). For each object also the computation time of the symmetry detection is shown.

We also created two damaged versions of the Ant and one damaged version of the Lion by clipping off some of their parts. The damaged objects together with their symmetry planes detected by our method are depicted in

Figure 4.9. Even on these damaged objects, which exhibit quite significant missing parts, the detected symmetry planes still seem to be visually correct and seem to capture the symmetry which remains in them.

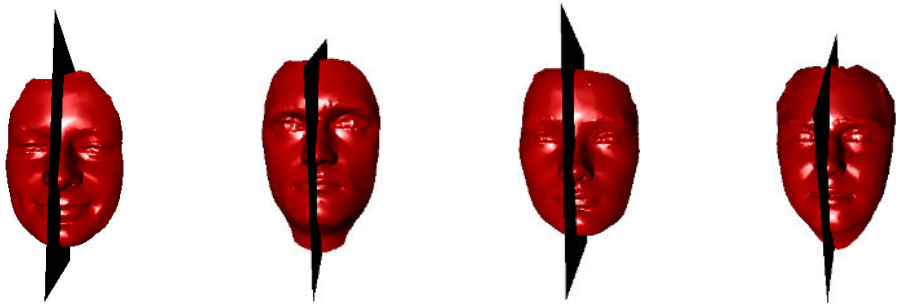
(a) 1952 points
Comp. time: 18.2 s

(b) 2945 points
Comp. time: 30.0 s

(c) 1284 points
Comp. time: 23.5 s

Figure 4.9: The symmetry planes detected by our method on the two damaged versions of the Ant and on the damaged version of the Lion. For each object also the computation time of the symmetry detection and the point count are shown.

To test our method on more realistic objects we used it to detect the symmetry planes of the four 3D-scanned human faces - Face 1, 2, 3 and 4 (from Figure 4.2). The symmetry planes of these objects detected by our method are depicted in Figure 4.10. The figure shows that the detected planes are visually correct and they capture the reflectional symmetry in the faces.



(a)
Comp. time: 16.6 s

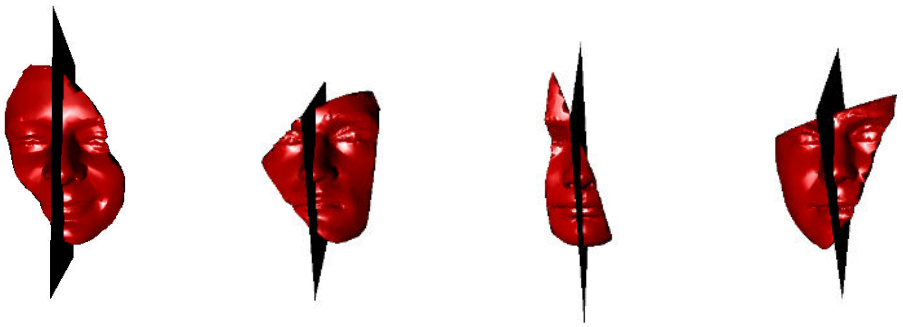
(b)
Comp. time: 19.9 s

(c)
Comp. time: 18.6 s

(d)
Comp. time: 15.4 s

Figure 4.10: Symmetry planes detected by our method on the four 3D-scanned faces (Face 1 - (a), Face 2 - (b), Face 3 - (c) and Face 4 - (d)). For each face also the computation time of the symmetry detection is shown.

We clipped the four faces to create their damaged versions and we used our method on them. The damaged faces and their symmetry planes detected by our method are depicted in Figure 4.11. The figure reveals that the detected planes for the damaged faces are visually the same as (or at least very similar to) the planes detected for their non-damaged versions. It can be noticed that in most of the damaged faces, especially the one shown in Figure 4.11c, quite little symmetry is preserved but our method is still able to detect it.



(a) 2224 points Comp. time: 18.3 s | (b) 1782 points Comp. time: 18.6 s | (c) 1401 points Comp. time: 28.8 s | (d) 1602 points Comp. time: 20.9 s

Figure 4.11: Symmetry planes detected by our method on the four damaged versions of the faces (Face 1 - (a), Face 2 - (b), Face 3 - (c) and damaged Face 4 - (d)). For each face also the computation time of the symmetry detection and the point count are shown.

Given the results shown in this section it can be stated that our method is capable of finding a visually correct plane of reflectional symmetry, not just in perfectly or strongly symmetrical objects, but also in approximately symmetrical objects and in objects which exhibit significant missing parts. Results of our method for some additional approximately symmetrical objects and objects with missing parts will be revealed in Section 4.1.4.

4.1.3 Objects with Noise

Apart from testing our method on objects with missing parts we tested it on objects which are damaged in a different way. We took the strongly symmetrical objects Lion, Ant, Starship and Formula (from Figure 4.1) and we added artificial noise to them. The noise was created by adding a random vector $[rand_x, rand_y, rand_z]^T \cdot l_{avrg} \cdot q$ to each point of the given point cloud, representing the input object, where $rand_x$, $rand_y$ and $rand_z$ are random values from $\langle -1; 1 \rangle$, l_{avrg} is the average distance of the point cloud points

from its centroid, and q is a constant which determines the noise magnitude. Figure 4.12 shows the symmetry planes detected by our method on the four objects with added noise with $q = 0.1$. Figure 4.13 shows the detected planes on the same objects with $q = 0.2$.

It can be seen that for $q = 0.1$ the noise is noticeable in the objects changing their shapes quite significantly. The symmetry planes detected by our method on these noisy shapes seem to be visually correct and not very much differing from the planes detected on the original shapes without the noise. For $q = 0.2$ the objects are damaged even more, being barely recognizable, but the symmetry planes detected on them still seem to be visually correct and very similar to those detected on the original shapes.

