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

Master's thesis

Generalized Mirror Symmetry of a Point Set in E3

Plzeň 2023

Bc. Eliška Mourycová

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Eliška MOURYCOVÁ**
Osobní číslo:	**A20N0061P**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Počítačová grafika**
Téma práce:	**Zobecněná zrcadlová symetrie množiny bodů v E3**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s takovými existujícími metodami hledání zrcadlové symetrie množiny bodů v E3, které by bylo možné využít pro symetrii generovanou obecnou plochou.
2. Analyzujte, jaké omezující požadavky je potřebné anebo vhodné vznést na plochu, aby bylo možné zadaný problém vyřešit.
3. Navrhněte vhodné řešení. Pokud to považujete za vhodné, obecná plocha může být nahrazena po částech lineární aproximací.
4. Navrženou metodu implementujte a vyzkoušejte. Můžete ve své implementaci využít předchozí programové vybavení založené na metodě Ing. Hrudy.
5. Dosažené výsledky zhodnoťte.

Rozsah diplomové práce:	**doporuč. 50 s. původního textu**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování diplomové práce:	**tištěná/elektronická**
Jazyk zpracování:	**Angličtina**

Seznam doporučené literatury:

dodá vedoucí diplomové práce

Vedoucí diplomové práce:	**Prof. Dr. Ing. Ivana Kolingerová** Katedra informatiky a výpočetní techniky
Datum zadání diplomové práce:	**9. září 2022**
Termín odevzdání diplomové práce:	**18. května 2023**

L.S.

Doc. Ing. Miloš Železný, Ph.D. děkan	**Doc. Ing. Přemysl Brada, MSc., Ph.D.** vedoucí katedry

V Plzni dne 11. října 2022

Declaration

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

Plzeň, 17th May 2023

Bc. Eliška Mourycová

Abstract

This thesis deals with generalised reflectional symmetry of point sets in E3. The generalisation is based on replacing a plane as a mirroring surface by a general curved surface. The goal of this work was to formulate a definition of such a generalisation of mirror symmetry and to propose methods capable of finding a set of points lying on the surface of symmetry or the surface itself. The proposed algorithms were implemented and their functionality was tested on various types of input data.

Abstrakt

Tato práce se zabývá zobecněnou zrcadlovou symetrií množin bodů v E3. Toto zobecnění je založeno na nahrazení roviny jako zrcadlící plochy obecně zakřivenou plochou. Cílem této práce bylo formulovat definici takového zobecnění zrcadlové symetrie a navrhnout metody pro detekci bodů, které leží na ploše symetrie, nebo pro detekci samotné plochy. Navržené algoritmy byly implementovány a jejich funkčnost byla testována na různých typech vstupních dat.

Acknowledgements

I would like to express my gratitude to Prof. Dr. Ing. Ivana Kolingerová, Ph.D., for her guidance, advice, and support throughout the process of writing this diploma thesis. I would also like to extend my thanks to Prof. RNDr. Miroslav Lávička, Ph.D., for offering his time and expertise to consult on the mathematical requirements of this work. Finally, I would like to thank Ing. Lukáš Hruda, for his help regarding the modification of the symmetry detection method.

Contents

1 Introduction

Many objects show various types of symmetry. These objects can be of mathematical or digital nature, such as curves, surfaces or point clouds or they can be real-world objects. Information about an object's symmetry is in many cases useful and important. For example, it can be used for object description and classification or for data compression.

An object is symmetrical when it is invariant, i.e., it remains unchanged under some transformation such as translation, reflection, rotation, scaling or more abstract operations. These transformations can also be used simultaneously or sequentially to create more complex types of symmetry.

Another possible classification of symmetry types is according to whether it is understood in the context of the whole input object or only in its part. These types of symmetries are called global or local, respectively.

This work deals with a generalisation of global reflectional symmetry of 3D point clouds. If a 3D point cloud retains its original shape after being reflected over a plane, it is reflectionally symmetrical with respect to that plane. In this case, the plane is known as the object's symmetry plane. Our focus in this work was to search for this type of symmetry with the generalisation of replacing the plane with a general surface, which we call the surface of symmetry. To our knowledge, there is no research in computer science studying reflectional symmetry with respect to a general curve or surface rather than a plane or axis.

In the case of generalised symmetry, the applications mentioned above can be extended to for instance more specific object description or to a new method of object deformation.

This thesis is part of the research work within the international GAČR project: 21-08009K Generalized Symmetries and Eqiuvalences of Geometric Data. In addition to the GAČR project, I would like to acknowledge and give my thanks to SGS projects that supported this work, these are: SGS-2022-015 New Methods for Medical, Spatial and Communication Data and SGS-2019-016 Synthesis and Analysis of Geometric and Computing Models.

The outline of this work is as follows. In Chapter 2 mathematical background and relevant methods for symmetry detection in computer graphics are introduced. Chapter 3 focuses on the details of the generalisation studied in this work and the proposed methods for the generalised symmetry detection. Chapter 4 describes the technical details of the implementation of the proposed algorithms. In Chapter 5 the results acquired by using the proposed methods are presented on various types of input data. Finally, Chapter 6 concludes the text.

2 Related Work

In this chapter, research related to our problem will be presented. Section 2.1 will talk about the mathematical background on symmetry. The study of symmetry is systematised and formalised in the branch of mathematics known as group theory. Even though we do not use a strictly mathematical approach to the definition and solution of our problem, we find it useful and important to understand the underlying properties of symmetry. Other types of symmetry and their research is briefly discussed in Section 2.3.

This work deals with the generalisation of global reflectional symmetry of point clouds, more specifically with mirror symmetry over curved surfaces (mirrors), therefore, Section 2.2 is devoted to research done in conventional reflectional symmetry of point clouds and Section 2.4 introduces principles and algorithms used for solving problems of non-planar reflection. Special attention should be paid to Subsection 2.2.4 where work by Hruda et al. is introduced and which was used as a basis for one of our methods searching for generalised symmetry.

2.1 Mathematical Background

In this chapter, we will mainly draw from [28] to define symmetry. This article attempts to introduce a universal concept of symmetry, which does not rely on any strong assumptions, such as the existence of the Euclidean structure for geometric symmetries. The definition of symmetry used in this article involves the use of distance-preserving transformations and since the metric to be preserved is not specifically defined, this definition is applicable to many problems, such as symmetry in both Euclidean and non-Euclidean spaces or symmetry of graphs.

Definition of symmetry is also offered in report [21] examining developments in symmetry detection, however, it focuses more on the geometrical understanding of symmetry.

2.1.1 The Assumptions about Objects

Intuitively, an object is symmetric when it is identical to a transform of itself. Therefore, we must be able to declare when an object is identical to one of its transforms. We want these transforms to be distance-preserving.

However, we do not require the distance to be Euclidean distance, it can be defined arbitrarily.

First, we define a set E, whose elements are points, which are the subject of the symmetry study in the context of this work. Then, we define objects as subsets of E, i.e., $A \subseteq E$.

To be able to declare when an object is identical to another object, we will define the equivalence relation between objects. We will denote this relation by $\equiv$. The equivalence relation holds between objects, if the following conditions are true [29]:
$\forall\, A, B, C$:

1. An object is identical to itself. (*Reflexivity*)

$$A \equiv A$$

2. If an object is identical to a second object, then the second one is identical to the first one. (*Symmetry*)

$$A \equiv B \iff B \equiv A$$

3. If an object is identical to a second object, and this latter is identical to a third object, then the first is identical to the third one. (*Transitivity*)

$$A \equiv B, A \equiv C \implies A \equiv C$$

Together with the set E, let us consider a function $\delta : (E \times E) \to \mathbb{R}$, which satisfies the following properties.
$\forall\, \mathbf{x}, \mathbf{y}, \mathbf{z} \in E$:

1. $\delta(\mathbf{x}, \mathbf{y}) \geq 0$
2. $\delta(\mathbf{x}, \mathbf{y}) = \delta(\mathbf{y}, \mathbf{x})$
3. $\mathbf{x} = \mathbf{y} \iff \delta(\mathbf{x}, \mathbf{y}) = 0$
4. $\delta(\mathbf{x}, \mathbf{z}) \leq \delta(\mathbf{x}, \mathbf{y}) + \delta(\mathbf{y}, \mathbf{z})$

Then (E, δ) is a metric space. This means that we are able to compute the distance between any two elements of E. Unless otherwise specified, a metric space (E, δ) will be denoted by E until the end of the section.

2.1.2 The Assumptions about Transforms

Having defined the metric space E (together with the metric δ) and the equality over the set of objects, we will now consider a set of transforms F. A key premise regarding the modelling of symmetry is that objects defined on E (a set of points) are transformed via transforms of the elements of E (points).

Let $\mathbf{x} \in E$ and $U \in F$. We will denote it $\mathbf{y} = U\mathbf{x}, \mathbf{y} \in E$ the image of $\mathbf{x}$. We assume that any element has exactly one image of $\mathbf{x}$ through a given transform $\mathbf{y} = U\mathbf{x}$.

With this in mind, we can define equality between two transforms:

$$U_1 = U_2 \iff U_1\mathbf{x} = U_2\mathbf{x}, \forall\, \mathbf{x} \in E.$$

This equality satisfies the properties 1, 2 and 3 of the equality relation mentioned in Section 2.1.1.

Looking for symmetry in an object leads us to comparison of this object to its image through some transform of the set E on which the object is defined. In this context, we would like to consider the inverse transform associating each element $\mathbf{x}$ of E to its image $\mathbf{y}$. Let U^{-1} be the inverse of U. In order for any element to have exactly one image through transform U^{-1}, all transforms $U \in F$ must be bijections of E onto E.

We will denote the transform of an object A as UA, which is obtained by applying the transform U to each element of A.

2.1.3 Composition of Bijections

Let G be the set of all bijections from E onto E, $F \subset G$. Composition (U_1U_2) of bijections U_1 and U_2 is defined below:

$$\forall\, \mathbf{x} \in E,\ \forall\, U_1, U_2 \in F : (U_1U_2)\mathbf{x} = U_1(U_2\mathbf{x})$$

The set G is a group for the composition of bijections. The existence of a group structure has been many times pointed out in the context of symmetry (even in our previous work which focused on symmetry of curves [22]). We need to define a class of transformations which indeed form a group for the operation of composition. Therefore, we define F as:

$$\forall\, \mathbf{x}, \mathbf{y} \in E \text{ and } \forall\, U \in F : \delta(U\mathbf{x}, U\mathbf{y}) = \delta(\mathbf{x}, \mathbf{y}).$$

In other words, F is a set of bijections preserving the distance δ defined on E. F has the following properties:

1. $F \neq \emptyset$ because the neutral element I_F satisifes the condition that $I_F\mathbf{x} = \mathbf{x}$, $\forall\, \mathbf{x} \in E$.
2. $\forall\ U_1, U_2 \in F :\ U_1U_2 \in F$, i.e., F is stable for the composition of distance-preserving bijections.
3. $\forall\, U \in F :\ U^{-1} \in F$.

2.1.4 Definition of Symmetry

We now have all the terminology needed for the definition of symmetry – to summarise:

Let E be a metric space, δ its associated distance function, and F the set of all bijections of E onto E preserving δ. The neutral element of F is denoted as I_F.

Let A be an object defined on E.

To define symmetry, we state the following: An object A is symmetric if there is a bijection $U \in F$, with $U \neq I_F$, such that $UA \equiv A$.

2.1.5 Symmetry in the Context of this Work

The definition of symmetry as it was introduced in this section will be summarised here, and we will also specify some of its components to put it in the context of this thesis.

This work deals with symmetry of point clouds in 3D space, therefore, to us the set E is a set of points of three coordinates. The metric δ associated with the set E is the Euclidean metric, therefore, (E, δ) is three-dimensional Euclidean space.

By an object as it was defined above, we understand the input set of points, therefore, we will search for symmetry in the context of some geometric transformation. More specifically, the transformation we consider is reflection.

It should be noted that we do not require the transformation to be bijective. The reason for this is that real data are seldom perfectly symmetrical, and the sampling of real-world objects may lead to incomplete or otherwise corrupted data. Moreover, in the case of reflecting points over curved (e.g. spherical) mirrors, a situation where multiple points have the same image may arise.

It should also be pointed out that our choice of a generalisation of symmetry is not the only possible one. Other types of symmetries or their generalisations may include scaling [15, Chapter 1]. In such a case, the condition of distance-preserving transformations may not be practical.

2.2 Planar Symmetry

There has been thorough research done in planar symmetry detection and the interest in this topic is still substantial. We will be mentioning a select amount of works which deal with finding reflectional symmetry in point clouds, because this work's interest is finding symmetry in point cloud data as well.

2.2.1 Combès et al.

This article [9] focuses on finding a symmetry plane in 3D point clouds. It uses an iterative ICP-like (Iterative Closest Point) approach to solve the problem.

The authors discuss the limitations of rigid-body transformation-based methods: they argue that the optimal transformation composed with a superposition of points over a plane, does not necessarily define a proper reflection, and therefore, the optimal plane cannot be computed directly using the composition. For instance, if the transformation is a pure translation that is not perpendicular to the tested plane, then this transformation composed with the superposition of points over this plane is not a reflection.

The authors propose an approach which looks for the symmetry plane directly, i.e., without relying on an intermediate rigid-body transformation. They formulate the problem as:

$$\widetilde{P} = \underset{P, y_1, \ldots, y_N}{\operatorname{argmin}} \sum_{x_i \in O} ||y_i - S_p(x_i)||^2 = \underset{P, y_1, \ldots, y_N}{\operatorname{argmin}} \varepsilon_P,$$

where $y_1, \ldots, y_N$ are the points belonging to the point cloud O and S_P is a function superposing (mirroring) the input point over a plane P.

The method was evaluated on both symmetrical and asymmetrical data. In Fig. 2.1 there are three objects shown along with their respective symmetry planes. In the case of the face and the chair displayed in this figure, the data is incomplete and in the case of the bunny the data is inherently asymmetrical.

The algorithm was also tested on point clouds acquired by scanning faces of more than a hundred subjects. In the conclusion of the article, the authors claim that the presented algorithm is fast, robust and accurate.

2.2.2 Lipman et al.

This paper [14] introduces tools to analyse and represent symmetries in a point set, these tools are Symmetry Factored Embedding (SFE) and Sym-

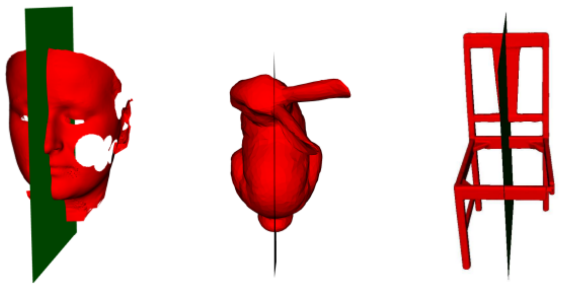


Figure 2.1: Estimation of the symmetry plane acquired by the algorithm proposed by Combès et al. (figure taken from [9])

metry Factored Distance (SFD). The SFE offers new coordinates and the SFD represents Euclidean distance. The space of symmetric correspondences between points, or as the authors call it - orbits, is defined by these constructs. One important finding is that in a correspondence graph created by pairwise similarities, a set of points in the same orbit appears as a clique. Therefore, the problem of finding approximate and partial symmetries in a point set reduces to the problem of measuring connectedness in the correspondence graph, which is a well-studied problem for which spectral methods provide a robust solution.

The authors' goal is to detect and quantify symmetries in a point set $\mathcal{X} = x_{i}{}_{i=1}^{n} \subset \mathbb{R}^d$ and their approach to this problem is based on constructing a correspondence graph. The symmetry correspondence graph is a graph whose vertices are the points in $\mathcal{X}$, with undirected edges (x_i, x_j) between points in the same orbit. Two points are in the same orbit if there exists a symmetry transformation which takes x_i to x_j. The symmetry correspondence graph can be described by an adjacency matrix $C \in \mathbb{R}^{n \times n}$. If the rows and columns of this matrix are rearranged according to the orbits, C is a block-diagonal matrix in which each block consists of only ones, and zeros appear everywhere outside the blocks.

The authors have proved that the number of non-zero eigenvalues of matrix C equals the number of orbits, the magnitude of each eigenvalue is the size of that orbit, and there exists an eigenvector corresponding to each eigenvalue that is constant on the corresponding orbit and zero everywhere else. The top eigenvectors multiplied with their eigenvalues are then used to define an embedding of the point set in a higher dimensional space where Euclidean distance in that space “factors out symmetry”. Since in the case of perfect symmetry, only the eigenvectors corresponding to nonzero eigenvalues are constant on orbits and they span the space of functions constant on orbits, this procedure will lead to an embedding where the Euclidean distance is zero between points in the same orbit and nonzero between points in different orbits. This embedding is called the Symmetry Factored Embedding (SFE) and the corresponding Euclidean distance in that space the Symmetry Factored Distance (SFD).

In the first step the symmetry correspondence matrix $C \in \mathbb{R}^{n\times n}$ for the point set $\mathcal{X} = x_{i\,i=1}^{\,n} \subset \mathbb{R}^d$ is calculated. The numbers $C_{ij} \in [0,1]$ quantify continuously how much the points x_i, x_j belong to the same orbit.

The computation is described by the dissimilarity measure of symmetry between pairs of points undergoing global, rigid transformations. They define the matrix $S \in \mathbb{R}^{n\times n}$ as:

$$S_{ij} = S(x_i, x_j) = \inf_{g\in\mathcal{T}:gx_i=x_j} D(\mathcal{X}, g\mathcal{X}),$$

where $\mathcal{T}$ denotes the rigid transformations, $D(\mathcal{X},\mathcal{Y})$ is a deviation measure between point sets $\mathcal{X},\mathcal{Y}$, the authors used Root Mean-Squared Deviation (RMSD):

$$D(\mathcal{X},\mathcal{Y}) = \left(\frac{\sum_{i=1}^{n} d(x_i,\mathcal{Y})^2 + \sum_{j=1}^{n} d(y_i,\mathcal{X})^2}{2n}\right)^{1/2},$$

where $d(x_i,\mathcal{Y}) = min_j\|x_i - y_j\|$ is the Euclidean distance from point x_i to the set $\mathcal{Y}$.

S_{ij} measures how well can $\mathcal{X}$ be preserved by a rigid transformation that takes x_i to x_j. If $Sij = 0$ then x_i, x_j are in the same orbit of a perfectly symmetric shape.

Once the dissimilarity matrix S is obtained, it can be converted to the (unnormalized) symmetry correspondence matrix via

$$\widetilde{C}_{ij} = e^{-\left(\frac{S_{ij}}{\sigma\ diam}\right)^2},$$

where $diam = max_{ij}\|x_i - x_j\|$ is the point set's diameter, and $\sigma > 0$ is a localization parameter which sets the confidence in the higher values of the dissimilarity symmetry measure.

Once the symmetry correspondence matrix is obtained, the Symmetry Factored Embedding can be calculated. The SFE $\Pi^t : \mathcal{X} \to \mathbb{R}^n$ is defined as:

$$\Pi^t(x_i) = (\lambda_1^t\psi_1(x_i), \lambda_2^t\psi_2(x_i), \ldots, \lambda_n^t\psi_n(x_i)),$$

where ψ_k and λ_k, $k = 1,\ldots,n$ are the eigenvectors and eigenvalues (resp.) of C and t is a parameter which controls how much importance to assign to eigenvectors with higher magnitude eigenvalues (hence more symmetry-aware), and how much to ignore the ones with small eigenvalues (less symmetry-aware).

The Symmetry Factored Distance is then defined as the Euclidean distance in the embedded space, that is $\mathbf{d}^t(x_i, x_j)^2 =$

$$\|\Pi^t(x_i) - \Pi^t(x_j)\|^2 = \sum_{k=1}^{n} \lambda_k^{2t} |\psi_k(x_i) - \psi_k(x_j)|^2.$$

Some of the results obtained by this work are shown in Fig. 2.2. In this figure, the top row shows stationary points of symmetry in black. Stationary points of symmetry are points $q \in \mathbb{R}^d$ left fixed by the symmetry transformations of $\mathcal{X}$, that is $g(q) = q$ for all $g \in \mathcal{G}$, where $\mathcal{G}$ is the set of rigid transformations. In the other rows, the stationary points are used to automatically identify the stationary set type of an object (point, line, plane, or none). The procedure of identifying the stationary set is performed using a Principle Component Analysis (PCA) on a modified version of the input set $\mathcal{X}$ and analysing the three eigenvalues with respect to a threshold $\epsilon = 0.001$ – if only one of the eigenvalues is above the threshold, then the stationary set is a line, if two - a plane, and if none - a point. The eigenvalues are shown as bar charts next to each figure, and $\epsilon = 0.001$ is visualised as red line in those plots.