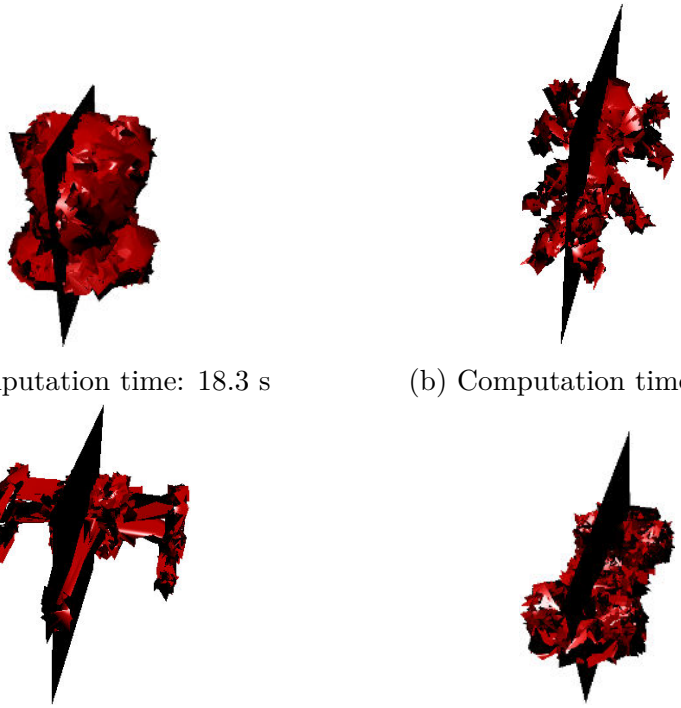
(a) Computation time: 19.8 s

(b) Computation time: 17.8 s

(c) Computation time: 17.5 s

(d) Computation time: 27.6 s

Figure 4.12: Symmetry planes detected by our method on the four strongly symmetrical objects with added noise with $q = 0.1$ (Lion - (a), Ant - (b), Starship - (c) and Formula - (d)). For each object also the computation time of the symmetry detection is shown.



(a) Computation time: 18.3 s

(b) Computation time: 14.5 s

(c) Computation time: 15.9 s

(d) Computation time: 19.5 s

Figure 4.13: Symmetry planes detected by our method on the four strongly symmetrical objects with added noise with $q = 0.2$ (Lion - (a), Ant - (b), Starship - (c) and Formula - (d)). For each object also the computation time of the symmetry detection is shown.

Table 4.5 shows the differences Δa, Δb, Δc and Δd (their computation is described in Section 4.1.1) between the coefficients of the detected planes and the coefficients of the correct symmetry planes for each noisy object with $q = 0.1$ and $q = 0.2$. As the correct symmetry planes we take the planes detected by our method on the original versions (without the noise) of the corresponding noisy objects. The correct planes are also shown in the table to see the context of the coefficient differences. For better orientation in the table, the coefficient values of the detected planes are not shown (otherwise the table would be quite chaotic), to evaluate the differences these values are not very important anyway.

Table 4.6 shows the differences γ, D_1 and D_2 of the planes detected on the noisy objects (represented by $\boldsymbol{p}_d$), with $q = 0.1$ and $q = 0.2$, from the correct symmetry planes detected on the corresponding original versions (without the noise) of these objects (represented by $\boldsymbol{p}_c$).

Object		a	b	c	d
Lion	Correct plane	0.19433	−0.71064	0.67615	17.62341
	Difference Δ ($q = 0.1$)	0.00522	0.00494	0.00676	0.00106
	Difference Δ ($q = 0.2$)	0.01646	0.01579	0.01139	0.01318
Ant	Correct plane	0.35859	0.60929	−0.70722	−15.63798
	Difference Δ ($q = 0.1$)	0.00056	0.00068	0.00088	0.00284
	Difference Δ ($q = 0.2$)	0.02145	0.00193	0.01299	0.01193
Starship	Correct plane	−0.99999	0.00030	−0.00346	0.01594
	Difference Δ ($q = 0.1$)	0.00000	0.00225	0.00047	0.00450
	Difference Δ ($q = 0.2$)	0.00000	0.00332	0.00303	0.00794
Formula	Correct plane	0.91000	−0.30611	0.27962	4.25174
	Difference Δ ($q = 0.1$)	0.00465	0.01269	0.00157	0.01048
	Difference Δ ($q = 0.2$)	0.01385	0.02702	0.01771	0.02825

Table 4.5: The Δa, Δb, Δc and Δd differences between the coefficients of the planes detected on the noisy objects and the coefficients of the planes (the correct planes) detected on their original versions (without the noise).

Object	q	$\gamma(\boldsymbol{p}_d, \boldsymbol{p}_c)$ [°]	$D_1(\boldsymbol{p}_d, \boldsymbol{p}_c)$	$D_2(\boldsymbol{p}_d, \boldsymbol{p}_c)$
Lion	0.1	0.5658	0.00993	0.02260
	0.2	1.4611	0.02872	0.03990
Ant	0.1	0.0718	0.00311	0.00543
	0.2	1.4417	0.02786	0.04306
Starship	0.1	0.1321	0.00506	0.00645
	0.2	0.2580	0.00914	0.01135
Formula	0.1	0.7800	0.01718	0.01267
	0.2	2.0144	0.04511	0.04795

Table 4.6: The differences $\gamma(\boldsymbol{p}_d, \boldsymbol{p}_c)$, $D_1(\boldsymbol{p}_d, \boldsymbol{p}_c)$ and $D_2(\boldsymbol{p}_d, \boldsymbol{p}_c)$ where $\boldsymbol{p}_d$ represents the detected symmetry plane of the given noisy object and $\boldsymbol{p}_c$ represents the correct symmetry plane detected on the corresponding original version (without the noise) of the object.

The tables reveal that for both $q = 0.1$ and $q = 0.2$ the differences between the correct symmetry planes (the planes detected on the original objects without the noise) and the planes detected on the noisy objects are quite small. For example, the angle $\gamma(\boldsymbol{p}_d, \boldsymbol{p}_c)$ is below 1° for $q = 0.1$ for all four objects, and barely reaches 2° for $q = 0.2$, which is very low considering how damaged the objects are by the noise. It is interesting that the values $\gamma(\boldsymbol{p}_d, \boldsymbol{p}_c)$, $D_1(\boldsymbol{p}_d, \boldsymbol{p}_c)$ and $D_2(\boldsymbol{p}_d, \boldsymbol{p}_c)$ are quite similar for $q = 0.1$ and $q = 0.2$ for all objects except for the Ant where the values are very low for $q = 0.1$ and increase significantly when q changes to 0.2.

Overall, given the results presented in this section, it seems our method performs well even on objects that contain noise. Even in the case the noise is very significant and damages the object noticeably our method is still able to detect a visually plausible symmetry plane.

4.1.4 Comparison with Sipiran et al.

In this section we provide a comparison of our method with the method by Sipiran et al. [21] (see Section 2.11). We chose the method by Sipiran et al. for comparison because it seems to be the only one, from the methods described in Chapter 2, which is specifically designed to work on objects with significant missing parts which was one of the major requirements for our method. Also it seems to work very well and, judging by what is shown in the paper, it gives very good results.

We will provide results of the method by Sipiran et al. for several objects, where these results were gathered from the paper [21], and we will show the results of our method for the same or very similar objects. At the end of this section we will also discuss the advantages and disadvantages of the method in comparison with our method.

Figure 4.14 shows the symmetry planes detected using the method by Sipiran et al. on four differently damaged versions of the Armadillo. Symmetry plane detected on another very significantly damaged version of the Armadillo is shown in Figure 4.15, in this case there are missing parts on both sides of the object.

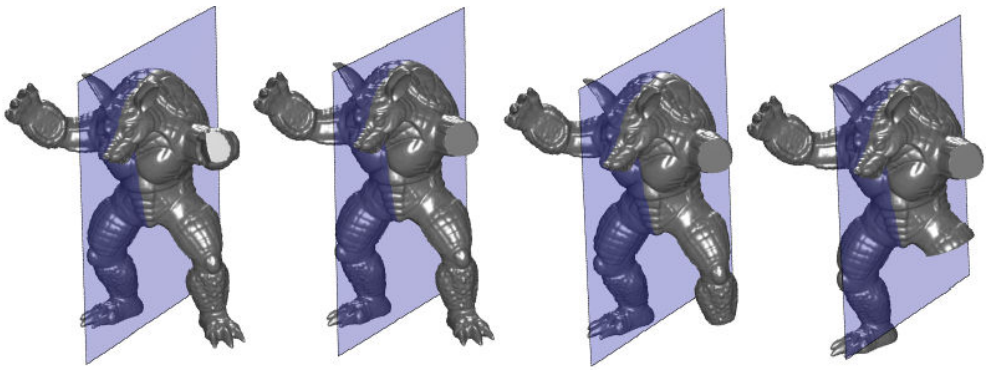


Figure 4.14: The symmetry planes detected on four differently damaged versions of the Armadillo using the method by Sipiran et al. (Figure taken from [21].)

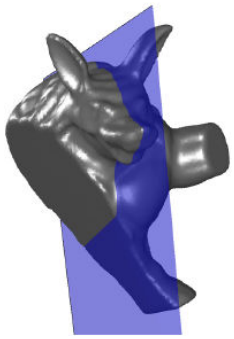
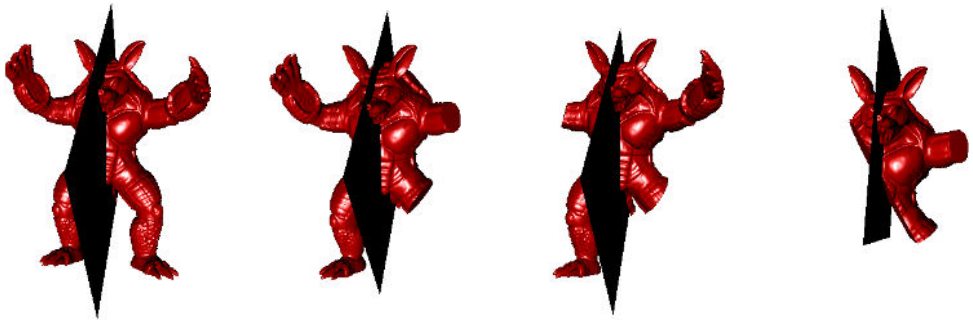


Figure 4.15: The symmetry plane detected on another very significantly damaged version of the Armadillo using the method by Sipiran et al. (Figure taken from [21].)

It can be seen that the symmetry planes detected using the method by Sipiran et al. are visually correct for all the damaged versions of the Armadillo. Figure 4.16 depicts several Armadillos, which are damaged in a similar manner, and the non-damaged Armadillo, together with their symmetry planes detected using our method.



(a) 172974 points Comp. time: 22.3 s | (b) 135601 points Comp. time: 19.4 s | (c) 134989 points Comp. time: 16.8 s | (d) 64647 points Comp. time: 20.5 s

Figure 4.16: Symmetry planes detected by our method on the Armadillo and on its damaged versions. For each Armadillo also the computation time of the symmetry detection and the point count are shown.

The symmetry planes detected by our method on the Armadillos all seem to be visually correct and the Armadillos exhibit very similar levels of missing parts as the Armadillos used by Sipiran et al. for testing their method (see Figures 4.14 and 4.15).

Another object used by Sipiran et al. is the 3D-scanned Embrasure. This object is interesting because it is also damaged and exhibits missing parts but it was not damaged artificially, the object was already broken in

a natural way when it was scanned. The results of the symmetry detection on the Embrasure using the method by Sipiran et al. are depicted in Figure 4.17. The figure actually shows the three best symmetry planes detected on this object. We can now see one advantage of the method by Sipiran et al. which is that it can be used to detect more then one symmetry plane in the case the input object is symmetrical with respect to more planes. Our method, as it is now, is only capable of detecting the one most significant symmetry plane. Extending our method for detecting more symmetry planes will be part of the future work and will be further discussed in Chapter 5. Such an extension should not be very difficult to implement since it is just a matter of selecting the right planes from the set of the candidate planes (see Section 3.4) instead of selecting just the best one.