2.2.3 Nagar et al.

This paper [24] poses the problem of reflection symmetry as an optimisation problem by using a reflection symmetry transformation. The authors parametrize the plane of reflection symmetry using a rotation matrix and a translation vector and represent the correspondences between the reflective symmetric points using a permutation matrix. The main problem of this approach is that this proposed optimisation problem is non-linear and non-convex in the reflection matrix and NP-hard in the estimation of correspondences between the reflective symmetric points. The authors propose a fast randomised algorithm to initialise the reflection matrix such that the estimated reflection matrix is close to the global minimum. They also initialise the translation vector as the mean of the input point cloud assuming that the reflection symmetry plane of an object passes through the centre of gravity of the object. The reflection matrix and the translation vector are also used to estimate the symmetric correspondences. The reflection matrix and the translation vector are iteratively updated using these correspondences until until the computation converges.

To formulate the problem, let $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ be the input point cloud with n points and represented by a matrix $\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_n \end{bmatrix} \in \mathbb{R}^{3\times n}$. The reflection of a point about a given symmetry axis is obtained by

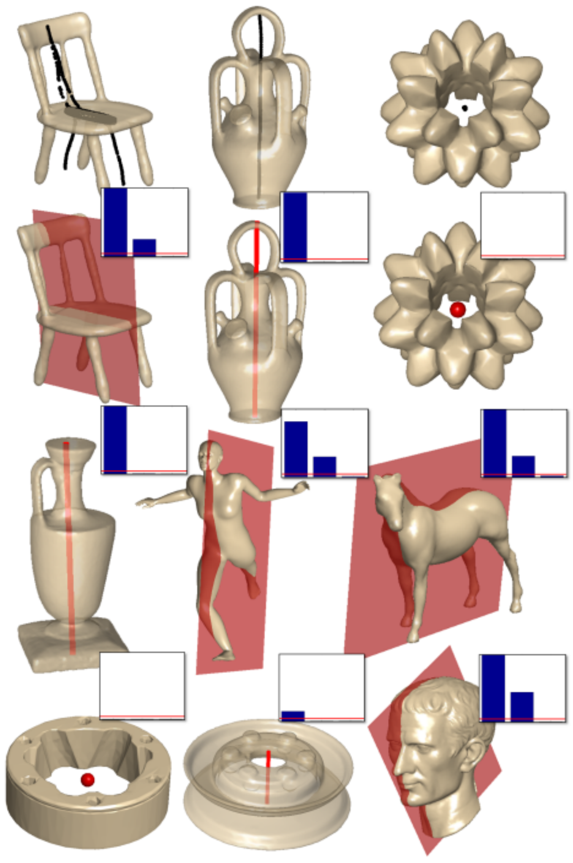


Figure 2.2: Finding approximate stationary locus of symmetry for a collection of 3D models, work by Lipman et al. (figure taken from [14])

translating the origin of the coordinate system on the symmetry axis and then by rotating it such that the x-axis is aligned with the symmetry axis. Then, the y-coordinate of the point is negated and rotated back the coordinate system to the original position. A similar sequence of transformations is followed in the 3D space. More formally, if $\mathbf{p}_j$ is the reflection of the point $\mathbf{p}_i$, then this whole procedure can be represented as follows:

$$\mathbf{p}_j = \mathbf{R}_{\theta_x}^T \mathbf{R}_{\theta_y}^T \mathbf{Q} \mathbf{R}_{\theta_y} \mathbf{R}_{\theta_x} (\mathbf{p}_i - \mathbf{c}) + \mathbf{c},$$

where the matrices $\mathbf{R}_{\theta_x}$ and $\mathbf{R}_{\theta_y}$ represent the rotation matrices by the angle θ_x about the x-axis and θ_y about the y-axis, $\mathbf{c}$ is a point lying on the plane of symmetry, and the matrix $\mathbf{Q} = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_2 \\ \mathbf{0}_2^T & -1 \end{bmatrix}$ negates the z-coordinate of a point. Here, $\mathbf{I}_2$ is the 2×2 identity matrix and $\mathbf{0}_2$ is the zero vector of size 2×1.

Let $\mathbf{R} = \mathbf{R}_{\theta_y}\mathbf{R}_{\theta_x}$, then:

$$\mathbf{p}_j = \mathbf{R}^T\mathbf{Q}\mathbf{R}(\mathbf{p}_i - \mathbf{c}) + \mathbf{c}.$$

Therefore, the goal is to find the reflection symmetry transformation matrices $\mathbf{R}_{\theta_x}$, $\mathbf{R}_{\theta_y}$, the point $\mathbf{c}$ and all the pairs $(\mathbf{p}_i, \mathbf{p}_j)$. It is desired for each pair of points that the equality as shown in Equation 2.1 holds. However, this is only possible for perfectly symmetrical objects and is almost never possible in practice. Therefore, it is desired for the error $\|\mathbf{R}^T\mathbf{Q}\mathbf{R}(\mathbf{p}_i - \mathbf{c}) + \mathbf{c} - \mathbf{p}_j\|_2^2$ to be as small as possible. The optimisation problem is defined in the following equation:

$$\underset{\substack{\mathbf{R},\mathbf{c} \\ (i,j),\forall i\in[n]}}{\operatorname{argmin}} \sum_{i\in[n]} \|\mathbf{R}^T\mathbf{Q}\mathbf{R}(\mathbf{p}_i - \mathbf{c}) + \mathbf{c} - \mathbf{p}_j\|_2^2. \tag{2.1}$$

To solve the optimisation problem defined by Eq. 2.1, the authors use an iterative approach. First, the matrices $\mathbf{R}$ and $\mathbf{c}$ are initialised and the pairs of reflective symmetric points are found. The matrices are then updated as well as the mirror images of each point. This process is repeated until there are no further changes in $\mathbf{R}$, $\mathbf{c}$ and the images of each point. The problem defined in Eq. 2.1 can be rewritten to a more convenient form as follows.

Let the correspondences between the reflective symmetric points be denoted by the matrix $\mathbf{\Pi} \in \{0,1\}^{n\times n}$, such that $\mathbf{\Pi}(i,j) = \mathbf{\Pi}(j,i) = 1$, if $\mathbf{p}_j = \mathbf{R}^T\mathbf{Q}\mathbf{R}(\mathbf{p}_i - \mathbf{c}) + \mathbf{c}$ and $\mathbf{\Pi}(i,j) = \mathbf{\Pi}(j,i) = 0$ otherwise. Further, let $\mathbf{S} = \mathbf{R}^T\mathbf{Q}\mathbf{R}$. The matrices $\mathbf{R}$ and $\mathbf{Q}$ are orthogonal and $\det(\mathbf{Q}) = -1$. It is assumed that the plane of symmetry passes through the centre of mass of the object, therefore, the object can be centered and $\mathbf{c} = \mathbf{0}$, since the point $\mathbf{c}$ is any point on the plane of symmetry. Now, the problem can be reconstructed as below.

$$\underset{\mathbf{\Pi}\in\{0,1\}^{n\times n},\mathbf{S}\in\mathbb{R}^{3\times 3},\mathbf{c}\in\mathbb{R}^3}{\operatorname{argmin}} \|\mathbf{S}(\mathbf{P} - \mathbf{c}\mathbf{1}^T) + \mathbf{c}\mathbf{1}^T - \mathbf{P}\mathbf{\Pi}\|_F^2, \tag{2.2}$$

where $\mathbf{1}$ is the vector of size $n \times 1$ with all its elements equal to 1.

To find the reflection matrix $\mathbf{S}$, the matrix $\mathbf{\Pi}$ is fixed and the problem defined in Eq. 2.2 is minimised with respect to the matrix $\mathbf{S}$. A closed-form solution is derived for the matrix $\mathbf{S}$ as follows. The optimisation problem in Eq. 2.2 can be rewritten for optimising it with respect to $\mathbf{S}$ as

$$\max_{\mathbf{S}} \operatorname{trace}(\mathbf{S}(\mathbf{P}\mathbf{\Pi} - \mathbf{c}\mathbf{1}^T)(\mathbf{P} - \mathbf{c}\mathbf{1}^T)^T).$$

Let $\mathbf{W} = (\mathbf{P}\mathbf{\Pi} - \mathbf{c}\mathbf{1}^T)(\mathbf{P} - \mathbf{c}\mathbf{1}^T)^T$ be a matrix and $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ be the singular value decomposition of the matrix $\mathbf{W}$. Then,

$$\max_{\mathbf{S}} \operatorname{trace}(\mathbf{S}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T) = \max_{\mathbf{S}} \operatorname{trace}(\mathbf{\Sigma}\mathbf{V}^T\mathbf{S}\mathbf{U}).$$

Since the matrices $\mathbf{S}$, $\mathbf{U}$, and $\mathbf{V}$ are orthogonal, $\mathbf{V}^T\mathbf{S}\mathbf{U}$ is also an orthogonal matrix. According to the authors, the optimal solution $\mathbf{S}^*$ to this problem satisfies the condition $\mathbf{S}^* = \mathbf{U}\mathbf{V}^T$.

In order to find the optimal $\mathbf{c}$, matrices $\mathbf{R}$ and $\mathbf{\Pi}$ are fixed. The problem defined in Eq. 2.2 is minimised with respect to $\mathbf{c}$. The error function can be rewritten as $\|\mathbf{A} - \mathbf{B}\mathbf{c}\mathbf{1}^T\|_F^2$, where $\mathbf{A} = \mathbf{S}\mathbf{P} - \mathbf{P}\mathbf{\Pi}$ and $\mathbf{B} = \mathbf{S} - \mathbf{I}$. In order to find the optimal point, the gradient of the error function is set to $-2\mathbf{B}^T\mathbf{A}\mathbf{1} - 4\mathbf{B}\mathbf{c}\mathbf{1}^T\mathbf{1}$ with respect to $\mathbf{c} = \mathbf{0}$. Therefore,

$$\mathbf{B}\mathbf{c}^* = \frac{1}{2n}\mathbf{B}^T\mathbf{A}\mathbf{1}.$$

The problem of finding the exact $\mathbf{\Pi}$ is an NP-hard problem. Instead, the approximate $\mathbf{\Pi}$ is found using the nearest neighbour approach as follows. The matrix containing the reflected points as columns is defined to be $\mathbf{P}_r = \mathbf{S}(\mathbf{P} - \mathbf{c}\mathbf{1}^T) + \mathbf{c}\mathbf{1}^T$. Then, the mirror reflection point for the point $\mathbf{p}_i$ is defined as as the nearest column from the columns of the matrix $\mathbf{P}_r$. After finding the approximate reflection points, only such pairs for which it is true that $\mathbf{p}_i$ is the reflection point of $\mathbf{p}_j$ and vice versa are kept.

In order to solve the problem as a whole, an initialisation of either the matrix $\mathbf{\Pi}$ or $\mathbf{S}$ is needed. Furthermore, it is desired for the initialisation to be close to the optimal solution. The authors further propose a randomised algorithm to search for a good initialisation.

There are some results of experiments conducted by the authors visible in Fig. 2.3. The blue coloured points represents the mid point of line segments joining two reflective symmetric points.

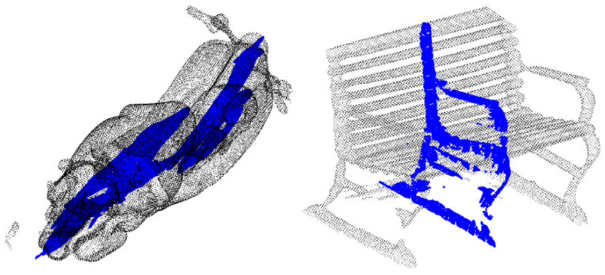


Figure 2.3: The detected plane of reflective symmetry on a few scans of some real-world objects using the proposed approach by Nagar et al. (figure taken from [24])

The authors also conducted experiments on data in which there were various types of noise and imperfections, such as outliers or missing parts in the input model, or perturbation to each point of the model. They have shown their method to be quite robust to these types of imperfections.

2.2.4 Hruda et al.

This article [11] describes a method for symmetry plane detection. This method was directly modified as part of this work to search for a spherical surface of symmetry. The details of the modification are explained in Section 3.4.4.

This method's main attribute is that it defines a differentiable symmetry measure of objects with relation to different planes, which offers fast localisation of maximum of the symmetry measure and therefore, obtain the plane of symmetry.

To define the symmetry measure, the authors first define a vector function $\mathbf{r}(\mathbf{p}, \mathbf{x}) \in E^3$ that reflects a point $\mathbf{x} = [x, y, z]^T \in E^3$ over a plane P given by its implicit equation $P : ax + by + cz + d = 0$ and denoted $\mathbf{p} = [a, b, c, d]^T$ a 4D vector of its coefficients. The vector $\mathbf{n_p} = [a, b, c]^T$ is the normal vector of the plane P.

$$\mathbf{r}(\mathbf{p}, \mathbf{x}) = \mathbf{x} - 2\frac{\mathbf{n_p}^T\mathbf{x} + d}{\mathbf{n_p}^T\mathbf{n_p}}\mathbf{n_p}$$

The components of the function $\mathbf{r}(\mathbf{p}, \mathbf{x})$ are continuous and differentiable with reference to $\mathbf{p}$ except for $\mathbf{p} = [0, 0, 0, d]^T$, which does not represent a valid plane. The authors propose a symmetry measure that evaluates how much a point set $X = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n], \mathbf{x}_i \in E^3, i = 1, \dots, n$ is symmetrical with respect to a given plane P represented by $\mathbf{p}$. The measure is defined as follows:

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

The function $\varphi(l)$ is a radial function such that $\varphi(0) = 1$ and its value approaches 0 as l increases, w_{ij} are weights of point pairs, by default the weights are not used, i.e., $w_{ij} = 1$ for all pairs of points. The symmetry measure s_X is acquired by considering each possible pair of points $\mathbf{x}_i, \mathbf{x}_j \in X$ - the point $\mathbf{x}_i$ is reflected over the plane P and its distance from $\mathbf{x}_j$ is computed and transformed into similarity using φ (called a similarity function). These similarities are summed for all pairs, giving the symmetry measure value for plane P. Maximising $s_X(\mathbf{p})$ will make as many points as possible reflect over P as close as possible to other points.

The symmetry measure $s_X(\mathbf{p})$ is differentiable with respect to $\mathbf{p}$ (except for $\mathbf{p} = [0, 0, 0, d]^T$) when $\varphi(l)$ is differentiable for $l \in \langle 0; \infty)$ and $\frac{d}{dl}\varphi(0) = 0$. The authors used the modified Wendland's function:

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

The value α is the shape parameter of the function.

The authors explain that this modified Wendland's function has a similar shape and spread to the Gaussian $\left(e^{-(\alpha l)^2}\right)$ for the same α. The main difference and reason why they chose this function is that is equals 0 for $\alpha l > 2.6$, therefore, the contribution of any point $\mathbf{x}_i \in X$ to the value of $s_X(\mathbf{p})$ is determined only by points that are not farther than $\frac{2.6}{\alpha}$ from $\mathbf{r}(\mathbf{p}, \mathbf{x}_i)$.

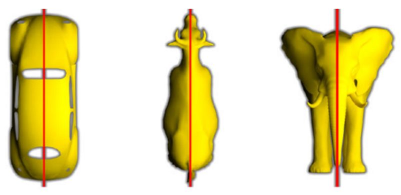
They also set α according to the size of the input object as $\alpha = \frac{15}{l_{\text{avrg}}}$, which makes $s_X(\mathbf{p})$ smoother and easier to optimise than for different settings of α, as the authors have experimentally determined. The value l_{avrg} is the average distance of the points in X from their centroid.

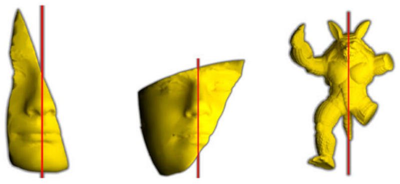
A brute force computation of $s_X(\mathbf{p})$ has time complexity of $\mathcal{O}(n^2)$ but for many pairs $\mathbf{x}_i, \mathbf{x}_j \in X$ the similarity $\varphi(\|\mathbf{r}(\mathbf{p}, \mathbf{x}_i) - \mathbf{x}_j\|)$ equals 0, therefore, it only needs to be computed for pairs where $\|\mathbf{r}(\mathbf{p}, \mathbf{x}_i) - \mathbf{x}_j\| \leq \frac{2.6}{\alpha}$. A uniform grid with the cell size $\frac{2.6}{\alpha} \times \frac{2.6}{\alpha} \times \frac{2.6}{\alpha}$ is constructed. After a point $\mathbf{x}_i$ is reflected over the given plane and falls within a cell C, only points in C and cells adjacent to C are considered for the symmetry measure computation. This way, because the Wendland's function is local, $s_X(\mathbf{p})$ can be computed efficiently and remain first-order differentiable.

To use all points in the input object for symmetry detection can still be computationally expensive if the process happens repeatedly, especially if the number of points is large. Because of this, the input point set is simplified to a lower number of points. The authors observed that their symmetry detection method works well even after the set gets simplified to a rather low number of points, given a proper simplification method is used. They use a simple and fast simplification algorithm: A 3D grid with the cell size of $\frac{l_{\text{avrg}}}{k} \times \frac{l_{\text{avrg}}}{k} \times \frac{l_{\text{avrg}}}{k}$ is created and each occupied cell returns one point of the simplified point set by averaging all points contained in the cell. It is desired to simplify the point set to approximately m points, so the simplification of the original point set is repeated several times with increasing value of k until the resulting point count reaches at least m.

To find a local maximum of the symmetry measure, a quasi-Newton optimisation method L-BFGS is employed. This method uses the gradient of the symmetry measure and it usually converges to a sufficient precision quickly. The authors also experimented with the Nelder-Mead optimisation method, which does not use the gradient, however, this method needed many more iterations to converge to a sufficiently precise result, making the optimisation slower. In our work, we used the Nelder-Mead method, which is a well-known algorithm for multidimensional unconstrained optimisation without derivatives [25][30].

To get a plane of symmetry of an object, several candidate symmetry


(a) Results of the method proposed by Hruda et al. performed on non-corrupted data


(b) Results of the method proposed by Hruda et al. performed on incomplete data

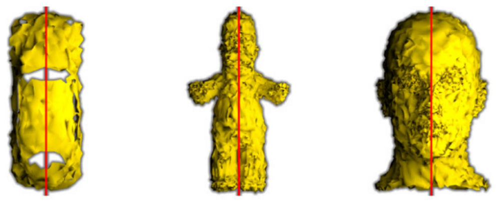

(c) Results of the method proposed by Hruda et al. performed on noisy data

Figure 2.4: Results of the method proposed by Hruda et al. performed on various types of data (figures taken from [11])

planes are created and a small number of them, with the largest symmetry measure, are selected as having the largest potential of being in the convergence region. Then the optimisation is started from these few planes and the resulting plane with the largest final symmetry measure is selected. Planes which are very similar are averaged to speed up the symmetry detection process. Planes are averaged using the following formula:

$$\operatorname{avrg}(\mathbf{p}_u, \mathbf{p}_v) = \begin{cases} \mathbf{p}_u + \mathbf{p}_v & \mathbf{n}_{\mathbf{p}u}^T \mathbf{n}_{\mathbf{p}v} \geq 0 \\ \mathbf{p}_u - \mathbf{p}_v & \mathbf{n}_{\mathbf{p}u}^T \mathbf{n}_{\mathbf{p}v} < 0, \end{cases}$$

where $\mathbf{p}_u, \mathbf{p}_v$ are two planes and $\mathbf{n}_{\mathbf{p}u}, \mathbf{n}_{\mathbf{p}v}$ are the normal vectors of the planes.

Some results of this method are visible in Fig. 2.4. In figures 2.4b and 2.4c we can see that this method performs well even on corrupted data.

2.3 Other Types of Symmetry

Here, in the last section of this chapter, a few of more general types of symmetry will be discussed. We do not mention different symmetry types by the geometric transformation they employ, such as rotational and other types of symmetry, since our focus is on generalised reflectional symmetry.

2.3.1 Intrinsic Symmetry

Intrinsic symmetry can be understood as symmetry which exists only on parts of a studied object. Unlike extrinsic reflectional symmetry which we have focused on in this chapter, intrinsic symmetry looks for symmetrical regions of objects rather than trying to map the object onto itself globally. In a way, it can be understood as a generalisation of standard reflectional symmetry since in the global sense of finding symmetrical pairs of points, different type of transformation is used.

More specifically, intrinsic symmetry is defined as a region over a shape that possesses a self-map that preserves geodesic distances [12].

The work of Jiang et al. [12] focuses on detecting intrinsic symmetry in imperfect 3D point cloud data. Their approach to solving the problem is to take advantage of curve skeleton extraction from point clouds. Starting from a curve skeleton extracted from an input point cloud they use a procedure to vote for symmetric node pairs indicating the symmetry map on the skeleton. A symmetry correspondence matrix (SCM) is constructed for the input point cloud through transferring the symmetry map from skeleton to point cloud. The final symmetry regions on the point cloud are detected via spectral analysis over the SCM. In Fig. 2.5 results of this method are shown. The intrinsically symmetrical parts are coloured with the same colour.

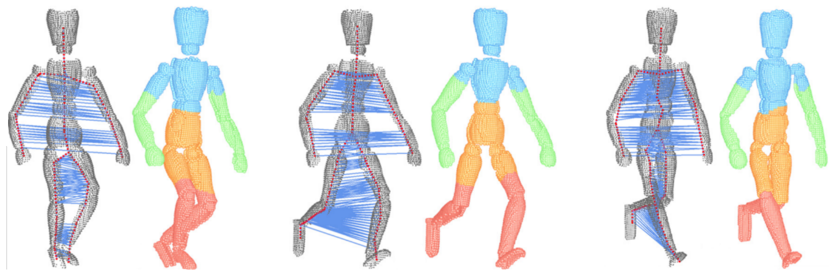


Figure 2.5: Pose-invariant symmetry detection for an articulated Wooden doll. Results of the proposed approach by Jiang et al. (figure taken from [12])

An algorithm introduced by Nagar et al. [23] uses a different approach to solving the problem of intrinsic symmetry detection. It focuses on finding intrinsic reflective symmetry in triangle meshes. Correspondences between functions defined on the shapes are established by extending the functional map framework and then the point-to-point correspondences are recovered. For the functional map the functional correspondences matrix is found. A closed-form solution for the matrix is proposed. To find the closed-form solution, the authors use the analysis of the eigenfunctions of the Laplace-

Beltrami operator for the given shape.

The work described in Section 2.2.2 can also be used for the detection of intrinsic symmetry.

2.3.2 General Symmetry

Methods for finding more general symmetries have also been researched and developed. For instance, work by Mitra et al. [20] studies the detection of meaningful (partial), including approximate or imperfect symmetries in digital 3D shapes. They can detect symmetries under quite general transformations, which may include translation, rotation, reflection, and uniform scaling at once. To achieve this goal, the authors separate the symmetry computation into two phases: In the first step, simple local shape descriptors at a selected set of points on the shape are computed. These descriptors are chosen so that they are invariant under the group actions of interest and they are used to pair up points that could be mapped to each other under a candidate symmetry action. Each such pair can be thought of as depositing mass, or voting, for a specific symmetry in the transformation space of interest. In the second step a stochastic clustering algorithm to extract the significant modes of this mass distribution is used. Since the mapping to transformation space does not preserve the spatial coherence or structure of samples on the input shape, it is verified whether a meaningful symmetry has been found by checking the spatial consistency of the extracted subparts of the surface. The results of the method are visible in Fig. 2.6.

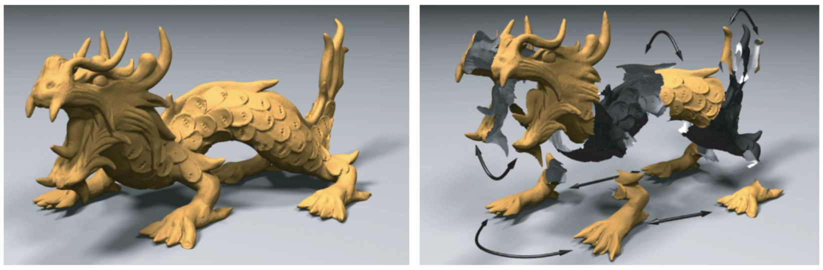


Figure 2.6: Symmetry detection on a sculpted model. Left: Original model, right: Detected partial and approximate symmetries. Results of the proposed approach by Mitra et al. (figure taken from [20])

Bokeloh et al. [7] propose an algorithm computing rigid symmetries, i.e., subsets of a surface of a model that reoccur several times within the model differing only by translation, rotation or mirroring. The method works by first extracting line features from the input model and forming a spatial neighbourhood graph of such features. Then, the graph is examined for reoccurring patterns and finally, the results from matching local clusters of features are transferred and validated on the original geometry.

2.4 Non-Planar Reflection

At the beginning of this section the law of reflection from smooth surfaces is presented. The rest of the section is devoted to algorithms solving problems of reflection from curved surfaces.

2.4.1 The Law of Reflection

This work's focus is in the field of geometrical transformations, however, beyond the scope of this work we also include a brief overview of principles related to the field of optics. Since we are dealing with mirror symmetry, we want to make sure we understand the theory behind reflection. Even though this is intuitively clear, let us use a more formal description of the process of reflection:

"*The law of reflection* [4] *states that, on reflection from a smooth surface, the angle of the reflected ray is equal to the angle of the incident ray. (By convention, all angles in geometrical optics are measured with respect to the normal to the surface – that is, to a line perpendicular to the surface.) The reflected ray is always in the plane defined by the incident ray and the normal to the surface.*"

We cannot exactly implement this law directly into the solution of our problem, since we are not dealing with reflecting light from a surface, but rather reflecting points over a surface. However, it is clear that the normals of the surface play an important role in reflection.

Geometrical optics is a branch of optics which describes light propagation in terms of rays and the use of optical elements such as mirrors [27]. The principles and theory relevant in this field of physics are too complex for our purposes and their use would hinder the applicability of our work in the context of symmetry.

2.4.2 Mitchell et al.

This paper [19] presents a method for the computation of reflected illumination from curved mirror surfaces onto other surfaces. This is comparable, in terms of Fermat's principle [2], to determining the extremal pathways that the light will take in order to travel from its source to the visible surface through the mirrors. The authors also focus on computing the irradiance of the surface after the pathways of illumiantion are found and for that they use the Gaussian curvature of the surface. The authors apply techniques from optics, differential geometry and interval analysis to this problem.

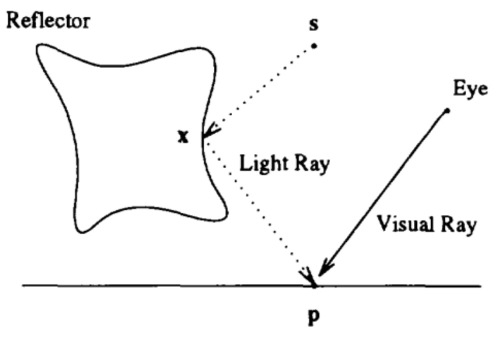


Figure 2.7: Reflected Illumination (figure taken from [19])

Fig. 2.7 illustrates the problem. The point **s** is the light source, point **x** is the point where the light ray bounces off the surface and point **p** is a visible point at which the reflected ray arrives. One of the problems this paper focuses on is to find the proper direction (or directions) in which to cast light rays, so that they arrive at the visible point **p**.

To formulate the path calculation, the authors use Fermat's Principle which states that light travels along extremal paths. The calculation is formulated as a multidimensional non-linear optimisation problem, which can be solved robustly using interval techniques.

Fig. 2.7 represents one ray path from the light **s** to **p** via a reflection at **x**. The total optical path length is a function of **x**:

$$d(x) = \sqrt{(\mathbf{s}-\mathbf{x})^2} + \sqrt{(\mathbf{p}-\mathbf{x})^2}$$

If the mirror surface is defined implicitly by $g(\mathbf{x}) = O$, then the optimization of $d(\mathbf{x})$ subject to the constraint that all points lie on g can be accomplished by the method of Lagrange multipliers. This will give a system of four equations in four variables:

$$\begin{aligned} \nabla d(\mathbf{x}) + \lambda \nabla g(\mathbf{x}) &= 0 \\ g(\mathbf{x}) &= 0 \end{aligned}$$

The solutions of these non-linear equations will yield paths of locally extremal length.

The results are visible in Fig. 2.8. In this experiment, wavefront intensity is not used, therefore, brightness of the reflection is simply a function of how many virtual light sources illuminate each visible point, however, the authors were able to calculate the proper irradiance as well using Gaussian curvature of the object.

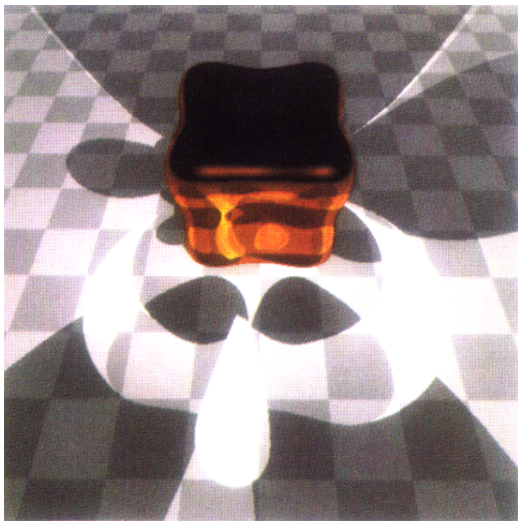


Figure 2.8: Results of the work by Mitchell et al. - Illumination from Virtual Lights (figure taken from [19])

2.4.3 Miguel et al.

This paper [18] offers a solution to rendering non-planar reflections in quadric mirrors. The method, which is based on forward projection, takes advantage of the global information of the vertices as they are computed from their initial positions to their reflection points in the mirror, and then to the camera. This solution does not require the computation of ray intersections.

This paper also heavily focuses on performance and speed of computation as it strives for real-time rendering and instant recalculation of the reflection points depending on the position of the camera.

The projection model the authors used in their work uses three main inputs:

First, a quadric surface reflector, defined by the following quadratic equation:

$$x^2 + y^2 + Az^2 + Bz - C = 0$$

where the coefficients A, B and C are arbitrary scalars. Rotationally symmetric mirrors such as spherical, parabolic, hyperbolic, and elliptic are included in this parameterization of the quadric mirrors. The quadric mirror can also be expressed by a quadric matrix $\mathcal{Q}$, in homogeneous coordinates, such that the point $\mathbf{x} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$ belongs to the quadric $\mathcal{Q}$ if and only if respects the equation $\mathbf{x}^T \mathcal{Q} \mathbf{x} = \mathbf{0}$.

The second input is the camera's center of projection (**COP**), which is placed at the point $\mathbf{COP} = \begin{bmatrix} c_x & c_y & c_z & 1 \end{bmatrix}^T$.

And the third input is the 3D point to be projected (object point), which is defined as $\mathbf{P} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$.

Fig. 2.9 shows the incident ray intersecting the reflector surface at the reflection point **R**, where the light ray is projected to the camera along the reflected direction.

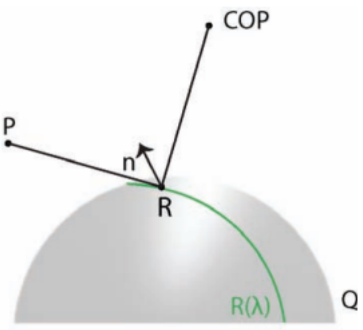


Figure 2.9: Reflection through a quadric reflector where the reflection point is searched in a parameterized quartic curve $R(\lambda)$ (figure taken from [18])

The reflection point is subjected to an extra constraint, which makes it possible to find the reflection point considerably faster. This constraint imposes that the reflection point belongs to both the reflector surface and to an analytical quadric, whose expression depends exclusively on the geometry of the projection (center of projection and 3D point to be projected). As the searched reflection point belongs to these two quadrics, it will be searched for in their intersection, which has only one dimension. Compared to other reflection techniques like the Law of Reflection or the Fermat Principle, this feature makes the procedure significantly faster.

The authors utilise the fact that the parametric curve given by the intersection algorithm is a function of only one parameter λ. The curve can be searched for the point where the total distance travelled by the light is minimum as stated by the Fermat Principle.

Therefore, the reflection point **R** belongs to the quadric reflector $\mathcal{Q}$ and also to the analytical quadric S, whose expression is given by the following equation:

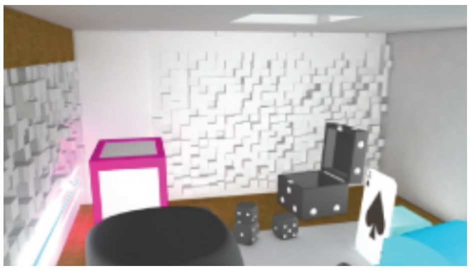
$$S = M^T \mathcal{Q}^*_\infty \mathcal{Q} + \mathcal{Q}^T \mathcal{Q}^*_\infty M,$$

where the matrix $\mathcal{Q}^*_\infty$ is the absolute dual quadric and M is a skew-symmetric matrix that depends on the center of projection of the camera and the 3D point to be projected. M is expressed by:

$$M = \begin{bmatrix} 0 & c_z - Z & -c_y + Y & c_y Z - c_z Y \\ -c_z + Z & 0 & c_x - X & -c_x Z + c_z X \\ c_y - Y & -c_x + X & 0 & c_x Y - c_y X \\ -c_y Z + c_z Y & c_x Z - c_z X & -c_x Y + c_y X & 0 \end{bmatrix}$$

For the general case, the parameterization obtained involves the solution of a polynomial up to the 8th degree.

In Fig. 2.10 the testing environment and the results of reflection are shown.


(a) Testing environment (a room with toys)

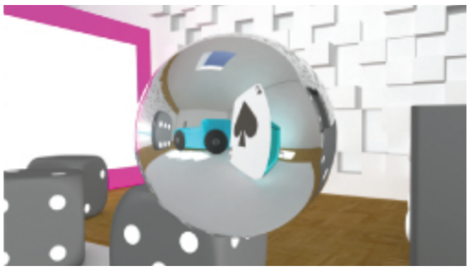

(b) Reflection of the environment on the surface of the mirror

Figure 2.10: Results of the method proposed by Miguel et al. (figures taken from [18])

3 Problem Formulation and Proposed Solutions

In this chapter our choice of reflectional symmetry generalisation will be explained. We will present the main difference between planar reflective and the generalised reflective symmetry this work focuses on. Two different approaches to mirroring points over the curved surface of symmetry will be introduced as well. From now on, when we talk about generalised symmetry, we mean the specific generalisation that will be introduced in this chapter.

In Section 3.1 the requirements on the input data are set. The output of the later discussed algorithms is also talked about. Section 3.2 talks about the general idea of generalised symmetry and shows the connection it has with planar symmetry.

In Sections 3.3 and 3.4 two different ways of mirroring points over surfaces of symmetry are explained. In each of these sections, algorithms designed to solve the problem of calculating points lying on the searched surface of symmetry are also presented.

3.1 Input and Output Data

The input data for all algorithms presented in Sections 3.3 and 3.4 are point clouds of 3D points, i.e., the input set can be described as $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}, \mathbf{x}_i \in E^3, i = 1, \ldots, n$, where n is the size of the input set.

The output of all algorithms presented in Sections 3.3 and 3.4 with one exception is always a set $Y = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m\}, \mathbf{y}_i \in E^3, i = 1, \ldots, m$ of points lying on the surface of symmetry, i.e., if f is the surface of symmetry, then

$$\forall\, \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m : \mathbf{y}_i \in f, i = 1, \ldots, m.$$

However, in most cases we expect the position of the output points to be computed with some error, therefore, the previous statement is rather theoretical than practical.

The output of one of the methods presented later in this chapter is not a set of points lying on the surface of symmetry, but the surface itself. The method calculates a spherical surface of symmetry, therefore, the output is a point $\mathbf{c} \in E^3$ denoting the centre of the sphere and a number $r \in \mathbb{R}$ denoting the radius of the sphere.

3.2 Connection between Planar and Generalised Symmetry

To see the connection between planar reflectional symmetry and its generalisation studied in this work, let us first recall some of the properties of planar symmetry.

Let $\mathbf{x}$ be an arbitrary 3D point, p be a plane and $\mathbf{n}_p$ be the unit normal vector of the plane. We know that points $\mathbf{x}$ and $\mathbf{x}'$ are symmetrical with respect to the plane p if the following formula holds, as was already shown in Section 2.2.4:

$$\mathbf{x}' = \mathbf{x} - 2l\mathbf{n}_p, \tag{3.1}$$

where l is the signed (orthogonal) distance from the point $\mathbf{x}$ to the plane p. If we express the plane p implicitly as $p : ax + by + cz + d = 0$, where the vector (a, b, c) is a unit vector and $(a, b, c) \neq (0, 0, 0)$, we can calculate l using the following formula:

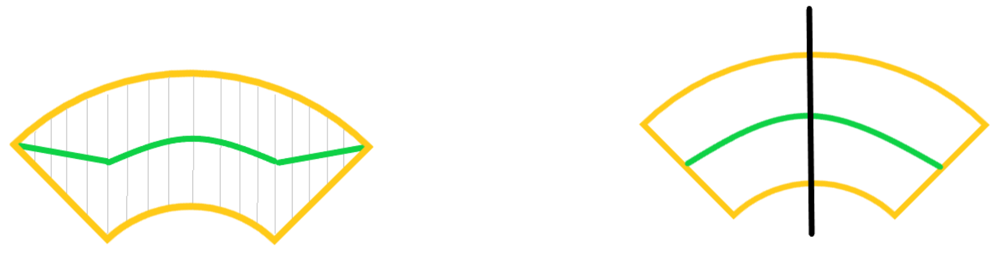
$$l = \frac{\mathbf{n}_p^T\mathbf{x} + d}{\mathbf{n}_p^T\mathbf{n}_p} \tag{3.2}$$

Now, let us consider an input set $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}, \mathbf{x}_i \in E^3, i = 1, \ldots, n$. For each point $\mathbf{x}_i$, we want to find the image $\mathbf{x}'_i$ such that the relationship shown in Eq. 3.1 holds between them.

This means that each point and its image lie on the same line perpendicular to the plane of symmetry and they are equidistant from the plane. When it comes to generalised symmetry, we want these properties to hold even if we replace the plane with a general surface. On the other hand, when reflecting points over a plane, the direction of mirroring is the same for every pair of points, therefore, instead of considering the line containing a point and its image to be perpendicular to the surface of symmetry, we can define the direction as fixed for all points.

See Fig. 3.1 with the illustration of these two options of understanding generalised symmetry. The surface of an object is coloured in yellow. In 3.1b the object's plane, or axis in this 2D example, of symmetry is coloured in black and its surface (curve) of symmetry is coloured in green. It is apparent that relatively to both the plane and the surface each pair of points lies on the same line perpendicular to the surface of symmetry and the surface passes through the midpoint of each pair of points. In 3.1a the surface of symmetry is different, because instead of perpendicular direction of mapping, a fixed direction (indicated by the grey lines inside the object) was used.

It is hard to say, which one of these approaches is better suited for the generalisation of reflectional symmetry. The case of perpendicular direction


(a) Surface of symmetry considering fixed direction of mapping of points

(b) Surface of symmetry considering perpendicular direction of mapping of points

Figure 3.1: Possible surfaces of symmetry in an object

of mapping takes the way curved mirrors operate into account. On the other hand, the task of finding the surface satisfying this condition is not simple, as is further discussed in Section 3.4. When it comes to fixed direction of mapping, the choice of the direction is the main problem of finding such a surface, but once it is determined, the task becomes fairly straightforward.

In most of the algorithms presented in Sections 3.3 and 3.4 we know nothing of the surface we are searching for, e.g., we do not specify whether the surface is generated by a polynomial, whether it can be self-intersecting, etc. The main problem is to determine the mapping of points onto their images in the input point cloud, because if we know the pairing of points, then the problem of calculating the points lying on the surface of symmetry becomes trivial.

3.3 Generalised Symmetry using Fixed Direction of Mapping

This section describes the details of searching for a surface of symmetry using fixed direction of mapping. In Subsection 3.3.1 an algorithm used for the solution of the problem is proposed.