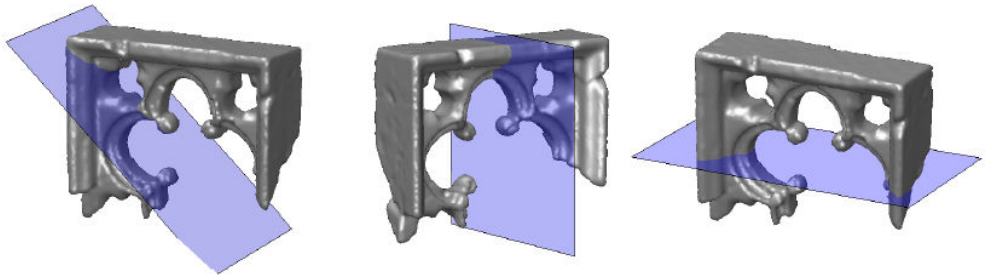


Figure 4.17: The three best symmetry planes detected on the Embrasure using the method by Sipiran et al. (Figure taken from [21].)

The visually most intuitive symmetry plane of the Embrasure is probably the one shown in the middle of the figure which appears the same as the one detected by our method as can be seen in Figure 4.18.

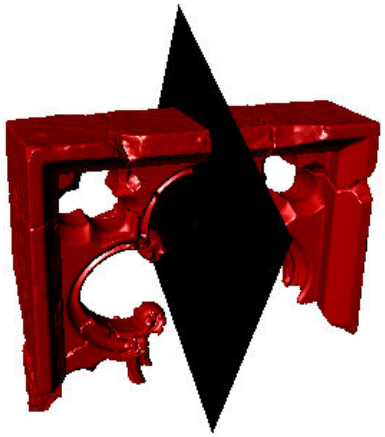


Figure 4.18: The symmetry plane detected by our method on the Embrasure. Computation time: 18.2 s.

The last object that we used for comparison with the method by Sipiran et al. is the 3D-scanned Column base. This object exhibits only a low level of missing parts but it was damaged naturally as well as the Embrasure. The two best symmetry planes detected on the Column base using the method by Sipiran et al. are depicted in Figure 4.19. In this case the symmetry plane on the right of the figure is probably the more intuitive one. Our method detects visually the same symmetry plane, see Figure 4.20.

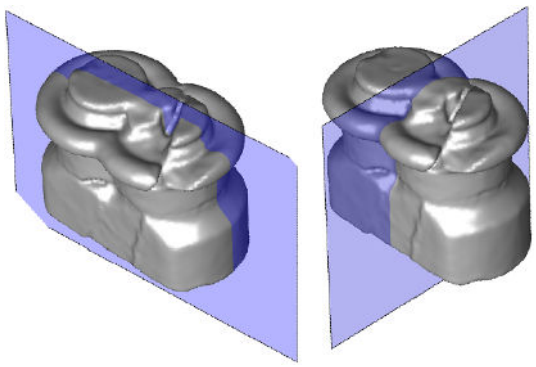


Figure 4.19: The two best symmetry planes detected on the Column base using the method by Sipiran et al. (Figure taken from [21].)

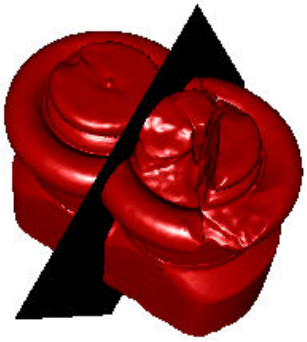


Figure 4.20: The symmetry plane detected by our method on the Column base. Computation time: 18.3 s.

It can be seen that our method can successfully detect visually correct symmetry planes on the same or very similar objects, with the same or similar levels of missing parts, as some of the objects on which the method by Sipiran et al. was tested. This shows that our method is capable of handling the same type of input objects with the same or similar level of missing parts as the method by Sipiran et al. One of the disadvantages of our method, which is the inability to detect more than one symmetry plane, was already mentioned above and could be quite easily removed by a change

in the implementation. Another small disadvantage can be the fact that our method is not rotationally invariant due to the grid-based simplification (see Sections 3.3, 3.4 and 3.5). But, since we successfully tested our method on various randomly rotated objects, this does not seem to pose a problem.

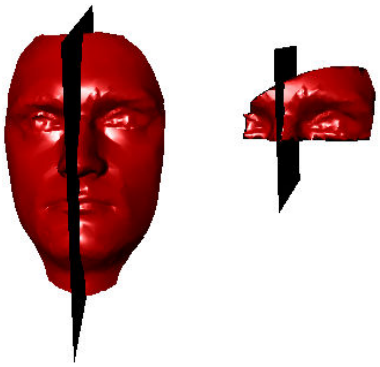
On the other side, our method seems to have several advantages. Probably the biggest advantage of our method is that it works on point clouds while the method by Sipiran et al. only works on manifold triangle meshes which is quite a constraint. Representation of the input object by a triangle mesh is not always available, let alone representation by a manifold mesh. Another advantage of our method is its robustness to noise which was demonstrated in Section 4.1.3. Although we do not have any results for comparison, we can assume that in this criterion our method would outperform the method by Sipiran et al. because it relies on detection of features (see Section 2.11) and the features get damaged by the noise. The feature-based symmetry detection also implies, and the authors even mention it in their paper, that the method by Sipiran et al. does not work on featureless objects such as very smooth shapes. Our method, on the other side, does not need to use any features (but it can if they are available - see below and Section 3.7) and overall puts virtually no constraints on the input object apart from that it must be represented by a set of points. The last advantage of our method is its extensibility granted by the weights in the symmetry measure (see Section 3.2, Equation 3.7). If there is more information about the input object than just the point positions, this information can be used to set the weights and improve the performance of our method. Until now we have been considering the basic method where the weights are not used (they all equal to 1) but in Section 4.2 we will show a case where the weights can be useful.