To solve the problem of mirroring over a general surface we can use the same formula shown in Eq. 3.1, only instead of using the normal vector $\mathbf{n}_p$ we use some unit vector $\mathbf{v}$ which will determine the direction of mapping. The calculation of an image of a point $\mathbf{x}$ becomes:

$$\mathbf{x}' = \mathbf{x} - 2d\mathbf{v},$$

or

$$\mathbf{x}' = \mathbf{x} + 2d\mathbf{v},$$

where the sign depends on the position of the point $\mathbf{x}$ relative to the surface of symmetry. In this case, d equals

$$d = \frac{\|\mathbf{x} - \mathbf{x}'\|}{2},$$

because the vectors $\mathbf{v}$ and $(\mathbf{x} - \mathbf{x}')$ are collinear.

Using this direction of mapping $\mathbf{v}$, we determine the pairing of points. The problem now becomes to select the vector $\mathbf{v}$ in such a way that the results satisfy the stated conditions and at the same time the resulting pairing of points does not create for example a surface resembling a plane of symmetry. Theoretically, if instead of point clouds the input data also contained information about the surface of the object, e.g., in the form of a triangle mesh, the vector $\mathbf{v}$ could be selected arbitrarily, because for any setting of $\mathbf{v}$, some set of points lying on the surface of symmetry could be found. In the case of discrete data, we should take the precision of points mapping into account, however, even in the case of point clouds, for the same input set there can be more than one vector $\mathbf{v}$ satisfying this condition. In the following subsection, an algorithm dealing with the case of fixed direction of mapping is presented.

3.3.1 Separation Algorithm

An algorithm searching for a surface of symmetry with the assumption of fixed direction of mapping will be proposed. It focuses on assigning an image to each point in the input set. It does so by projecting two subsets of the input set onto each other and testing some criteria which are described in the following text.

This algorithm is based on the idea of first separating the input point cloud into two subsets between which we should find the mapping, hence the name "Separation Algorithm". If we were to test every possible combination of pairing of points, the time complexity of the procedure would be $\mathcal{O}(n!)$.

We use a plane to separate the point cloud and we use its normal vector as the vector $\mathbf{v}$. We check how precisely this vector $\mathbf{v}$ causes the two subsets to map onto one another, but we also take into account the fact that we do not want the resulting surface to be a plane, therefore, we also penalise such surfaces that are too close to the input point cloud. The method is outlined in Algorithm 1 and illustrated in Fig. 3.2.

In step 1 of Algorithm 1 different planes passing through the centre of mass of the input point cloud are generated. The reason for that is that the centre of mass can be expected to be fairly “in the middle” of the input points, provided the density of points is roughly the same across the whole domain. If the point density is highly variable in different places, the separating plane can be placed inconveniently. To prevent this from happening, some of the points in the higher density area can be removed from the input using the procedure described in Subsection 2.2.4, where a uniform grid is created and the cells of the grid determine which points should be removed in order to achieve a roughly uniform density of points.

After the direction of mapping is determined using a separating plane (step 2) - see Fig. 3.2a, we determine how well the subsets $\mathbf{X}_1, \mathbf{X}_2 \subset \mathbf{X}$ map onto each other under the tested direction of mapping (steps 3 - 8). We do so by constructing different lines passing through points in one of the subsets with their direction vector being the normal vector of the separating plane (see Fig. 3.2b), and orthogonally projecting points in the other subset onto these lines. To project a point $\mathbf{x}$ onto a line given by a point $\mathbf{x}_0$ and the direction vector $\mathbf{v}$, the following formula is used:

$$\mathbf{x}_p = \mathbf{x}_0 + \frac{(\mathbf{x} - \mathbf{x}_0) \cdot \mathbf{v}}{\mathbf{v} \cdot \mathbf{v}} \mathbf{v},$$

where $\mathbf{x}_p$ is the projected point and the operator $\cdot$ denotes the dot product.

For each point in subset $\mathbf{X}_1$, the point $\mathbf{b}$ closest to the line l constructed from said point is found (see Fig. 3.2c). In step 10, we calculate the total projection error as:

$$\epsilon = \frac{1}{k} \sum_{i=1}^{k} \|\mathbf{b}_i - \mathbf{b}_i^l\|^2, \tag{3.3}$$

where k is the number of points in the subset $\mathbf{X}_1$ and $\mathbf{b}^l$ is the projection of the point $\mathbf{b}$ on the line l. The lower the error of mapping, the stronger candidate the separating plane (and the associated surface of symmetry) is. To prevent the algorithm from favouring surfaces resembling a plane of symmetry, the shortest distance d between the points lying on the surface of symmetry and the input point cloud is computed (step 11) and taken into account when evaluating the fitness of a solution - the greater the distance, the better the fitness. The total score of a solution is calculated as:

$$s = \ln(d + 1) + e^{-\epsilon}, \tag{3.4}$$

where the chosen functions cause the score to be the highest for solutions with low ϵ and high d and the lowest for high ϵ and low d.

Algorithm 1 Separation Algorithm

Input: A point set $\mathbf{X}$ of n elements (points) to which a surface of symmetry should be found

Output: A point set $\mathbf{Y}$ of m elements (points) lying on the detected surface of symmetry

```
 1: Generate a set $\mathbf{P}$ of differently rotated planes, all passing through the
    centre of gravity of $\mathbf{X}$, then for each plane $p_i$ in $\mathbf{P}$:
 2:     Get the normal vector $\mathbf{n}_i$ of the plane $p_i$
 3:     Separate $\mathbf{X}$ into two subsets $\mathbf{X}_1, \mathbf{X}_2$ depending on their position
        relative to $p_i$
 4:     For each point in $\mathbf{X}_1$:
 5:         Construct a line $l$ from each point in $\mathbf{X}_1$ using $\mathbf{n}_i$ as its direction
            vector
 6:         Orthogonally project points in $\mathbf{X}_2$ onto the line $l$
 7:         Find the point $\mathbf{b} \in \mathbf{X}_2$ which was closest to the line $l$
 8:         Find the point $\mathbf{m}$ as the midpoint between $\mathbf{b}$ and the current
            point in $\mathbf{X}_1$
 9:         Add $\mathbf{m}$ to $\mathbf{Y}$ as a point lying on the surface of symmetry
10:     Count the total projection error $\epsilon$ using Eq. 3.3
11:     Find the shortest distance $d$ from points in $\mathbf{X}$ to points in $\mathbf{Y}$
12: Return the set $\mathbf{Y}$ yielding the highest score calculated using Eq. 3.4
```

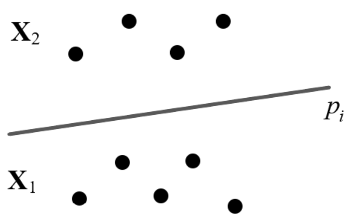
The calculation of the score in Algorithm 1 at step 12 favours surfaces which do not “intersect” the input point cloud and at the same time mirror the points with low error.

The main advantage of this approach is that there is no need to specify the formula or even the type of what the resulting surface of symmetry should be. On the other hand, it seems to produce better results from such input point clouds, that are clearly separable by a plane into two non-overlapping subsets.

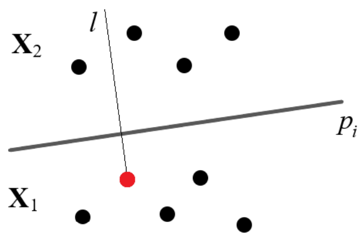
3.4 Generalised Symmetry using Perpendicular Direction of Mapping

This section describes the details of searching for a surface of symmetry using perpendicular direction of mapping. In Subsections 3.4.2, 3.4.3 and 3.4.4 algorithms used for the solution of the problem are proposed.

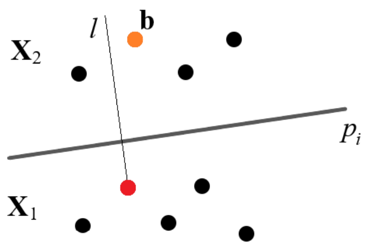
Let f be the surface of symmetry of an input point cloud $\mathbf{X}$ considering



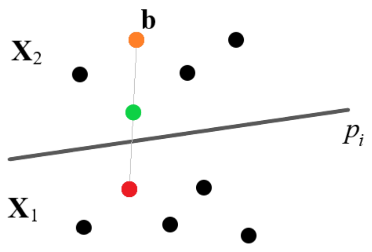
(a) The input set separated into subsets $\mathbf{X}_1$ and $\mathbf{X}_2$ using a plane p_i



(b) The line l created using a point in $\mathbf{X}_1$ (red) and the normal vector of the plane p_i



(c) The closest point $\mathbf{b}$ to l is found (orange)



(d) A point lying on the surface of symmetry is found (green) as the midpoint between the point in $\mathbf{X}_1$ and $\mathbf{b}$

Figure 3.2: Illustration of the Separation Algorithm at work

perpendicular direction of mapping. Every line connecting a point $\mathbf{x}_i$ and its image $\mathbf{x}'_i$ is also a normal line of the surface f. The lines connecting points to their images are not expected to be parallel as in the case of planar symmetry or generalised symmetry using fixed direction of mapping.

To determine the pairing of points without the knowledge of the surface of the input object or the surface of symmetry itself, the task is hard to solve. Algorithms presented in the following subsections all exploit some extra information about the input point cloud, specifically the information about (approximate) normals in all input points (Algorithms 2 and 3) or the added information about the surface of symmetry we are searching for (Algorithm 4).

Algorithms 2 and 3 use the information about the normals of points. Since the input does not contain this information, the normals are estimated using a method described in Subsection 3.4.1. By approximating normals in the input point cloud, we assume the point cloud to represent the sampling of some surface. Estimating normals in a completely random set of points would not be very meaningful. Therefore, we do not expect these methods to be very robust to noise in the input data.

3.4.1 Estimation of Normals of Points in a Point Cloud

Estimation of normals of points $\mathbf{x}_i$ in an input point cloud $\mathbf{X}$ is done by considering a selected amount of points closest to $\mathbf{x}_i$ and fitting a plane through these points. The plane fitting is achieved by using multiple linear regression (MLR).

Multiple linear regression [10] is a statistical method used to model the relationship between two or more independent variables and a single dependent variable $(x_{i1}, x_{i2}, \ldots, x_{i,(p-1)}, y_i)$ for $i = 1, 2, \ldots, n$ units of observation, in our case the size of the input set. The integer p denotes the number of dependent variables, in our case $p = 2$. The multiple linear regression model assumes that the relationship between the dependent variables and independent variables is linear. Therefore, the relationship between the independent variables and the dependent variable can be written as:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_{p-1} x_{i,(p-1)} + \epsilon_i,$$

where $\beta_k, k = 1, 2, \ldots, p-1$ are called regression coefficients. The model is estimated using the least squares method, i.e., β_k is chosen to minimise the sum of squared vertical distances between the predicted values from the model and the actual observed values of y_i:

$$\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_{p-1} x_{i,(p-1)}))^2.$$

3.4.2 Separation Algorithm with Added Normal Estimation

This section describes an algorithm which is a modified version of Algorithm 1, the modified parts are highlighted in Algorithm 2. It still relies on the separation of the input set into two subsets using a plane. As before, the normal vector of the separating plane determines the direction of mapping. The modification lies in exploiting the information about the estimated normals of the input points. While we still search for a low-error projection of points onto their images, we also favour such pairing of points whose normals make a small angle (steps 8 and 11). For this reason, the calculation of ϵ used in Algorithm 1 (Equation 3.3) is modified for this algorithm as:

$$\epsilon = \frac{1}{k} \sum_{i=1}^{k} \|\mathbf{b}_i - \mathbf{q}_i\|^2, \tag{3.5}$$

where $\mathbf{q}_i$ is the point whose normal makes the smallest angle with the normal of the point $\mathbf{x}_i$ (see Fig. 3.3).

We assume points with similar normals to form a pair if we consider perpendicular direction of mapping. However, we can expect points with similar positions (points close to one another) to have similar normals, therefore, the separation of the set into two subsets is still present in this algorithm, since we cannot rely solely on the information about the similarity of normals. The separation part of this algorithm is identical to the one presented in Algorithm 1.

Algorithm 2 Separation Algorithm with added Normal Estimation

Input: A point set $\mathbf{X}$ of n elements (points) to which a surface of symmetry should be found

Output: A point set $\mathbf{Y}$ of m elements (points) lying on the detected surface of symmetry

```
 1: Generate a set $\mathbf{P}$ of differently rotated planes, all passing through the
    centre of gravity of $\mathbf{X}$, then for each plane $p_i$ in $\mathbf{P}$:
 2:     Get the normal vector $\mathbf{n}_i$ of the plane $p_i$
 3:     Separate $\mathbf{X}$ into two subsets $\mathbf{X}_1, \mathbf{X}_2$ depending on their position
        relative to $p_i$
 4:     For each point in $\mathbf{X}_1$:
 5:         Construct a line $l$ from each point in $\mathbf{X}_1$ using $\mathbf{n}_i$ as its direction
            vector
 6:         Orthogonally project points in $\mathbf{X}_2$ onto this line
 7:         Find the point $\mathbf{b} \in \mathbf{X}_2$ which was closest to the line $l$
 8:         Find the point $\mathbf{q}$ whose normal makes the smallest angle with
            the normal of the current point in $\mathbf{X}_1$
 9:         Find the point $\mathbf{m}$ as the midpoint between $\mathbf{b}$ and the current
            point in $\mathbf{X}_1$
10:         Add $\mathbf{m}$ to $\mathbf{Y}$ as a point lying on the surface of symmetry
11:     Count the total projection error using Eq. 3.5
12:     Find the shortest distance from points in $\mathbf{X}$ to points in $\mathbf{Y}$
13: Return the set $\mathbf{Y}$ yielding the highest score calculated using Eq. 3.4
```

In Fig. 3.3 the selection of point **b** and **q** is illustrated. In this figure, the subsets $\mathbf{X}_1$ and $\mathbf{X}_2$ are identical, only shifted vertically, therefore, the point with the most similar normal (blue) is “directly above” the point in question (red).

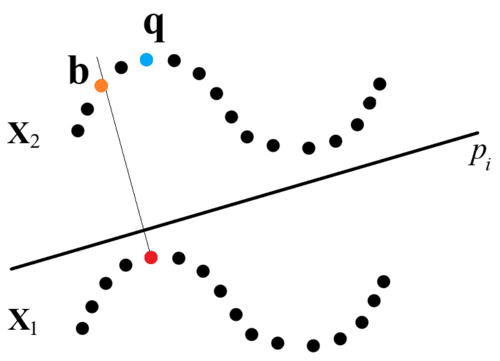


Figure 3.3: Illustration of the Separation Algorithm using normal estimation at work

3.4.3 Algorithm using Normal Estimation Only

In this section, an algorithm which searches for the surface of symmetry using only the information about the estimated normals of the input points is presented, i.e., it does not separate the set into two subsets like Algorithms 1 and 2.

If we assume that the connecting lines of each pair of points are of the same length, we can make very good use of the information about the normals of the points. In this scenario, the images of points can be searched for along the direction of the points' normals. That is

$$\mathbf{x}' = \mathbf{x} - 2d\hat{\mathbf{n}},$$

where d is the distance of point $\mathbf{x}$ from the surface of symmetry in the direction of its normal $\mathbf{n}$. The normalised form of $\mathbf{n}$ is denoted as $\hat{\mathbf{n}}$. Of course, this assumption generally cannot be made, however, this approach can provide some initial strategy in searching for the surface of symmetry, or, if we know that this assumption is true for the studied point cloud, this method can provide fairly accurate results.

An algorithm that integrates this approach is outlined in Algorithm 3.

This algorithm is suitable for such point clouds, which represent a sampled surface of an object, because in such a case it is reasonable to consider the estimation of normals as a relevant piece of information. This method is expected to be sensitive to perturbation of the input point cloud, because this type of noise is likely to cause the normals of points to be estimated incorrectly.

Algorithm 3 Algorithm using Normal Estimation Only

Input: A point set $\mathbf{X}$ of n elements (points) to which a surface of symmetry should be found

Output: A point set $\mathbf{Y}$ of m elements (points) lying on the detected surface of symmetry

```
1: For each point $\mathbf{x}_i \in \mathbf{X}$ estimate its normal $\mathbf{n}_i$
2: Construct a line $l$ from each point in $\mathbf{X}_1$ using $\mathbf{n}_i$ as its direction vector
3: Orthogonally project all other points in $\mathbf{X}$ onto line $l$
4: Find the point $\mathbf{b} \in \mathbf{X}$ which was closest to the line $l$
5: Find the point $\mathbf{m}$ as the midpoint between $\mathbf{b}$ and the current point in $\mathbf{X}$
6: Add $\mathbf{m}$ to $\mathbf{Y}$ as a point lying on the surface of symmetry
```

3.4.4 Algorithm Modifying a Method of Planar Symmetry Detection

This approach is a modification of the existing method by Hruda et al. introduced in Section 2.2.4. Instead of searching for a plane of symmetry, the modification calculates a spherical surface of symmetry. It uses the same framework, the difference is in calculating reflections of points $\mathbf{x}$ over the surface s:

$$\mathbf{r}(s, \mathbf{x}) = \mathbf{x} + 2\|\mathbf{u}\| - r_s\hat{\mathbf{u}},$$

where $\mathbf{u} = \mathbf{c}_s - \mathbf{x}$, r_s is the radius of the sphere s and $\mathbf{c}_s$ is the centre of the sphere. The normalised form of $\mathbf{u}$ is denoted as $\hat{\mathbf{u}}$

The main reason for choosing a sphere as the target surface is that for a spherical surface it is easily possible to formulate the mirroring function. That is because every normal to the surface of the sphere intersects its centre, therefore, it is fairly simple to construct a normal to the surface from a point which does not lie on the surface. For general surfaces, the task is much harder to solve and our goal was to show that such a modification of the existing method is possible.

For clarity, the outline of the algorithm is presented in Algorithm 4. The only difference between the original and the modified method is the mirroring function and the use of an optimisation technique which does not rely on the gradient of the symmetry measure.

The main advantage of this approach is that it merely modifies a robust and extensively tested algorithm, most of the parts of the original method are left unchanged. The drawback is the fact that it is necessary to select the target surface before the algorithm can be run.

Algorithm 4 Planar Symmetry Detection Modification

Input: A point set $\mathbf{X}$ of n elements (points) to which a surface of symmetry should be found

Output: A sphere s

```
1: Generate a set of spheres $\mathbf{S}$, for each $s_i \in \mathbf{S}$:
2:     Reflect points $\mathbf{x}_i \in \mathbf{X}$ over sphere $s_i \in \mathbf{S}$
3:     Find the closest point to the reflected point and calculate
       the symmetry measure using a radial function as in the original
       method
4: Return the sphere with the maximal symmetry measure
```

3.5 Constructing a Surface from Points lying on the Surface of Symmetry

In most of the presented algorithms we focused on finding a set of points lying on the surface of symmetry, rather than the surface itself. While such a result can be enough for some applications, for others one would prefer to have complete information about the surface. This work does not include the calculation of a surface from a set of points, because as the specific application is unknown, the requirements on the resulting surface are also unknown. Furthermore, the thesis supervisor requested that problem of constructing a surface from a point set not be solved.

Constructing a surface out of a set of points is a well-studied problem and there are several methods fit for solving the task. In the following paragraphs a selected amount of methods is mentioned, however, the list is not exhaustive.

When reconstructing a surface from a set of 3D points, there are several options available. One possible approach is to reconstruct the surface using triangulation. A well-known technique is Delaunay triangulation, which tessellates the domain optimally with simplices satisfying the property that no point within the triangulated set lies inside the circumsphere of any simplex [16]. A survey studying 3D surface reconstruction [13] makes a reference to an improved version of the Delaunay triangulation called the Crust algorithm. It is a Voronoi-based algorithm, which guarantees the output to be topologically correct and convergent to the original surface, given the sampling of the object is dense enough in highly detailed areas [5] [6].

In some cases, it may be more desirable to acquire a smooth surface rather than a triangulation from the set of points. To achieve this, an option is to interpolate or approximate the points with a surface defined by some function. One possibility is to use polynomial interpolation [3] of the given points. When it comes to polynomial interpolation, a preferred option may be the use of spline interpolation [17], since it avoids the problem of creating oscillations between points when interpolating using high-degree polynomials.

Another option is to use radial basis functions (RBF) for the interpolation or approximation of the given data set. The use of RBFs for such a task is a well-studied problem [8] and it can achieve satisfactory results with low time and memory complexity [31].

An alternative approach is to use methods from machine learning which solve regression tasks, such as polynomial regression [26] or an artificial neural network fit for solving regression problems [32]. However, the use of a neural network might not be suitable, as neural networks usually require a large amount of training data in order to provide sufficiently accurate results.

4 Technical Documentation

This chapter offers a brief overview of the technical specifics of the project that has been developed as part of this work. It will discuss some implementation details concerning the presented methods for generalised symmetry detection as well as a concise outline of the project architecture. Special attention is paid to the modification of the planar symmetry detection method.

The project was implemented using the C# programming language with the target framework being .NET Framework 4.6.1.

The solution architecture is outlined in Fig. 4.1. It shows all of the projects contained within the solution. Those coloured in yellow were not implemented as part of this work, however, some of them were modified and if that is the case, the modified or added class is shown in white inside the yellow box. Moreover, not all classes contained in the projects which were not implemented as part of this work are shown in the diagram. Only those that are to a significant extent important for the working of the implemented code are included.

Some implementation details of the presented algorithms are provided in the following text.

Algorithms 1 and 2 are implemented in the **SeparationAlgorithm** class. Their functionality depends on a set of differently rotated planes. The generation of this set is contained in the **Utils** class and it is acquired by generating points on the surface of a unit sphere with its centre in the origin. These points are then considered the normal vectors of the planes. The points are generated using the following equations for the calculation of their x, y and z coordinates:

$$x = \sin(\varphi)\cos(\theta),$$

$$y = \sin(\varphi)\sin(\theta),$$

$$z = \cos(\varphi),$$

where the interval for both φ and θ is $(0, \pi)$ and the step in the interval is set to $\frac{1}{10}\pi$.

Algorithm 2 is implemented in the **NormalOnlyAlgorithm** class. Together with Algorithm 2, it relies on the estimated normals of points. The estimation of normals is implemented in the **PointCloudTriangleMesh**

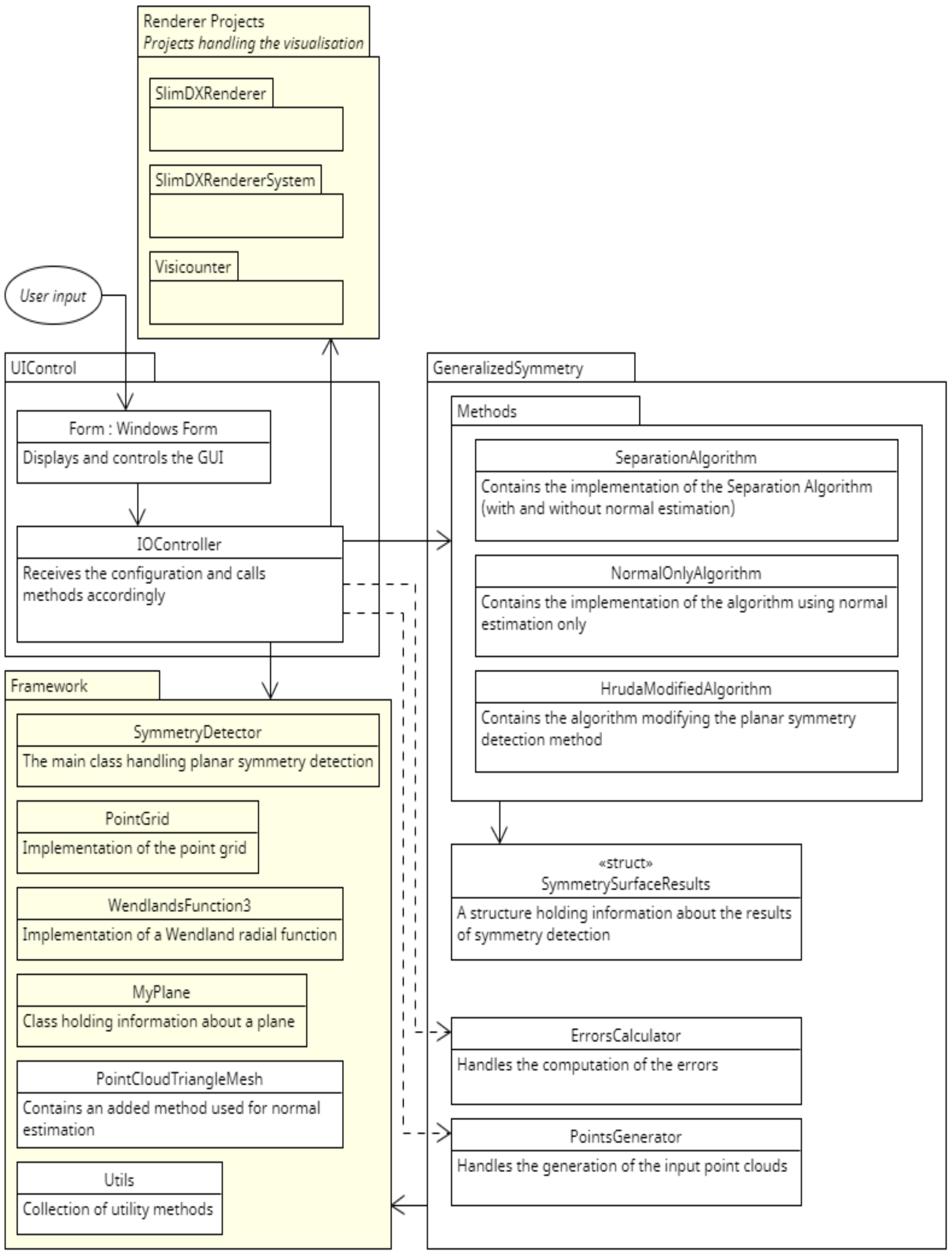
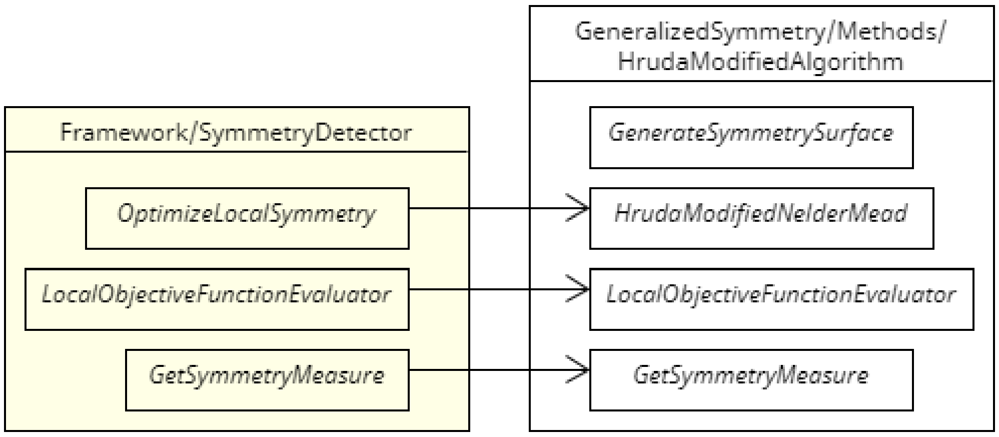

Renderer Projects
Projects handling the visualisation
SlimDXRenderer
SlimDXRendererSystem
Visicounter
User input
UIControl
Form : Windows Form
Displays and controls the GUI
IOController
Receives the configuration and calls methods accordingly
Framework
SymmetryDetector
The main class handling planar symmetry detection
PointGrid
Implementation of the point grid
WendlandsFunction3
Implementation of a Wendland radial function
MyPlane
Class holding information about a plane
PointCloudTriangleMesh
Contains an added method used for normal estimation
Utils
Collection of utility methods
GeneralizedSymmetry
Methods
SeparationAlgorithm
Contains the implementation of the Separation Algorithm (with and without normal estimation)
NormalOnlyAlgorithm
Contains the implementation of the algorithm using normal estimation only
HrudaModifiedAlgorithm
Contains the algorithm modifying the planar symmetry detection method
«struct»
SymmetrySurfaceResults
A structure holding information about the results of symmetry detection
ErrorsCalculator
Handles the computation of the errors
PointsGenerator
Handles the generation of the input point clouds

Figure 4.1: Project architecture diagram

class using the Accord.NET library. [1] A plane is fit through a close neighbourhood of the point whose normal is to be estimated by using multiple linear regression. The size of the neighbourhood was chosen to be ten closest points.

Algorithm 4 is implemented in the **HrudaModifiedAlgorithm** class.

This class contains copied and modified methods originally imlpemented by Ing. Lukáš Hruda in the **SymmetryDetector** class. Fig. 4.2 shows a detail of these classes with their methods. The arrows show which methods were copied and modified in this work.



Framework/SymmetryDetector
OptimizeLocalSymmetry
LocalObjectiveFunctionEvaluator
GetSymmetryMeasure
GeneralizedSymmetry/Methods/HrudaModifiedAlgorithm
GenerateSymmetrySurface
HrudaModifiedNelderMead
LocalObjectiveFunctionEvaluator
GetSymmetryMeasure

Figure 4.2: Detail of the symmetry detection classes

The *GenerateSymmetrySurface* method in the **HrudaModifiedAlgorithm** class takes care of generating and evaluating an initial set of spheres. It does so by first generating a set of radii - the minimal and maximal radii are user defined. Then, it simplifies the input point cloud to 300 points using the uniform grid implemented in the **PointGrid** class. A set of sphere centres is generated from the simplified point set, where each centre is positioned such that the surface of the sphere lies between each pair of points when considering one of the radii. The combination of the radii and centres creates the initial set of the tested spheres.

Each of these spheres is evaluated by reflecting the input point cloud over its surface and calculating the symmetry measure using the radial function implemented in the **WendlandsFunction3** class. The argument to this function is the distance between the reflected point and the point closest to this reflection in the input set. The sphere yielding the highest symmetry measure is then used as a starting sphere for the Nelder-Mead optimisation technique.

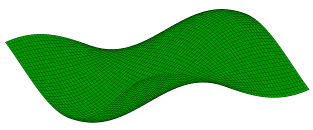
The optimisation is implemented in the *HrudaModifiedNelderMead* method. It uses the *GetSymmetryMeasure* method to evaluate the fitness of the sphere. The previous paragraph provided an explanation of how the symmetry measure is calculated, which represents the main point of modification in the algorithm. In the original implementation, this method reflected points over a plane, while the modified version uses a sphere as the reflecting surface.

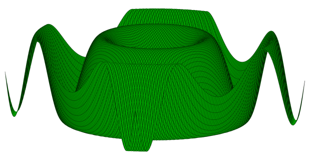
Modifying the algorithm requires appropriate changes to the *GetSymmetryMeasure* method. However, it is important to ensure that the other described methods remain consistent with this modification, i.e., the optimised variables must correspond to the symmetry measure evaluation.

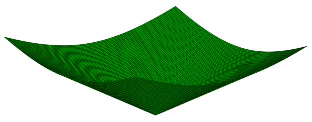
5 Experiments and Results

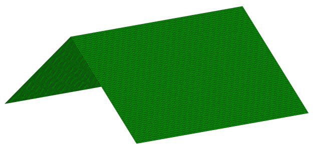
In this chapter results of methods introduced in sections 3.3 and 3.4 will be presented. Experiments with both artificially generated data and real data are included. Artificial data generation will also be explained. Apart from real objects, in some of the experiments, sampled surfaces given by functions listed below are used (see Fig. 5.1 for their visualisation):

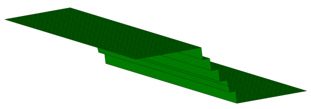
$$
\begin{aligned}
f_1(x,y) &= \sin(0.7x)\cos(0.7y)\\
f_2(x,y) &= \tfrac{\sin(0.7(x^2+y^2))}{0.7}\\
f_3(x,y) &= \sqrt{x^2+y^2}\\
f_4(x,y) &= 1-|y|\\
f_5(x,y) &= \tfrac{\operatorname{sgn}(-0.65-x)+\operatorname{sgn}(-0.35-x)+\operatorname{sgn}(-0.05-x)+\operatorname{sgn}(0.55-x)}{7}\\
f_6(x,y) &= \sqrt{1-x^2-y^2}
\end{aligned}
$$


(a) Graph of f_1


(b) Graph of f_2


(c) Graph of f_3


(d) Graph of f_4


(e) Graph of f_5

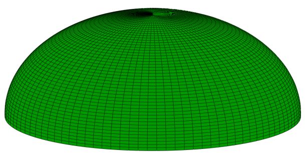

(f) Graph of f_6

Figure 5.1: Visualisation of the surfaces given by functions $f_1, \dots, f_6$ (rendered using GNU Octave)

We selected this set of surfaces to be experimented with, because they collectively exhibit varying degrees of curvature, and by using them as mirroring surfaces, we can create complex point sets. For example, the surface shown in Fig. 5.1a undergoes minor fluctuations in elevation, which makes it ideal for generating predictable and easily testable data. In contrast, the second surface (Fig. 5.1b) exhibits high frequency oscillations, making it more challenging for algorithms to process. The surfaces shown in Fig. 5.1c - 5.1e are not complex in shape, but they contain points, in which they are not differentiable, causing sharp transitions in curvature. Finally, the surface shown in Fig. 5.1e was selected to easily test the implementation of the algorithm searching for a spherical surface (Algorithm 4).

Apart from f_6, the domains of these functions are not restricted. In the case of the f_6 function, only its real values are considered for visualisation and use during data generation, i.e., only arguments $x, y \in \mathbb{R} : x^2 + y^2 \leq 1$ are assumed.

5.1 Artificial Data Generation

In order to be able to compare the results of the introduced algorithms, data for which the ground truth is known is needed. This section describes the process of generating different types of data sets, each suitable for different understanding of generalised symmetry, as was defined in Chapter 3.

5.1.1 Data Generation for Fixed Direction of Mapping

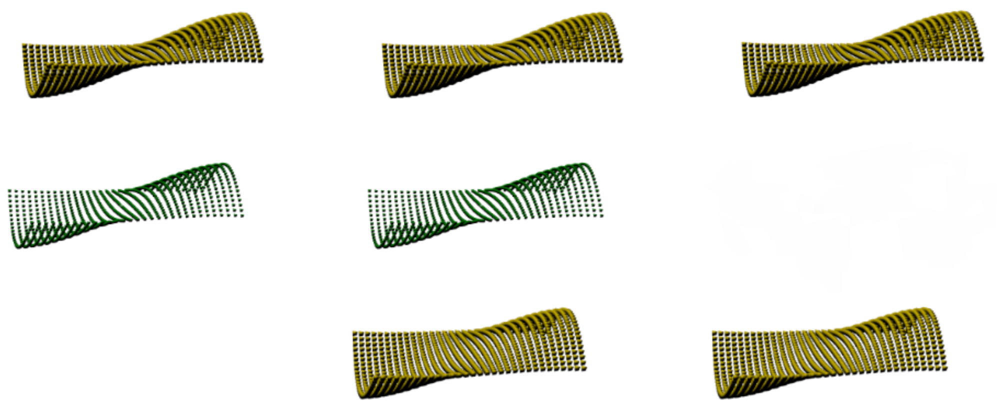
Artificial data generation is done by selecting a point cloud $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}, \mathbf{p}_i \in E^3, i = 1, \ldots, n$ and by mirroring this point cloud over the surface given by a function $f(x, y) = z$. As the vector $\mathbf{v}$, which determines the direction of mapping, we selected $\mathbf{v} = (0, 0, 1)^T$. By mirroring points $\mathbf{p}_i$ over the surface given by $f(x, y) = z$ using the vector $\mathbf{v}$, we get a new point set $\mathbf{P'} = \{\mathbf{p'_1}, \mathbf{p'_2}, \ldots, \mathbf{p'_n}\}, \mathbf{p'}_i \in E^3, i = 1, \ldots, n$ as

$$\mathbf{p'}_i = \mathbf{q} + (\mathbf{q} - \mathbf{p}_i),$$

where

$$\mathbf{q} = [\mathbf{p}_i^x, \mathbf{p}_i^y, f(\mathbf{p}_i^x, \mathbf{p}_i^y)].$$

The vector $(\mathbf{q} - \mathbf{p}_i)$ is parallel to the vector $\mathbf{v}$. It is expressed in this form to be of the correct length such that a point and its image are equally distant from the surface in this direction. The point set $\mathbf{X}$ used as input for the symmetry detection procedure is acquired as $\mathbf{X} = \mathbf{P} \cup \mathbf{P'}$. See Fig. 5.2 for the visualisation of the data generation.



(a) Input set **P** (yellow) and the mirroring surface f (green)

(b) Output of data generation

(c) Input set **X** for symmetry detection

Figure 5.2: Visualisation of data generation using fixed direction of mapping

5.1.2 Data Generation for Perpendicular Direction of Mapping over a General Surface

Data for perpendicular direction of mapping over a general surface is acquired by selecting a function $f(x, y) = z$, which generates a surface, and selecting uniformly sampled set of points $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n\}$ on the surface. To determine the normal in each point on the surface, we also input gradient ∇f of the function f. The points $\mathbf{s}_i$ lying on the surface given by f are then shifted in the direction of the normal (and in the opposite direction) to acquire two sets of shifted points $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ and $\mathbf{P}' = \{\mathbf{p}'_1, \mathbf{p}'_2, \ldots, \mathbf{p}'_n\}$. We denote the normal vector in point $\mathbf{s}_i$ as:

$$\mathbf{n}_i^{x,y} = \nabla f(\mathbf{s}_i^x, \mathbf{s}_i^y),$$

where $\mathbf{n}_i^{x,y}$ denotes the x and y coordinates of the normal vector and $\mathbf{n}_i^z = -1$. The numbers $\mathbf{s}_i^x$ and $\mathbf{s}_i^y$ represent the x and y coordinate of the point $\mathbf{s}_i$ respectively. The normalised form of $\mathbf{n}$ is denoted as $\widehat{\mathbf{n}}$. The calculation of points $\mathbf{p}_i$ and $\mathbf{p}'_i$ then becomes:

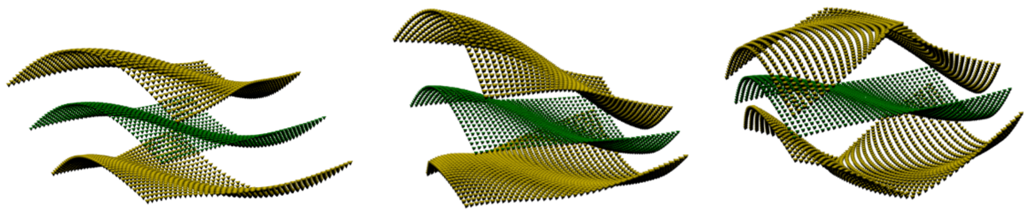
$$\mathbf{p}_i = \mathbf{s}_i + d_i\widehat{\mathbf{n}},$$

and

$$\mathbf{p}'_i = \mathbf{s}_i - d_i\widehat{\mathbf{n}},$$

where d_i represents the distance by which the point $\mathbf{s}_i$ is shifted away from the surface. The distance can be constant for all points, but it can also vary. Non-constant d_i can be used to introduce noise in the result data and it can also be used to create more complex data sets. See Fig. 5.3 in which an example of generated data is visible, both constant and non-constant parameters d_i are used.

As before, the point set $\mathbf{X}$ used as input for the symmetry detection procedure is acquired as $\mathbf{X} = \mathbf{P} \cup \mathbf{P}'$.



(a) Visualisation of the points (yellow) at a constant distance from the surface (green)

(b) Visualisation of the points (yellow) at a varying distance from the surface (green) - one "wave"

(c) Visualisation of the points (yellow) at a varying distance from the surface (green) - two "waves"

Figure 5.3: Visualisation of data generation using perpendicular direction of mapping

5.1.3 Data Generation for Perpendicular Direction of Mapping over a Spherical Surface

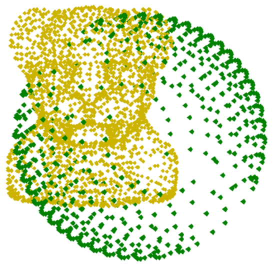
Data for perpendicular direction of mapping over a spherical surface is done by selecting a point cloud $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ and a sphere s given by its centre $\mathbf{c}_s$ and radius r_s and selecting a subset $\mathbf{Q} \subseteq \mathbf{P}$ of points $\mathbf{p}_i$ which are "inside" the sphere s, i.e., which satisfy the condition $\|\mathbf{p}_i - \mathbf{c}_s\| < r_s$. Points in $\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m\}$ are then mirrored over the surface of the sphere using the following formula:

$$\mathbf{q'}_i = \mathbf{q}_i + 2\|\mathbf{u}\| - r_s\hat{\mathbf{u}},$$

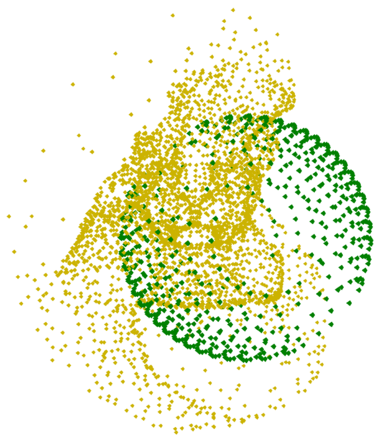
where $\mathbf{u} = \mathbf{c}_s - \mathbf{q}_i$ and points $\mathbf{q'}_i$ make a set $\mathbf{Q}'$.