4.2 Results of the Modified Method

Curvatures in general seem to be very good for detecting local features of 3D models representing human faces or human heads [11] where the local features mostly represent the eyes, the ears, the nose or the mouth. Therefore, in the following text, we will show that the modified version of our method, which uses the Gaussian curvature similarity and the normal vector symmetry to set the weights w_{ij} in the symmetry measure (see Section 3.7, especially 3.7.1 and 3.7.2), can be used to detect the symmetry plane of very small parts of human faces as long as at least some local features are preserved. This modified version also uses the candidate plane filtering

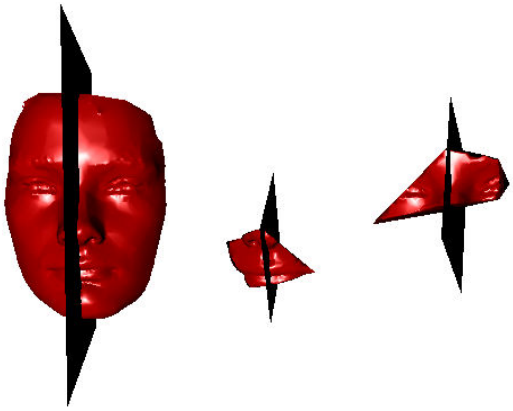
described in Section 3.7.3.

Figures 4.21 and 4.22 show the symmetry planes detected on (by clipping) heavily damaged versions of two of the scanned human faces (from Figure 4.2). Each figure also contains the corresponding original, non-damaged face with its symmetry plane, detected by the basic method, for comparison.



(a) (b)

Figure 4.21: Small part of Face 2 with its symmetry plane detected using the modified version of our method - (b) (980 points, computation time: 9.1 s) and the original Face 2 with its symmetry plane detected using the basic method shown for comparison - (a).



(a) (b) (c)

Figure 4.22: Two small parts of Face 3 with their symmetry planes detected using the modified version of our method - (b) (1026 points, computation time: 9.4 s) and (c) (803 points, computation time: 8.3 s) and the original Face 3 with its symmetry plane detected using the basic method shown for comparison - (a).

In all three cases the symmetry planes seem to be detected correctly despite the fact that the input objects contain very little symmetry information. This is because the symmetry detection was done using the similarity of local features represented by the Gaussian curvatures and the features still seem to be symmetrical even when the whole objects are not. On different types of objects, other than human faces, different local feature representations could probably be used to achieve similar results and on some object types maybe even the Gaussian curvature would work the same way, but we have not tested it. Also the normal vector symmetry weighting helps to make the symmetry detection more accurate.

It can be noticed that the computation time of the modified method is lower than that of the basic method, approximately by 50 %, despite the fact that computing the weights adds overhead to the symmetry measure computation and slows it down. This is due to the candidate plane filtering (see Section 3.7.3) which ensures that only several candidate planes are considered when selecting the best candidate which means the symmetry measure is computed much fewer times.

When we used the basic method on these three objects, it failed completely in all three cases because, when ignoring the local features, the objects are really not symmetrical at all and therefore, there is no way the basic method could detect a plausible symmetry plane. This suggests that the weights in the symmetry measure can be very useful in some cases. Designing more ways to set the weights and finding more uses for them will be a matter of future research.

4.3 Summary

In this chapter we have shown that our symmetry plane detection method works very accurately on perfectly symmetrical objects (see Section 4.1.1), is also able to detect a visually plausible symmetry plane of an approximately symmetrical object and is robust to missing parts (see Sections 4.1.2 and 4.1.4) and even to noise (see Section 4.1.3). Furthermore, our method works on point clouds which means it has basically no requirements on the input data, it only needs the point positions. But when more information about the input object is available, our method can take advantage of it and, in some cases, can use it to improve its performance (see Section 4.2).

Also, the running time of our method is acceptable, although there are methods which seem to be faster, at least in the case of objects with lower point count (thousands of points), namely [10] (see Section 2.12), [22] (see

Section 2.13), [12] (see Section 2.14) and [19] (see Section 2.15), whose running times are mostly in the order of few seconds or even under one second. But our method seems to outperform all these methods in the robustness and puts less requirements on the input data than most of these methods. Furthermore, the running times of these methods usually depend on the point count of the input object while the running time of our method almost does not depend on it. This means that for objects with higher point count our method could actually be faster.

5 Generalizations

The basic scheme of our method seems to be quite general which offers a possibility to generalize the whole method to extend its usability. In this chapter we give several possible ways how the method could be generalized or modified. It should be noted that these generalizations have not been implemented and therefore, everything described in this chapter is purely theoretical.

5.1 Detecting More Planes of Symmetry

As was already mentioned above, there are objects that are more or less symmetrical with respect to more than one plane, e.g. a cube. Our method, as it is now, only detects one plane of symmetry, specifically the most significant one, but it could possibly be extended to detect more than just one plane. Such an extension could be done by selecting more than just one best candidate plane in the best candidate selection step of our method (see Section 3.5).

The problem is that when more candidates are selected, all of them, or some of them, can be very close to each other resulting in several symmetry planes that are either exactly the same or very similar. Therefore, some metric for measuring plane distance, possibly one of those defined in Section 4.1.1, needs to be used and only planes with certain mutual distance need to be selected. One possible way to do this efficiently is to use some clustering algorithm in the plane space and allow to only select one candidate from each cluster.

5.2 Detecting Symmetries of Different Types

If we replace the function $\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{x})$, which reflects a point $\boldsymbol{x}$ over a plane P represented by $\boldsymbol{p}$ (see Equation 3.1), with a function that performs a different transformation, the symmetry measure s_X (see Equation 3.7) can possibly be used to detect symmetries of different types. The symmetry measure is not the only thing that needs to be modified, we also need to adjust the candidate creation (see Section 3.4) according to the given symmetry type, which for many symmetry types is not a simple task.

5.2.1 Reflectional Symmetry w.r.t. a Point

Reflection of a point $\boldsymbol{x} \in E^3$ over a point $\boldsymbol{q} \in E^3$ can be defined as translating $\boldsymbol{x}$ to the other side of $\boldsymbol{q}$ to the same distance from $\boldsymbol{q}$ as it was. A function $\boldsymbol{r}(\boldsymbol{q}, \boldsymbol{x}) \in E^3$ that reflects a point $\boldsymbol{x}$ over a point $\boldsymbol{q}$ can therefore be defined as shown in Equation 5.1.

$$\boldsymbol{r}(\boldsymbol{q}, \boldsymbol{x}) = \boldsymbol{x} + 2(\boldsymbol{q} - \boldsymbol{x}) = 2\boldsymbol{q} - \boldsymbol{x} \tag{5.1}$$

An object X is reflectionally symmetrical (perfectly) with respect to a point $\boldsymbol{q}$ when for every point $\boldsymbol{x}_i \in X$ there is a point $\boldsymbol{x}_j \in X$ for which it is that $\boldsymbol{r}(\boldsymbol{q}, \boldsymbol{x}_i) = \boldsymbol{x}_j$. This definition is the same for all types of reflectional symmetry, only the function $\boldsymbol{r}$ changes, so we will not repeat it for the other types of reflectional symmetry.