The point set $\mathbf{X}$ used as input for the symmetry detection procedure is then acquired as $\mathbf{X} = \mathbf{Q} \cup \mathbf{Q}'$. See Fig. 5.4 for the visualisation of the data generation.

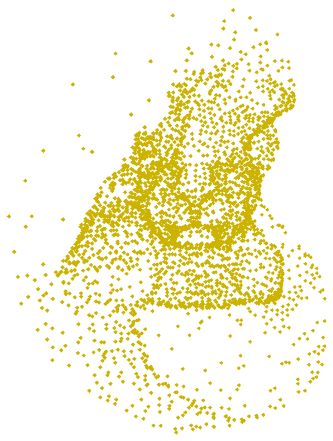
(a) Input set **P**



(b) Input set **P** (yellow) and sphere s (green)



(c) Input set **X** (yellow) and sphere s (green)



(d) Input set **X** for symmetry detection

Figure 5.4: Visualisation of data generation using a spherical surface

5.2 Results on Artificial Data

In this section, experiments on data to which the correct results are known will be presented. As an evaluation of the results, multiple values indicating the efficacy of the solution were computed. These values are the RMSE (Root Mean Square Error), the average error (e_{avg}) and the maximal error (e_{max}), which are calculated as:

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(x_i - f(x_i))^2}{n}}, \tag{5.1}$$

$$e_{avg} = \sum_{i=1}^{n} \frac{(x_i - f(x_i))}{n}, \tag{5.2}$$

$$e_{max} = \max_i(|x_i - f(x_i)|), \tag{5.3}$$

where f is a function of two variables generating the correct surface of symmetry and n is the number of points detected as points lying on the surface of symmetry.

In the case of Algorithm 4 presented in Section 5.2.4, the detected surface is a sphere given by its centre $\mathbf{c}$ and radius r, therefore, we calculate the differences between these values and those of the known correct sphere's s as:

$$e_c = \frac{\|s_{\mathbf{c}} - \mathbf{c}\|}{\text{diag}}, \tag{5.4}$$

$$e_r = \frac{|s_r - r|}{\text{diag}}, \tag{5.5}$$

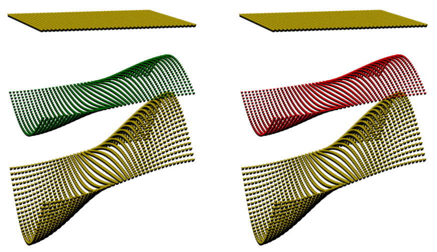
where $s_{\mathbf{c}}$ is the centre of the correct sphere, s_r is the radius of the correct sphere and diag is the length of the diagonal of the bounding box of the input point cloud.

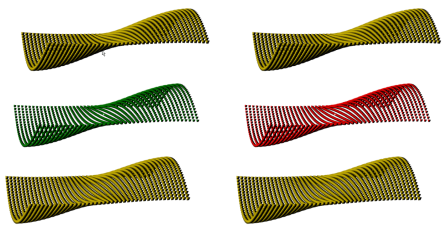
In the figures included in the following subsections, the yellow points are the input points, the green points are sampled on the correct surface of symmetry, i.e., the surface used for mirroring the point cloud and the red points are the detected points lying on the surface of symmetry. The tables list errors of the results using Equations 5.1, 5.2 and 5.3 in Sections 5.2.1, 5.2.2 and 5.2.3. In Section 5.2.4, the table listing the errors used Equations 5.4 and 5.5.

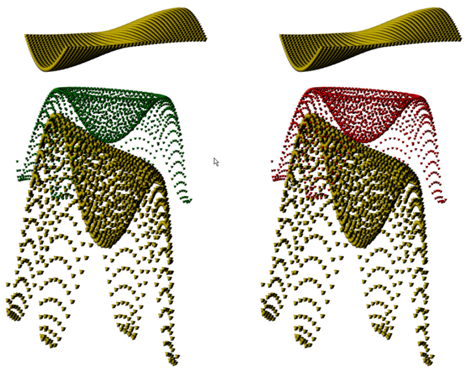
5.2.1 Separation Algorithm

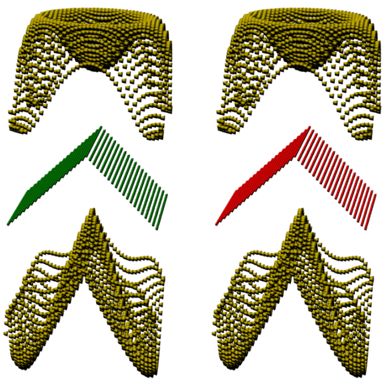
The results of the Separation Algorithm (Algorithm 1) on data generated using the method described in Section 5.1.1 are visible in Fig. 5.5 and the computed errors are listed in Tab. 5.1.

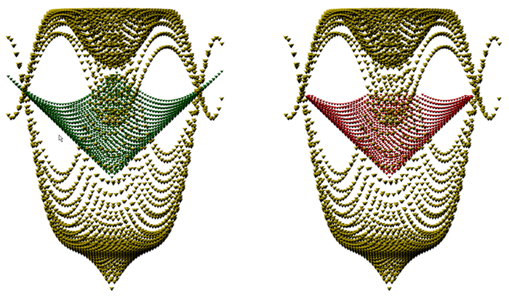
Both the visual comparison of the correct and the detected surfaces in Fig. 5.5 and the computed errors in Tab. 5.1 indicate that this algorithm performs very well on data, which were purposely generated using a fixed direction of mapping. Most of the point clouds generated for this set of experiments were chosen to be easily separable by a plane, because the focus of this assessment was in different types and degrees of deformation of the mirrored point sets and whether or not would this algorithm select the correct mapping between the subsets of points. However, even in the case of the result shown in Fig. 5.5e where the input subsets are not separable by a plane, the algorithm was still able to generate the correct output. In


(a) Experiment 1: Slightly curved surface


(b) Experiment 2: Slightly curved surface


(c) Experiment 3: Highly curved surface


(d) Experiment 4: Surface with a sharp edge


(e) Experiment 5: Surface with a sharp point

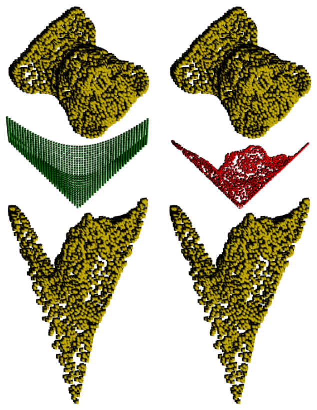

(f) Experiment 6: Mirroring real data

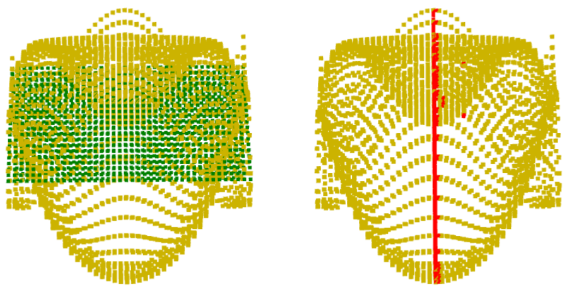
Figure 5.5: The results of the Separation Algorithm on artificial data

	RMSE	$\mathbf{e_{max}}$	$\mathbf{e_{avg}}$
Experiment 1	5.47E-8	1.19E-7	3.97E-8
Experiment 2	4.77E-8	1.19E-7	3.65E-8
Experiment 3	5.80E-8	2.38E-7	3.04E-8
Experiment 4	1.04E-7	2.38E-7	7.55E-8
Experiment 5	3.09E-8	1.19E-7	1.14E-8
Experiment 6	2.74E-8	1.32E-8	1.19E-7
Experiment 7	1.05	0.88	1.99

Table 5.1: Errors of the Separation Algorithm on artificial data

Fig. 5.5f a real object is mirrored over the selected surface. The reason for including this experiment is to show that it is possible to detect the correct surface even for deformed "closed" objects.

To see the limitations of this algorithm, refer to Fig. 5.6. The used surfaces are the same as were shown in Fig. 5.5d, only the distance of the yellow points from the mirroring surface is different, in this case it is smaller. As a result of this, the point set cannot be easily divided into two distinct subsets using a plane and consequently, the detected surface is incorrect and resembles a plane of symmetry.



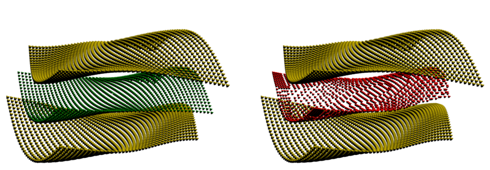
(a) Experiment 7

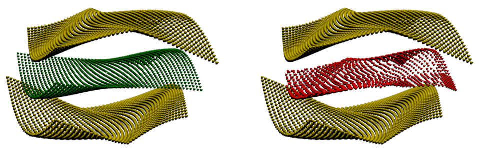
Figure 5.6: Incorrect results of the Separation Algorithm due to inseparability of the point cloud

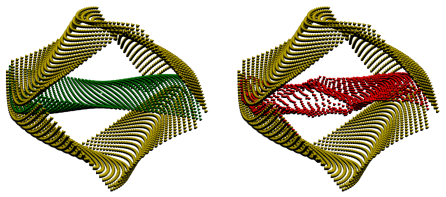
The need to separate the input into two subsets can be seen as a drawback or an advantage of this algorithm, depending on the input data. In this artificial setting, the algorithm typically performs very well, because the data were generated using a technique that is well-suited to the algorithm's way of operating. However, in more realistic scenarios, this separation requirement may become a disadvantage, since the input data may have a higher level of complexity or noise, making it more difficult for the algorithm to accurately partition the data into two subsets.

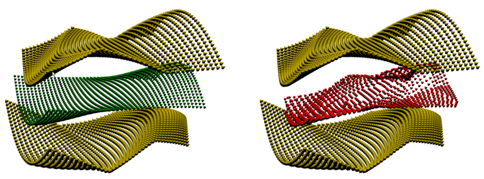
5.2.2 Separation Algorithm with Normal Estimation

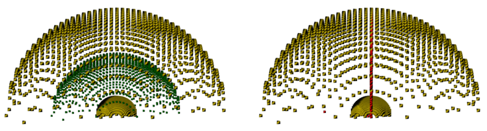
The results of the Separation Algorithm using normal estimation (Algorithm 2) on data generated using the method described in Section 5.1.2 are visible in Fig. 5.7 and the computed errors are listed in Tab. 5.2.


(a) Experiment 1: Constant distance


(b) Experiment 2: Non-constant distance


(c) Experiment 3: Non-constant distance


(d) Experiment 4: Non-constant distance


(e) Experiment 5: Inseparable data

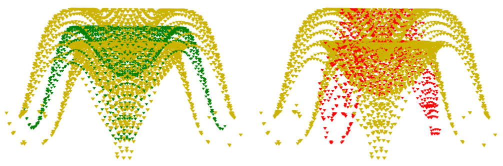

(f) Experiment 6: Inseparable data

Figure 5.7: The results of the Separation Algorithm using normal estimation on artificial data

It is noticeable that the results shown in Fig. 5.7 are not identical to the correct solutions, however, most of the surfaces are similar in their shape and position relative to the ground truth surfaces. In Figures 5.7b - 5.7d, the perpendicular distance of points from the surface of symmetry is not constant, which causes the detected surface to slightly deviate from the correct one.

	RMSE	$\mathbf{e_{max}}$	$\mathbf{e_{avg}}$
Experiment 1	0.01	0.05	0.01
Experiment 2	0.04	0.11	0.03
Experiment 3	0.19	0.88	0.14
Experiment 4	0.06	0.16	0.05
Experiment 5	0.65	1.06	0.57
Experiment 6	1.05	3.12	0.85

Table 5.2: Errors of the Separation Algorithm using normal estimation on artificial data

To demonstrate why the normal estimation is added to the Separation Algorithm, a result acquired from the input shown in Fig. 5.7c is included in Fig. 5.8. In this case, the estimated normals were not taken into account when searching for the surface of symmetry and, therefore, the detected surface is incorrect.

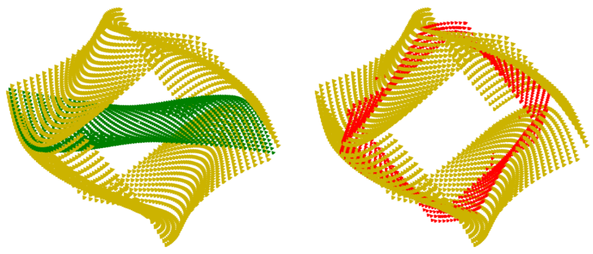


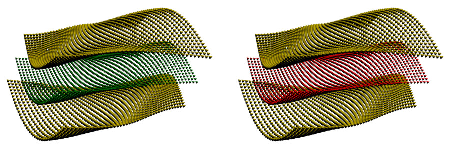
Figure 5.8: The results of the Separation Algorithm without taking the normals of points into account

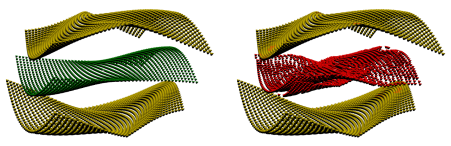
As the input subsets shown in Fig. 5.7e are not separable by a plane, the result is incorrect and resembles a plane of symmetry (notice the line of red points in the middle of the image on the right). This is similar to the case presented in Fig. 5.6. Fig. 5.7f also shows an inaccurate outcome, which is due to the same issue of the point cloud not being separable by a plane.

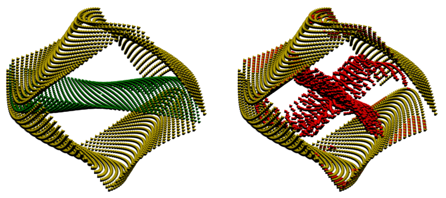
The main difference between this algorithm and the Separation Algorithm is in the type of data that they are designed to work with. Unlike the Separation Algorithm which is built for detecting a surface of symmetry assuming a fixed direction of mapping, this algorithm assumes perpendicular direction of mapping in the input point set.

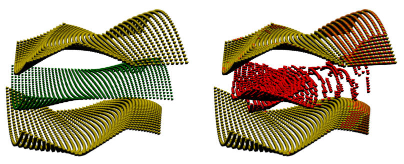
5.2.3 Algorithm using Normal Estimation Only

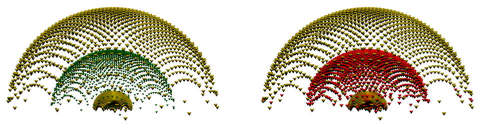
The results of the algorithm using normal estimation only (Algorithm 3) on data generated using the method described in Section 5.1.2 are visible in Fig. 5.9 and the computed errors are listed in Tab. 5.3.


(a) Experiment 1: Constant distance


(b) Experiment 2: Non-constant distance


(c) Experiment 3: Non-constant distance


(d) Experiment 4: Non-constant distance


(e) Experiment 5: Inseparable data

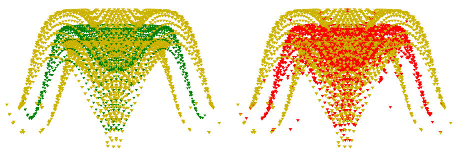

(f) Experiment 6: Inseparable data

Figure 5.9: The results of the algorithm using normal estimation only on artificial data

This set of the tested data is the same as the one presented in Section 5.2.2. Point clouds visible in Fig. 5.9b - 5.9d were generated by shifting points on the surface of symmetry in the normal direction by non-constant distance. This fact causes the algorithm to fail to generate the correct results, because it searches for the images of points in the direction of the estimated normal and it assumes the distance between each pair of points to be constant. On the other hand, the result shown in Fig. 5.9e is much more similar to the correct surface compared to the result obtained by Algorithm 2 (shown in Fig. 5.7e), however, the errors are still quite large even in this

	RMSE	$\mathbf{e_{max}}$	$\mathbf{e_{avg}}$
Experiment 1	0.09	1.06	0.01
Experiment 2	0.29	1.53	0.14
Experiment 3	0.72	2.30	0.56
Experiment 4	0.74	1.69	0.47
Experiment 5	0.42	0.94	0.24
Experiment 6	0.40	2.55	0.22

Table 5.3: Errors of the algorithm using normal estimation only on artificial data

case, because the surface is not identical to the correct one (see Fig. 5.10 for the detail of the detected surface). There is a smaller hemisphere circled in yellow inside the larger one, this is caused by incorrectly selected images of points. The result presented in Fig. 5.9f is also much more similar to the correct output compared to the result shown in Fig. 5.7f, because this algorithm does not rely on the separation of the point set.

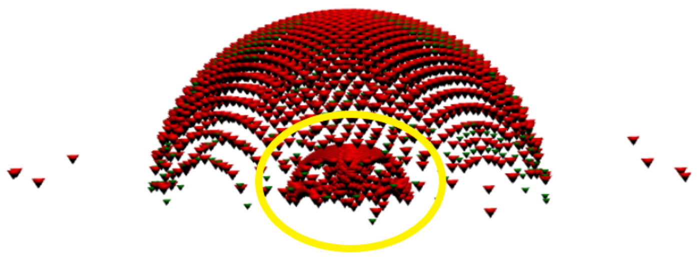


Figure 5.10: Experiment 5 - detail of the detected surface

The main advantage of this method is that it does not rely on any separating plane, which makes it more suitable for the use on some types of realistic models (see Section 5.4). On the other hand, it relies on a strong assumption that the perpendicular distance of points from the surface of symmetry is constant. As the presented experiments have demonstrated, it is relatively simple to generate data that can cause the algorithm to fail or produce inaccurate results by introducing non-constant distance of points from the surface.

5.2.4 Algorithm Modifying a Method of Planar Symmetry Detection

The results of Algorithm 4 on data generated using the method described in Section 5.1.3 are visible in Fig. 5.11 and the computed errors are listed in Tab. 5.4.

The radii of the spheres used for generating the input data in Figures 5.11b - 5.11e all have $r = \frac{\text{diag}}{4}$, where diag is the length of the diagonal that spans the axis-aligned bounding box (AABB) which encloses the original point cloud. The range of the tested radii was set to $(\frac{\text{diag}_d}{3}, \frac{\text{diag}_d}{5})$ for all of the experiments, where diag_d is the AABB diagonal of the deformed point cloud. In the case of Fig. 5.11a, the correct radius is $r = 1$ and the range of tested radii was the same as for the rest of the experiments.

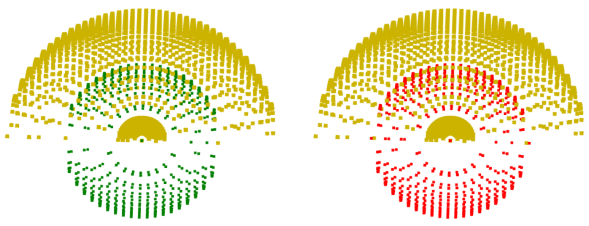
The results of this method all have low error rates and the visual comparison of the detected and the correct spheres also suggests that this algorithm performed very well on these data sets. The result is correct even in the case of the experiment shown in Fig. 5.11d, where the lengths of the AABB diagonals of the original and the deformed point clouds differ significantly. Therefore, the $r = \frac{\text{diag}}{4}$ (the correct radius) is outside the range of the $(\frac{\text{diag}_d}{3}, \frac{\text{diag}_d}{5})$ interval. Despite this fact, the optimisation method was able to converge to the correct result.

The advantage of this method is that it merely modifies an extensively tested and robust algorithm for planar symmetry detection. As a consequence, the results of the modified algorithm often converge to a solution that is very close to the global optimum.