The point of reflectional symmetry of a given 3D object could possibly be found using our method after only a slight modification. Obviously, the function $\boldsymbol{r}$ must be replaced in the symmetry measure s_X and the candidate creation needs to be modified to create candidate points instead of planes. A candidate point of symmetry can be simply created by two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as $\frac{\boldsymbol{x}_i + \boldsymbol{x}_j}{2}$. The rest of the method could remain the same.

5.2.2 Reflectional Symmetry w.r.t. a Line

Let us define a general line Q by its parametric equation as $Q : \boldsymbol{q}(t) = \boldsymbol{q}_s + \boldsymbol{q}_d t, t \in \mathbb{R}$ where $\boldsymbol{q}_s$ is some point on the line and $\boldsymbol{q}_d$ is its direction vector. The function $\boldsymbol{r}(Q, \boldsymbol{x}) \in E^3$, which reflects a general point $\boldsymbol{x} \in E^3$ over the line Q, can be defined as shown in Equation 5.2.

$$\boldsymbol{r}(Q, \boldsymbol{x}) = \boldsymbol{x} + 2(\boldsymbol{q}(t_r) - \boldsymbol{x}) = 2\boldsymbol{q}(t_r) - \boldsymbol{x} \tag{5.2}$$

The value t_r is such a value of the parameter t for which the vector $\boldsymbol{q}(t) - \boldsymbol{x}$ is perpendicular to the line Q. Such a value of t can be found by solving Equation 5.3 for t.

$$(\boldsymbol{q}(t) - \boldsymbol{x})^T \boldsymbol{q}_d = 0 \tag{5.3}$$

This equation can be rewritten as shown in Equations 5.4 and 5.5. Its solution is shown in Equation 5.6

$$(\boldsymbol{q}_s + \boldsymbol{q}_d t - \boldsymbol{x})^T \boldsymbol{q}_d = 0 \tag{5.4}$$

$$\boldsymbol{q}_s^T \boldsymbol{q}_d + \boldsymbol{q}_d^T \boldsymbol{q}_d t - \boldsymbol{x}^T \boldsymbol{q}_d = 0 \tag{5.5}$$

$$t = \frac{\boldsymbol{x}^T \boldsymbol{q}_d - \boldsymbol{q}_s^T \boldsymbol{q}_d}{\boldsymbol{q}_d^T \boldsymbol{q}_d} \tag{5.6}$$

We can now set t_r as this value of t. The reflection function $\boldsymbol{r}(Q, \boldsymbol{x})$ can therefore be defined as shown in Equation 5.7.

$$\boldsymbol{r}(Q, \boldsymbol{x}) = 2\boldsymbol{q}(\frac{\boldsymbol{x}^T \boldsymbol{q}_d - \boldsymbol{q}_s^T \boldsymbol{q}_d}{\boldsymbol{q}_d^T \boldsymbol{q}_d}) - \boldsymbol{x} = 2(\boldsymbol{q}_s + \boldsymbol{q}_d \frac{\boldsymbol{x}^T \boldsymbol{q}_d - \boldsymbol{q}_s^T \boldsymbol{q}_d}{\boldsymbol{q}_d^T \boldsymbol{q}_d}) - \boldsymbol{x} \tag{5.7}$$

To detect a line of reflectional symmetry of a given 3D object we also need to change the candidate creation, but this time it is not as simple as for the point symmetry. A line in E^3 cannot be defined as a line of reflectional symmetry of two points because there is an additional degree of freedom which is the rotation around the axis on which the two points lie. This problem could possibly be resolved by taking a whole neighborhood of one of the two points and trying to find such a line for which, when this neighborhood is reflected over the line, it best fits onto the neighborhood of the second point.

5.2.3 Reflectional Symmetry w.r.t. a Curve

Let us now suppose Q is not a line but a general curve in space whose parametric equation is a general vector function $\boldsymbol{q}(t) = [q_x(t), q_y(t), q_z(t)]^T, t \in \mathbb{R}$. The function $\boldsymbol{r}(Q, \boldsymbol{x})$, which reflects the point $\boldsymbol{x} \in E^3$ over the curve Q, can be defined in the same way as for the reflection over the line as $\boldsymbol{r}(Q, \boldsymbol{x}) = 2\boldsymbol{q}(t_r) - \boldsymbol{x}$ (see Equation 5.2) where t_r is such a value of t for which the vector $\boldsymbol{q}(t) - \boldsymbol{x}$ is perpendicular to the tangent vector of the curve in the point $\boldsymbol{q}(t)$. If there are multiple values of t that satisfy this condition, the one for which the distance between $\boldsymbol{q}(t)$ and $\boldsymbol{x}$ is the smallest is used as t_r. The tangent vector of the curve Q in the point $\boldsymbol{q}(t)$ is defined as $[\frac{d}{dt}q_x(t), \frac{d}{dt}q_y(t), \frac{d}{dt}q_z(t)]^T$. The value of t_r can therefore be obtained by solving Equation 5.8 for t.

$$(\boldsymbol{q}(t) - \boldsymbol{x})^T [\frac{d}{dt}q_x(t), \frac{d}{dt}q_y(t), \frac{d}{dt}q_z(t)]^T = 0 \tag{5.8}$$

The concrete solution, and whether it can be obtained analytically, depends on the type of the function $\boldsymbol{q}(t)$ which defines the curve Q.

Creating the candidate curves, for the symmetry curve detection, would be even more problematic than creating the candidate lines. In this case, probably some more global approach would have to be used and the curves would have to be created using the whole object's shape, not just two of its points.