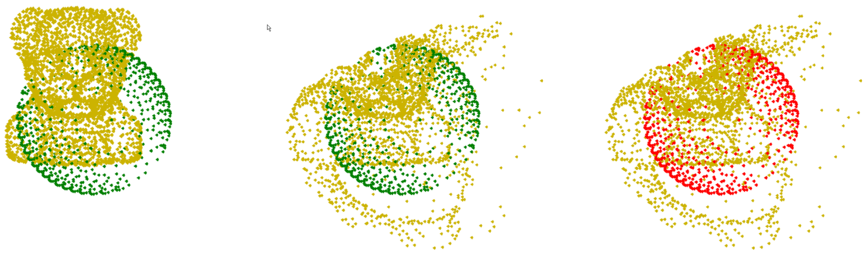
The disadvantage of this algorithm is that it requires choosing the surface of symmetry that will be searched.

	$\mathbf{e_c}$	$\mathbf{e_r}$
Experiment 1	2.98E-4	5.41E-4
Experiment 2	1.24E-4	1.98E-4
Experiment 3	2.99E-5	2.52E-5
Experiment 4	9.12E-4	6.73E-4
Experiment 5	1.92E-3	1.35E-3

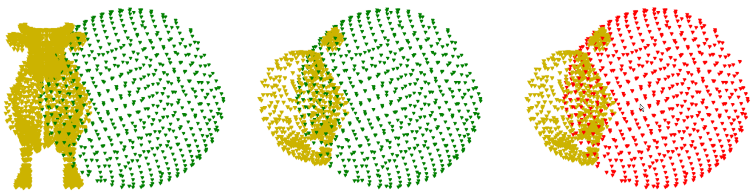
Table 5.4: Errors of the algorithm modifying a method of planar symmetry detection on artificial data



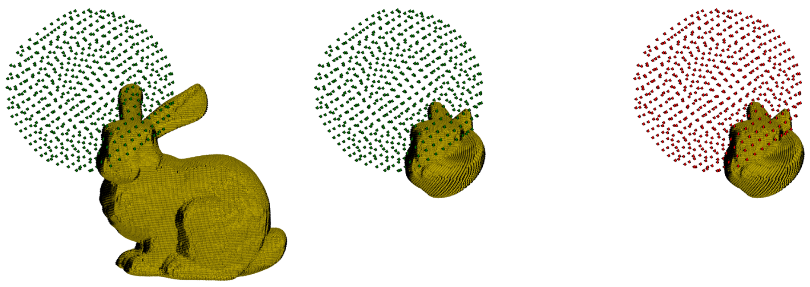
(a) Experiment 1: Hemisphere



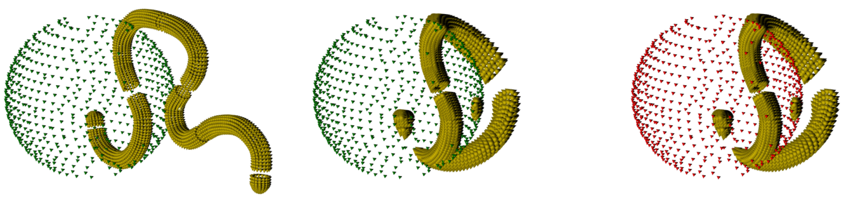
(b) Experiment 2: Lion



(c) Experiment 3: Cow



(d) Experiment 4: Bunny



(e) Experiment 5: Doodle

Figure 5.11: The results of the algorithm modifying a method of planar symmetry detection on artificial data

5.3 Results on Damaged Data

In this section, experiments that evaluate the robustness of the introduced algorithms to various types of point cloud damage will be presented. Specifically, we consider four types of damage: random deviation of points from their correct positions, removing compact parts of the point clouds, making the point density uneven in the point clouds, and rotation of the point clouds. The damage types of rotation and forced uneven density of points are included, because Algorithms 1 and 2 rely on differently rotated planes passing through the centroid of the point cloud, and thus, the performance can be affected by point cloud rotation and uneven sampling.

The results of the experiments presented in the following sections are listed in Tab. 5.5. The listed errors were calculated using Equations 5.1, 5.2, 5.3, 5.4 and 5.5.

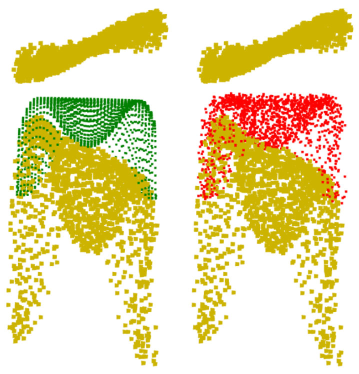
Algorithm	**Damage type**	**Experiment**	**RMSE**	$\mathbf{e_{max}}$	$\mathbf{e_{avg}}$	$\mathbf{e_c}$	$\mathbf{e_r}$
Separation Algorithm	Random noise	Experiment 1	0.30	1.14	0.22	-	-
		Experiment 2	0.18	0.86	0.10	-	-
	Removed part	Experiment 1	4.43E-8	1.19E-7	3.38E-8	-	-
	Uneven points	Experiment 1	2.01	2.77	1.99	-	-
		Experiment 2	1.32E-3	4.87E-3	9.22E-4	-	-
	Rotation	Experiment 1	-	-	-	-	-
Separation Algorithm with Normal Estimation	Random noise	Experiment 1	0.22	1.09	0.16	-	-
		Experiment 2	0.22	1.11	0.15	-	-
	Removed part	Experiment 1	1.12E-2	4.93E-2	7.97E-3	-	-
	Uneven points	Experiment 1	0.89	1.12	0.74	-	-
		Experiment 2	0.15	1.20	2.28E-2	-	-
	Rotation	Experiment 1	-	-	-	-	-
Algorithm using Normal Estimation Only	Random noise	Experiment 1	0.99	1.57	0.89	-	-
		Experiment 2	0.71	1.58	0.47	-	-
	Removed part	Experiment 1	0.36	1.17	0.12	-	-
	Uneven points	Experiment 1	0.87	1.22	0.68	-	-
		Experiment 2	0.59	1.22	0.31	-	-
Algorithm Modifying a Method of Planar Symmetry Detection	Random noise	Experiment 1	-	-	-	0.30	5.18E-2
		Experiment 2	-	-	-	0.38	0.18
	Removed part	Experiment 1	-	-	-	7.43E-5	7.86E-5
		Experiment 2	-	-	-	0.29	0.32
	Uneven points	Experiment 1	-	-	-	1.67E-4	1.41E-4
		Experiment 2	-	-	-	2.22E-4	2.05E-4

Table 5.5: Results of the experimnts performed on noisy data

5.3.1 Separation Algorithm

In this section, the results of experiments with the Separation Algorithm carried out on damaged data are presented.

In Fig. 5.12 there are two experiments on data to which random noise was introduced. In the experiment on the left, the whole domain was affected by the noise, whereas in the one on the right, only a part was affected. It is apparent that the surface was detected quite accurately, however, the noise causes the surface to be noisy as well in the parts where it is present in the input data. Comparing the errors of these experiments in Tab. 5.5 shows that the result obtained on the partially noisy data is more precise, because there are more detected points closer to the correct surface.


(a) Experiment 1: Random noise

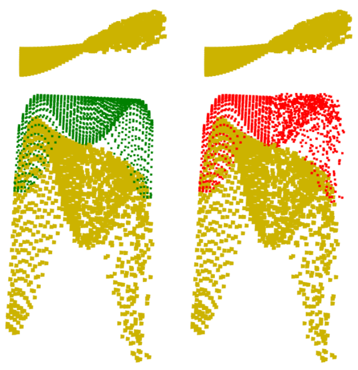

(b) Experiment 2: Random noise

Figure 5.12: The results of the Separation Algorithm on data with random noise

In the experiment shown in Fig. 5.13 a part of the input was removed completely. We can see that there is a segment of the detected points missing as well, however, the rest was detected very accurately, which is supported by the errors shown in Tab. 5.5.

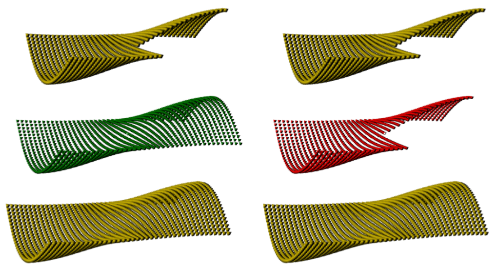
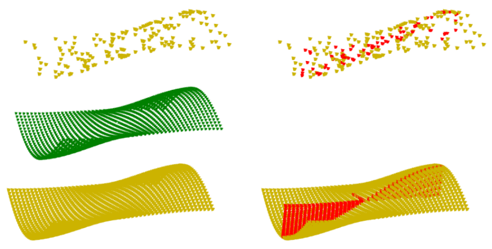

(a) Experiment 1: Missing part

Figure 5.13: The results of the Separation Algorithm on data with missing parts

In Fig. 5.14, the density of the input points is uneven. In Experiment 2 (on the right), the input was the same as in Experiment 1 (on the left), but the number of points was reduced to one quarter of the original amount using the uniform grid described in Section 2.2.4. When the downsampling is not used and the variability of the point density is this high in the input data, this algorithm seldom produces accurate results since the centroid of the point cloud is moved towards the higher density area, see Fig. 5.14a. When the density is at least partially equalised, the results can be much more precise, see Fig. 5.14b.


(a) Experiment 1: Uneven point density

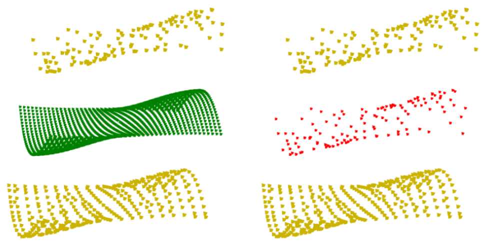

(b) Experiment 2: Uneven point density (downsampled)

Figure 5.14: The results of the Separation Algorithm on data with uneven point density

For this algorithm, an experiment with rotated input points is included, because it depends on a separation by a plane, which must be conveniently rotated to separate the point cloud correctly in order to find the correct mapping between the points. In Fig. 5.15, the result of the experiment is shown. Visually, the detected surface is very similar to the correct one.

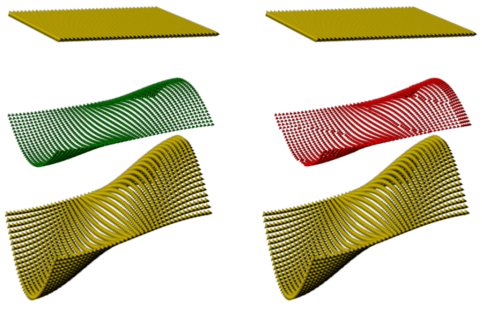

(a) Experiment 1: Rotation

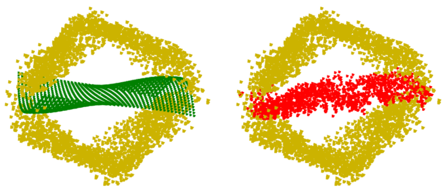
Figure 5.15: The results of the Separation Algorithm on rotated data

Among all of the tested types of noise, uneven sampling of the input data seems to be the factor that has the greatest negative impact on the performance of this algorithm.

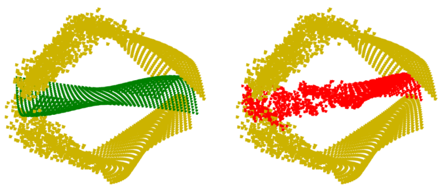
5.3.2 Separation Algorithm with Normal Estimation

In this section, the results of experiments with the Separation Algorithm with added normal estimation carried out on damaged data are presented.

In Fig. 5.16 random noise affecting the whole input set (left) and a part of the input set (right) was added to the data. The results are very similar to the ones presented in Fig. 5.12 - the detected points lie close to the correct surface, but they are noisy, because of the corruption of the input set.



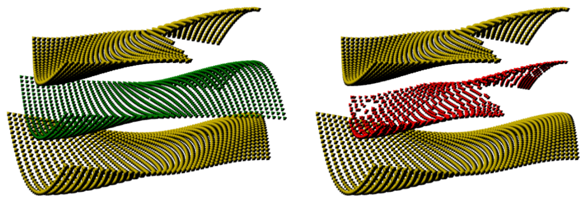
(a) Experiment 1: Random noise



(b) Experiment 2: Random noise

Figure 5.16: The results of the Separation Algorithm with added normal estimation on data with random noise

In Fig. 5.17, the results of this algorithm on data with a missing part are shown. Again, the algorithm does not seem to be highly negatively affected by this type of damage to the data (see the errors in Tab. 5.5).

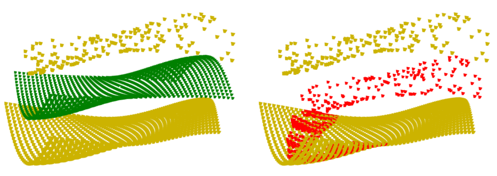


(a) Experiment 1: Missing part

Figure 5.17: The results of the Separation Algorithm with added normal estimation on data with missing parts

In Fig. 5.18 results on data with uneven point density are shown. Unlike the results of Algorithm 1 shown in Fig. 5.14a, here, they are much more precise even in the case of high variability of points density (Fig. 5.18a) - compare the results in Tab. 5.5. This is caused by the added information about the normals of the points which helped to pair the points with their images more accurately.

In Fig. 5.19 an experiment on rotated data is presented. The results are very similar to the ones shown in 5.15. While the rotation of the input set


(a) Experiment 1: Uneven point density

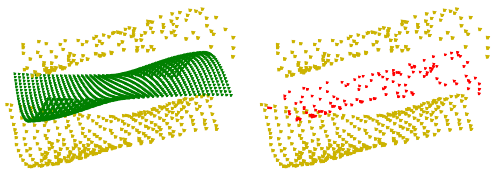

(b) Experiment 2: Uneven point density (downsampled)

Figure 5.18: The results of the Separation Algorithm with added normal estimation on data with uneven point density

in the coordinate system does affect the output of both Algorithm 1 and 2, the change in the results is slight and the detected surface is very similar to the correct one.

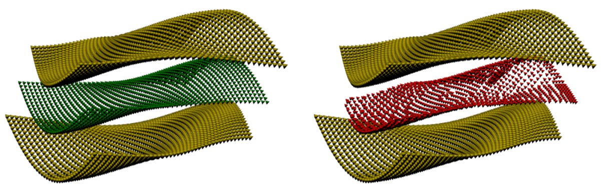

(a) Experiment 1: Rotation

Figure 5.19: The results of the Separation Algorithm with added normal estimation on rotated data

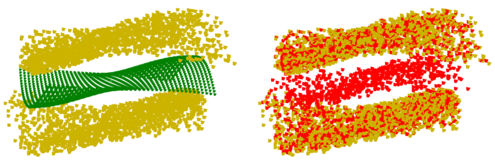
Overall, the way this algorithm responds to different inputs is very similar to Algorithm 1, however, it seems to be more robust to high variability of points density in the input.

5.3.3 Algorithm using Normal Estimation Only

In this section, the results of experiments with the algorithm using only the estimation of normals of points (Algorithm 3) carried out on damaged data are presented.

In Fig. 5.20, results on input with random noise are presented. In both experiments, a lot of symmetry surface points (red) were detected “very close” to the noisy part of the input. This is caused by the fact that this algorithm relies solely on the information about the normals of the input points and in this case, the normals are not estimated accurately due to the perturbation of the point cloud.

In Fig. 5.21 the result on input with a missing part is shown. Much like in the case of the experiments presented in Fig. 5.20, some of the surface


(a) Experiment 1: Random noise

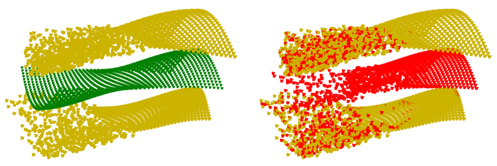

(b) Experiment 2: Random noise

Figure 5.20: The results of the algorithm using normal estimation only on data with random noise

points lie in near proximity of the input. In this case, the cause of this is the missing area in the upper part of the input - the lines constructed using the estimated normals in this area do not intersect the upper half of the point cloud, therefore, the closest projections of points are the ones "beneath" the missing part.

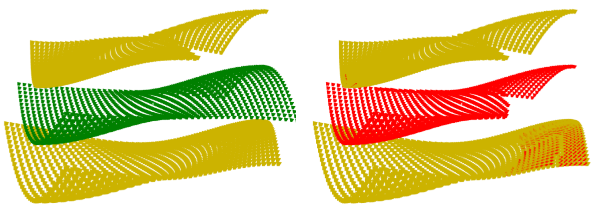
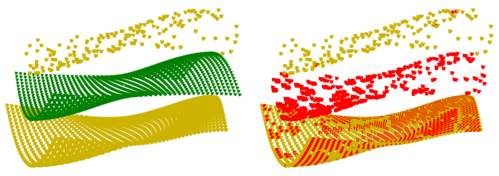

(a) Experiment 1: Missing part

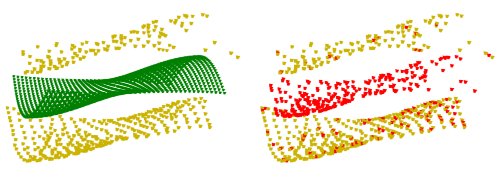
Figure 5.21: The results of the algorithm using normal estimation only on data with missing parts

In Fig. 5.22 experiments with uneven points distribution are shown. There are again points detected very close to the input point cloud, especially in the higher density areas. The reason for this is similar to the case of a missing part - there are not enough images for the points in the high density area, therefore, the images end up being detected close to those points.

Since this algorithm does not attempt to separate the input set into two sets, it often detects some symmetry surface points incorrectly when the input is not uniformly sampled or when there are parts missing completely. Despite this fact, it still detects a satisfactory number of points close to the correct surface of symmetry.



(a) Experiment 1: Uneven point density



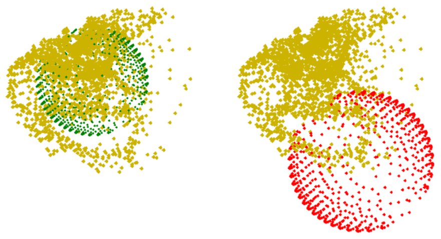
(b) Experiment 2: Uneven point density (downsampled)

Figure 5.22: The results of the algorithm using normal estimation only on data with uneven point density

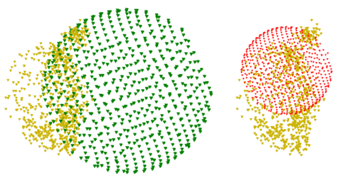
5.3.4 Algorithm Modifying a Method of Planar Symmetry Detection

In this section, the results of experiments with the algorithm modifying a method of planar symmetry detection (Algorithm 4) carried out on damaged data are presented.

Fig. 5.23 shows experiments with point clouds affected by random noise. Both the visual comparison of the results and the errors listed in Tab. 5.5 indicate that the spherical surfaces were detected incorrectly. This fact is quite surprising, since the original method is quite robust to this type of noise. It may be caused by a different strategy of candidate surfaces selection or the difference in the optimisation method which in our case is not gradient-based (see Section 2.2.4).



(a) Experiment 1: Random noise

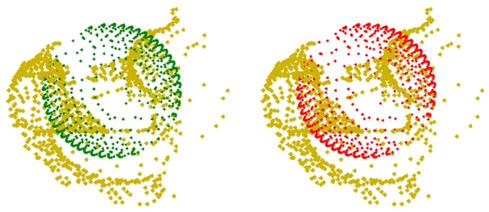


(b) Experiment 2: Random noise

Figure 5.23: The results of the algorithm modifying a method of planar symmetry detection on data with random noise

Results presented in Fig. 5.24 were obtained on data with a missing section of points. In the case of Experiment 1, the result is very precise, however, the spherical surface detected in Experiment 2 is incorrect.

In Fig. 5.25, experiments with uneven points distribution are shown. In both of the inputs, the correct surface was detected.