5.2.4 Reflectional Symmetry w.r.t. a Surface

If a point in E^3 can be reflected over a plane or over a curve, there is no reason why it could not be reflected over a 3D surface. We can define a surface Q by its parametric equation as $Q : \boldsymbol{q}(u,v) = [q_x(u,v), q_y(u,v), q_z(u,v)]^T, u \in \mathbb{R}, v \in \mathbb{R}$. The function $\boldsymbol{r}(Q, \boldsymbol{x}) \in E^3$, which reflects a point $\boldsymbol{x} \in E^3$ over the surface Q, can be defined in a similar way as for the curve, as shown in Equation 5.9.

$$\boldsymbol{r}(Q, \boldsymbol{x}) = \boldsymbol{x} + 2(\boldsymbol{q}(u_r, v_r) - \boldsymbol{x}) = 2\boldsymbol{q}(u_r, v_r) - \boldsymbol{x} \tag{5.9}$$

The values u_r and v_r are such values of the parameters u and v for which the vector $\boldsymbol{q}(u,v) - \boldsymbol{x}$ is perpendicular to the tangent plane of the surface in the point $\boldsymbol{q}(u,v)$. If there are multiple values of u and v that satisfy this condition, those for which the distance between $\boldsymbol{q}(u,v)$ and $\boldsymbol{x}$ is the smallest are used. The values of u_r and v_r can be obtained by solving the system of two equations shown in Equation 5.10 for u and v.

$$\begin{aligned}(\boldsymbol{q}(u,v) - \boldsymbol{x})^T [\frac{\partial}{\partial u} q_x(u,v), \frac{\partial}{\partial u} q_y(u,v), \frac{\partial}{\partial u} q_z(u,v)]^T = 0 \\ (\boldsymbol{q}(u,v) - \boldsymbol{x})^T [\frac{\partial}{\partial v} q_x(u,v), \frac{\partial}{\partial v} q_y(u,v), \frac{\partial}{\partial v} q_z(u,v)]^T = 0\end{aligned} \tag{5.10}$$

The concrete solution again depends on the type of the function $\boldsymbol{q}(u,v)$ which defines the surface Q.

Creation of the candidate surfaces for the detection of the symmetry surface would be as problematic as, or even more than, creation of the candidate curves for the symmetry curve detection. Again, some global approach would probably have to be used.

5.2.5 Rotational Symmetry

Let us now replace the reflection function $\boldsymbol{r}$ with function $\boldsymbol{rot}(Q, \beta, \boldsymbol{x})$ that preforms a rotation of a point $\boldsymbol{x} \in E^3$ around an axis Q by an angle β. We can define rotational symmetry in a very similar way as the reflectional symmetry. We can say an object X is rotationally symmetrical (perfectly) with respect to an axis Q when such an angle $\beta \neq k \cdot 360°, k \in \mathbb{Z}$ exists that for every point $\boldsymbol{x}_i \in X$ there is a point $\boldsymbol{x}_j \in X$ for which it is that $\boldsymbol{rot}(Q, \beta, \boldsymbol{x}_i) = \boldsymbol{x}_j$. In order to detect such a symmetry, we need to find the axis Q and also the angle β, which could possibly be done using our method but the candidate creation would again need to be modified.

Instead of candidate planes we would now need to create candidate axes together with candidate rotation angles. This could possibly be done using

the same approach as suggested for the candidate line creation in the reflectional symmetry line detection (see Section 5.2.2). Meaning, for each pair of points we could take the neighborhood of one of the two points and try to find such a rotation after which it best fits onto the neighborhood of the second point.

5.2.6 General Affine Symmetry

We can further generalize the symmetry by replacing the $\boldsymbol{rot}$ function with a function $\boldsymbol{trans}(\boldsymbol{T}, \boldsymbol{t}, \boldsymbol{x})$ which applies a general affine transformation on a given point $\boldsymbol{x} \in E^3$. Such a transformation is defined by a transformation matrix $\boldsymbol{T}$ and a translation vector $\boldsymbol{t}$. The $\boldsymbol{trans}$ function is then defined as shown in Equation 5.11.

$$\boldsymbol{trans}(\boldsymbol{T}, \boldsymbol{t}, \boldsymbol{x}) = \boldsymbol{T}\boldsymbol{x} + \boldsymbol{t} \tag{5.11}$$

We can say an object X has an affine symmetry (perfect) when such a matrix $\boldsymbol{T}$ and such a vector $\boldsymbol{t}$ exist that for every point $\boldsymbol{x}_i \in X$ there is a point $\boldsymbol{x}_j \in X$ for which it is that $\boldsymbol{trans}(\boldsymbol{T}, \boldsymbol{t}, \boldsymbol{x}_i) = \boldsymbol{x}_j$, excluding the case when $\boldsymbol{T} = \boldsymbol{I}$ and $\boldsymbol{t} = [0, 0, 0]^T$ where $\boldsymbol{I}$ is an identity matrix.

To detect such a symmetry we again need to change the candidate creation step to create general candidate affine transformations. This could possibly be done using the same approach as suggested in the rotational symmetry detection (see Section 5.2.5), only creating general affine transformations instead of rotation transformations.

5.3 Registration

We can consider symmetry detection as a special case of registration, when we try to find such a transformation of a given type which, after applied, best fits an object onto itself. Let us now suppose we have two objects represented by sets of points $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ and $Y = \{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_m\}$ and we want to find such a transformation that, after applied on X, best fits X onto Y. Of course, the best fit can only include some of the points of X and Y, not all of them. In general, we need to define which transformations we allow, for example, we can only search for reflection transformations or rigid transformations, but for the purpose of generality we will now consider general affine transformations. We can now modify the symmetry measure s_X (see Equation 3.7) as shown in Equation 5.12.

$$s_{X,Y}(\boldsymbol{T}, \boldsymbol{t}) = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} \varphi(||\boldsymbol{trans}(\boldsymbol{T}, \boldsymbol{t}, \boldsymbol{x}_i) - \boldsymbol{y}_j||) \tag{5.12}$$

Using the function $s_{X,Y}$ we could possibly find the transformation, defined by $\boldsymbol{T}$ and $\boldsymbol{t}$, that best fits X onto Y in the same way as we use the symmetry measure s_X for symmetry detection. Creation of the candidate transformations would not be any different from how it would be defined for the affine symmetry detection (see Section 5.2.6) only for each pair of points, one point would be from X and the other from Y.

6 Conclusion

In this thesis a new method for symmetry plane detection on 3D objects represented by point clouds was described (see Chapter 3) and its results were presented. The method seems to work very well on perfectly as well as on approximately symmetrical objects and exhibits good results even when used on objects with quite significant missing parts (see Chapter 4). It also showed to be robust to quite noticeable noise.

Our method was also compared to another symmetry plane detection method, specifically the method designed by Sipiran et al. [21]. It was shown that our method can handle the same or very similar objects as the method by Sipiran et al. and the advantages of our method in comparison with this method were pointed out (see Section 4.1.4). The new method is also easily extensible by additional information, if it is available, and it was shown that this extensibility can be very useful in some cases (see Section 4.2).

In addition, a way to make the method detect more than one plane of a given object was suggested and several ways to generalize the method for detection of symmetries of different types were suggested as well (see Chapter 5). In the future, we would like to deeper examine these possibilities to further extend or generalize the method.

Furthermore, since the symmetry measure used by our method (see Section 3.2) is differentiable, we can test various optimization methods for the last step of our method (see Section 3.6) and find such that converges to the global maximum more quickly or even from larger distance, possibly allowing to use a lower number of candidate planes (see Section 3.4).

Bibliography

[1] PRESIOUS 3D Data Sets. `http://presious.eu/resources/3d-data-sets`, 2013. Accessed: 2017-10-21.

[2] Fidentis Project. `https://www.fidentis.cz`, 2012. Accessed: 2017-10-21.

[3] Microsoft Solver Foundation. `https://msdn.microsoft.com/en-us/library/ff524497(v=vs.93).aspx`. Accessed: 2017-10-09.

[4] The Princeton Shape Benchmark. `http://shape.cs.princeton.edu/benchmark/`. Accessed: 2017-10-20.

[5] The Stanford 3D Scanning Repository. `http://graphics.stanford.edu/data/3Dscanrep/`. Accessed: 2017-10-20.

[6] Besl, P. J. – McKay, N. D. – others. A method for registration of 3-D shapes. *IEEE Transactions on pattern analysis and machine intelligence*. 1992, 14, 2, s. 239–256.

[7] Combès, B. et al. Automatic symmetry plane estimation of bilateral objects in point clouds. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, s. 1–8. IEEE, 2008.

[8] Hruda, L. – Dvořák, J. Estimating Approximate Plane of Symmetry of 3D Triangle Meshes. In *Proc. Central European Seminar on Computer Graphics*, Smolenice, Slovakia, 2017.

[9] Kakarala, R. – Kaliamoorthi, P. – Premachandran, V. Three-dimensional bilateral symmetry plane estimation in the phase domain. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, s. 249–256, 2013.

[10] Korman, S. et al. Probably approximately symmetric: Fast rigid symmetry detection with global guarantees. In *Computer Graphics Forum*, 34, s. 2–13. Wiley Online Library, 2015.

[11] Kubásková, K. Recognition of Important Features of Triangulated Human Head Models. In *Proc. Central European Seminar on Computer Graphics*, Smolenice, Slovakia, 2016.

[12] Li, B. et al. Efficient 3D reflection symmetry detection: A view-based approach. *Graphical Models*. 2016, 83, s. 2–14.

[13] Lipman, Y. et al. Symmetry factored embedding and distance. In *ACM Transactions on Graphics (TOG)*, 29, s. 103. ACM, 2010.

[14] Martinet, A. et al. Accurate detection of symmetries in 3d shapes. *ACM Transactions on Graphics (TOG)*. 2006, 25, 2, s. 439–464.

[15] Mitra, N. J. – Guibas, L. J. – Pauly, M. Partial and approximate symmetry detection for 3D geometry. In *ACM Transactions on Graphics (TOG)*, 25, s. 560–568. ACM, 2006.

[16] Nelder, J. A. – Mead, R. A simplex method for function minimization. *The computer journal*. 1965, 7, 4, s. 308–313.

[17] Podolak, J. et al. A planar-reflective symmetry transform for 3D shapes. *ACM Transactions on Graphics (TOG)*. 2006, 25, 3, s. 549–559.

[18] Pottmann, H. et al. Geometry and convergence analysis of algorithms for registration of 3D shapes. *International Journal of Computer Vision*. 2006, 67, 3, s. 277–296.

[19] Schiebener, D. et al. Heuristic 3d object shape completion based on symmetry and scene context. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, s. 74–81. IEEE, 2016.

[20] Simari, P. – Kalogerakis, E. – Singh, K. Folding meshes: Hierarchical mesh segmentation based on planar symmetry. In *Symposium on geometry processing*, 256, s. 111–119, 2006.

[21] Sipiran, I. – Gregor, R. – Schreck, T. Approximate symmetry detection in partial 3D meshes. In *Computer Graphics Forum*, 33, s. 131–140. Wiley Online Library, 2014.

[22] Stephenson, M. – Clark, A. – Green, R. Novel methods for reflective symmetry detection in scanned 3D models. In *Image and Vision Computing New Zealand (IVCNZ), 2015 International Conference on*, s. 1–6. IEEE, 2015.

[23] Sun, C. – Sherrah, J. 3D symmetry detection using the extended Gaussian image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1997, 19, 2, s. 164–168.

[24] Thrun, S. – Wegbreit, B. Shape from symmetry. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2, s. 1824–1831. IEEE, 2005.

[25] VÁŠA, L. et al. Mesh statistics for robust curvature estimation. `http://graphics.zcu.cz/curvature.html`. Accessed: 2018-3-29.

[26] WENDLAND, H. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in computational Mathematics*. 1995, 4, 1, s. 389–396.

[27] WOLD, S. – ESBENSEN, K. – GELADI, P. Principal component analysis. *Chemometrics and intelligent laboratory systems*. 1987, 2, 1-3, s. 37–52.

[28] ZABRODSKY, H. – PELEG, S. – AVNIR, D. Symmetry as a continuous feature. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1995, 17, 12, s. 1154–1166.