(a) Experiment 1: Missing part

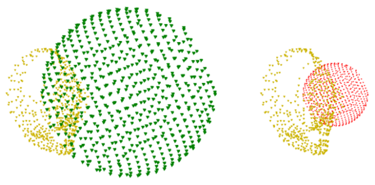
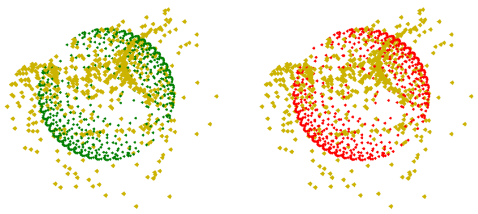

(b) Experiment 2: Missing part

Figure 5.24: The results of the algorithm modifying a method of planar symmetry detection on data with missing parts


(a) Experiment 1: Uneven point density

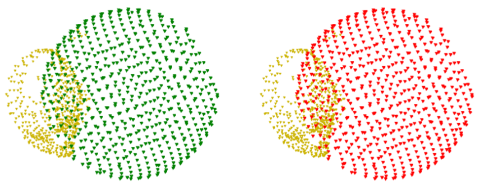

(b) Experiment 2: Uneven point density

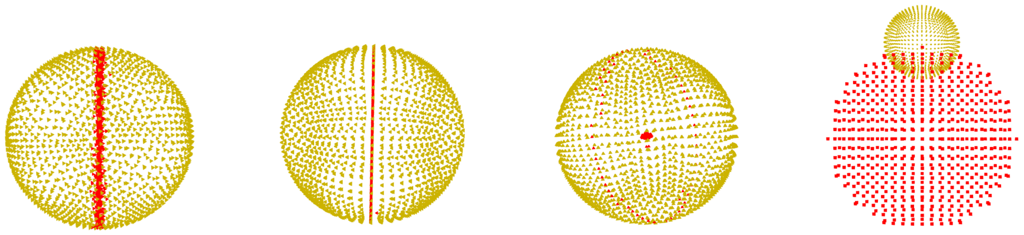
Figure 5.25: The results of the algorithm modifying a method of planar symmetry detection on data with uneven point density

This algorithm can sometimes detect the spherical surfaces of symmetry incorrectly when working with damaged data. Identifying the correct outcome is often hindered by the presence of random noise in the data.

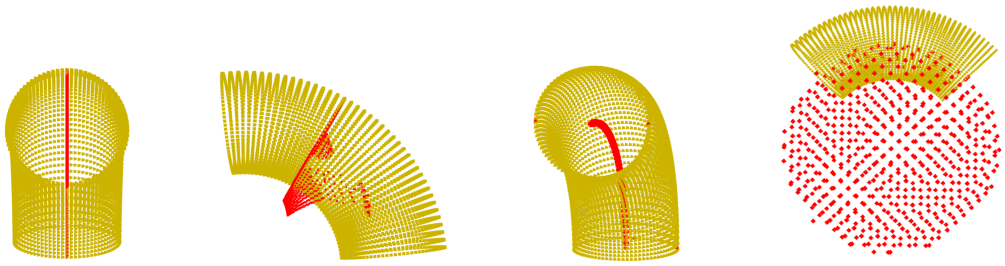
5.4 Results on Real Data

In this section, we present experiments conducted on data to which the correct result is unknown. Although we do not have a way to evaluate the quality of the results objectively, these experiments are important because they provide a way to gain a better understanding of how the algorithms behave in a realistic setting.

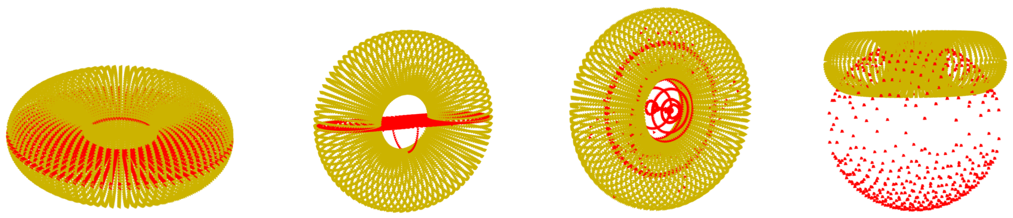
The results obtained by using each of the presented algorithms are shown next to one another for easier comparison in Figures 5.26 and 5.27. The objects may be shown from different angles for more convenient visualisation of the detected surfaces of symmetry. In each of the presented figures, the results are presented from left to right in the following order of algorithms: 1, 2, 3, and 4.



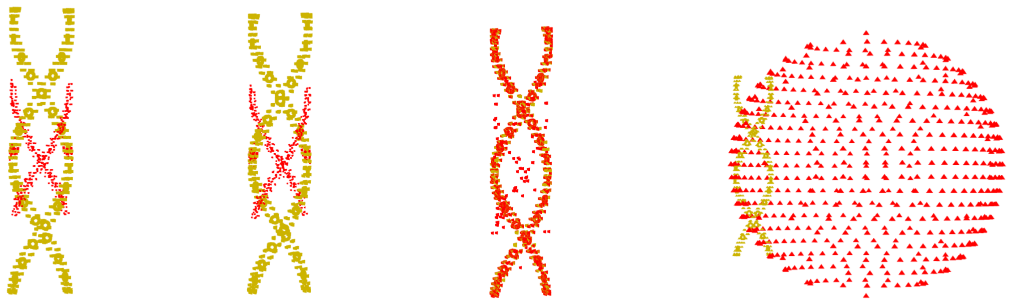
(a) Sphere



(b) Pipe



(c) Torus



(d) DNA

Figure 5.26: The results on real objects

In Fig. 5.26a, Algorithms 1 and 2 both detected points lying close to a plane of symmetry of the object. Algorithm 3 detected a point inside the sphere, which was anticipated, given its operating principles. It also detected a belt of points going along the surface of the sphere, this was caused by wrongly estimated normals in those points. In the case of the result obtained by Algorithm 4, the initial radius of the detected spherical surface was set to the length of the object's AABB diagonal. The detected spherical surface passes roughly through the middle of the object, which can also be considered an expected result.

Fig. 5.26b shows experiments performed on a bent object. Again, Algorithm 1 detected points near the object's symmetry plane. Algorithm 3 detected a "bent axis" of the object, which is a very satisfactory result, as it accurately characterises the object's shape. For Algorithm 4, the initial radius of the detected sphere was set to one half of the length of the object's AABB diagonal. The surface of the detected sphere roughly copies the shape of the object.

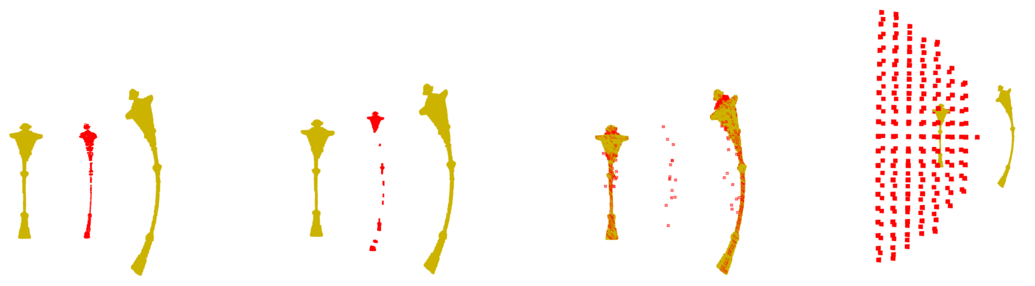
Fig. 5.26c shows a set of experiments conducted on the torus object. It is worth noting that both Algorithm 1 and 2 detected points in proximity to a symmetry plane, but each near a different symmetry plane. Algorithm 3 detected a number of circular "curves", one of which goes through the inside of the torus. The initial range of the tested radii for Algorithm 4 for the detected sphere was set from one fifth to one third of the length of the object's AABB diagonal. The detected sphere does not seem to be significantly accurate.

In Fig. 5.26d, the results obtained by Algorithms 1 and 2 are very similar. Most of the points detected by Algorithm 3 lie near the input points. The surface of sphere detected by Algorithm 4 encompasses almost the whole point cloud, which is a surprising outcome. The initial radius was set to two thirds of the object's AABB diagonal.

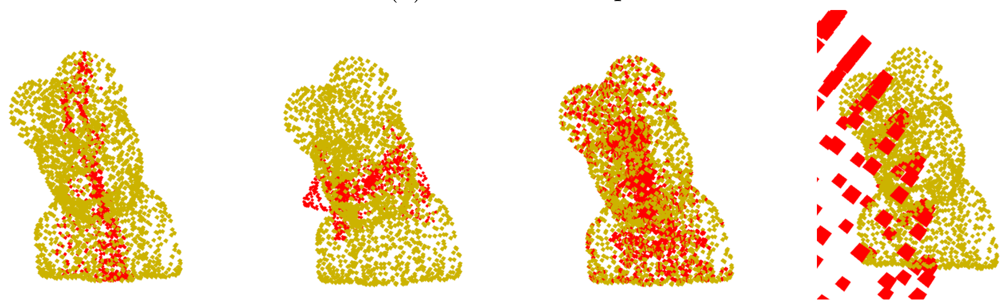
In the next set of experiments shown in Fig. 5.27, objects were deformed using the procedure described in 5.1.3 and used as an input for each of the presented algorithms.

In Fig. 5.27a, Algorithms 1 and 2 formed a set of points resembling an average of the two subsets. This set of points seems to describe the mapping between the subsets rather well. The sphere detected by Algorithm 4 also copies the curvature of the deformed object successfully.

In the case of the object shown in Fig. 5.27b, the set of points detected by Algorithms 1, 2 resembles the results shown in Fig. 5.26c, i.e., the detected points lie along different "axes" of the object. Algorithm 3 detected a cloud of points inside the input object, which is not a very successful outcome. As far as the result produced by 4 is concerned, the sphere was placed very accurately to describe the deformation of the object.



(a) Deformed lamp



(b) Deformed lion

Figure 5.27: The results on real deformed objects

5.5 Evaluation of the Algorithms

The algorithms presented in Sections 3.3.1, 3.4.2, 3.4.3 and 3.4.4 were tested and evaluated on various types of data in Sections 5.2, 5.3 and 5.4. In the following text, we will provide a summary of the experimental findings.

Algorithm 1 seems to be best suited for point clouds, which can be clearly separated by a plane into two subsets. It often fails when the point density is highly uneven and when the set is not separable.

Algorithm 2 is also best suited for data separable by a plane, however, it is not as sensitive to uneven point distribution in the input.

Algorithm 3 performs effectively on point cloud data that represent "hollow" objects or sampled surfaces in general. Since this algorithm heavily depends on estimated normals in the input point set, it is preferable for the point set to be free of noise.

Finally, Algorithm 4 usually yields satisfactory results, provided the range of the tested radii is set appropriately. It exhibits good performance on various inputs, although it is not entirely robust to noise in the data.

6 Conclusion

In this thesis, the problem of generalised mirror symmetry was defined and formalised. The definition relies on two different approaches to mirroring points over curved surfaces, it either considers a fixed direction of mapping, where every point is mirrored in the same direction, alternatively, an approach of perpendicular direction of mapping was considered, causing the points to be mirrored perpendicularly to the surface.

In the practical part of the work, four algorithms were presented and tested on various types of data. The results of the performed tests show that each of the proposed methods is suitable for different types of input point sets.

One of the aims of this work was to modify an existing method of planar symmetry detection to search for a spherical surface of symmetry. The modification of the original algorithm was successful and future research may focus on further modification of this method to consider other than spherical surface as the searched surface of symmetry. It is expected that there will be further work within the GAČR project that will expand on the methods proposed in this thesis.

Bibliography

[1] Accord.NET Machine Learning Framework. Available at: `http://accord-framework.net/`.

[2] Fermat's principle. Available at: `https://www.britannica.com/science/Fermats-principle`.

[3] Introduction to Numerical Analysis: Polynomial Interpolation. Available at: `https://engcourses-uofa.ca/introduction-to-numerical-analysis/polynomial-interpolation/`.

[4] Reflection and refraction. Available at: `https://www.britannica.com/science/light/Reflection-and-refraction`.

[5] Amenta, N. – Bern, M. – Kamvysselis, M. A new Voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, s. 415–421, 1998.

[6] Amenta, N. – Choi, S. – Kolluri, R. K. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, s. 249–266, 2001.

[7] Bokeloh, M. et al. Symmetry detection using feature lines. In *Computer Graphics Forum*, 28, s. 697–706. Wiley Online Library, 2009.

[8] Cervenka, M. – Skala, V. Behavioral Study of Various Radial Basis Functions for Approximation and Interpolation Purposes. In *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, s. 135–140, 2020. doi: 10.1109/SAMI48414.2020.9108712.

[9] Combes, B. et al. Automatic symmetry plane estimation of bilateral objects in point clouds. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, s. 1–8, 2008. doi: 10.1109/CVPR.2008.4587605.

[10] Eberly, L. E. Multiple linear regression. *Topics in Biostatistics*. 2007, s. 165–187.

[11] Hruda, L. – Kolingerová, I. – Váša, L. Robust, fast and flexible symmetry plane detection based on differentiable symmetry measure. *The Visual Computer*. 2022, 38, 2, s. 555–571.

[12] Jiang, W. et al. Skeleton-based intrinsic symmetry detection on point clouds. *Graphical Models*. 2013, 75, 4, s. 177–188. ISSN 1524-0703. doi:

https://doi.org/10.1016/j.gmod.2013.03.001. Available at: `https://www.sciencedirect.com/science/article/pii/S1524070313000118`.

[13] Khatamian, A. – Arabnia, H. R. Survey on 3D surface reconstruction. *Journal of Information Processing Systems*. 2016, 12, 3, s. 338–357.

[14] Lipman, Y. et al. Symmetry Factored Embedding and Distance. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, New York, NY, USA, 2010. Association for Computing Machinery. doi: 10.1145/1833349.1778840. Available at: `https://doi.org/10.1145/1833349.1778840`. ISBN 9781450302104.

[15] Lowe, T. *Exploring Scale Symmetry*. World Scientific, 2021.

[16] Maur, P. Delaunay triangulation in 3d. *Technical Report, Departmen. of Computer Science and Engineering*. 2002.

[17] McKinley, S. – Levine, M. Cubic spline interpolation. *College of the Redwoods*. 1998, 45, 1, s. 1049–1060.

[18] Miguel, A. L. – Nogueira, A. C. – Goncalves, N. Real-time 3D visualization of accurate specular reflections in curved mirrors a GPU implementation. In *2014 International Conference on Computer Graphics Theory and Applications (GRAPP)*, s. 1–8, 2014.

[19] Mitchell, D. – Hanrahan, P. Illumination from curved reflectors. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, s. 283–291, 1992.

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

[21] Mitra, N. J. et al. Symmetry in 3D Geometry: Extraction and Applications. *Computer Graphics Forum*. 2013, 32, 6, s. 1–23. doi: https://doi.org/10.1111/cgf.12010. Available at: `https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12010`.

[22] Mourycová, E. Hledání Symetrie Funkcí S Využitím Symetrie množiny bodů, Jun 2020. Available at: `http://hdl.handle.net/11025/41784`.

[23] Nagar, R. – Raman, S. Fast and Accurate Intrinsic Symmetry Detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[24] Nagar, R. – Raman, S. 3DSymm: Robust and Accurate 3D Reflection Symmetry Detection. *Pattern Recognition.* 2020, 107, s. 107483. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2020.107483. Available at: `https://www.sciencedirect.com/science/article/pii/S0031320320302867`.

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

[26] Ostertagová, E. Modelling using polynomial regression. *Procedia Engineering.* 2012, 48, s. 500–506.

[27] Pedrotti, L. S. Basic geometrical optics. *Fundamentals of photonics.* 2008, s. 73–116.

[28] Petitjean, M. A definition of symmetry. *Symmetry: Culture and Science.* 2007, 18, 2-3, s. 99–119.

[29] Rosen, J. *Symmetry in science: an introduction to the general theory.* Springer, 1995.

[30] Singer, S. – Nelder, J. Nelder-mead algorithm. *Scholarpedia.* 2009, 4, 7, s. 2928.

[31] Smolik, M. – Skala, V. Large scattered data interpolation with radial basis functions and space subdivision. *Integrated Computer-Aided Engineering.* 11 2017, 25, s. 1–14. doi: 10.3233/ICA-170556.

[32] Zatout, C. A brief introduction to neural networks: A regression problem, Dec 2022. Available at: `https://towardsdatascience.com/a-brief-introduction-to-neural-networks-a-regression-problem-c58c26e18008`.

List of Figures

List of Tables

Appendix 1: User Manual

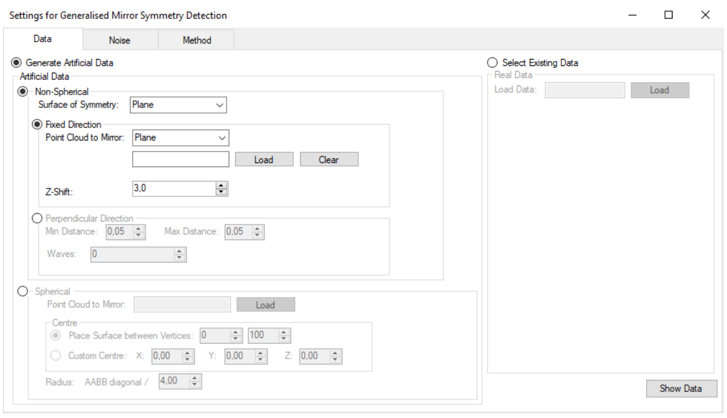
Compilation and Execution

Run the program by opening the `UIControl.exe` file located in the `Aplication_and_libraries/UIControl/bin/Debug` directory.

The compilation of the code requires two programs: MSBuild (version 16.11.2.50704 was used for testing) and Make (version 3.81 was used for testing). To compile the code, open the `Aplication_and_libraries` directory. Here, the Makefile is located. From this directory, execute `make` in the command line of your computer. To run the application execute `make run` or use the method described above.

Controls

After the program is started, a window with a form switched to the *Data* tab will open (see Fig. A1). At the same time, the system console will open, where information about the working of the program will be printed.



[✓ Generate Artificial Data]
Artificial Data
[✓ Non-Spherical]
Surface of Symmetry: Plane
[✓ Fixed Direction]
Point Cloud to Mirror: Plane
Z-Shift: 3,0
Perpendicular Direction
Min Distance: 0,05 Max Distance: 0,05
Waves: 0
Spherical
Point Cloud to Mirror:
Centre
[✓ Place Surface between Vertices:] 0 100
Custom Centre: X: 0,00 Y: 0,00 Z: 0,00
Radius: AABB diagonal / 4,00
Select Existing Data
Real Data
Load Data:

Figure A1: The window of the application switched to the *Data* tab

Data Tab

In the *Data* tab, the data to be generated or loaded can be configured. This tab is divided into sections, which are described below. The *Show Data* button located in the bottom right corner of this tab allows visualising the generated data (see the Visualisation section for more information).

Artificial Data

To generate artificial data, click on the *Generate Artificial Data* radio button. The associated controls will become available. To generate data using fixed or perpendicular direction of mapping, select the Non-Spherical radio button. Select the surface of symmetry from the list labelled *Surface of Symmetry.* Depending on whether fixed or perpendicular direction of mapping should be used, select the corresponding radio button, i.e., either *Fixed Direction* or *Perpendicular Direction.* The description of the associated controls follows:

- ***Fixed Direction***
  - ***Point Cloud to Mirror***

    Select a sampled surface from the list on the right or choose a file to be loaded using the *Load* button next to the text field. If both a surface and a file are selected, the file will be given priority and will be used. To remove the file selection, use the *Clear* button.
  - ***Z-Shift***

    Specifies, by what distance should the whole point set be moved in the direction of the z axis before it is mirrored over the surface of symmetry.
- ***Perpendicular Direction***
  - ***Min Distance, Max Distance***

    These controls specify the minimal and maximal perpendicular distance from the surface of symmetry. The maximal distance is taken into account only if the Waves setting is not equal to zero. Otherwise, only the minimal distance is considered.
  - ***Waves***

    This setting determines how many times the distance between points and the surface of symmetry will gradually change. For example, when the number is set to 0, all points will have constant distance (Min Distance) from the surface of symmetry. If the number is set to 1, the distance of points will gradually change, starting from the minimum distance and ending at the maximum distance. When the number is set to 2, the distance of points will change twice: starting at the minimum distance, then reaching the maximum distance, and finally returning to the minimum distance. This pattern continues for higher values of the parameter.

To generate data using a spherical surface, click the *Spherical* radio button. The description of the associated controls follows:

- ***Point Cloud to Mirror***

  Select a file containing data to be deformed using a spherical surface. Use the *Load* button to select the file. A file must be selected.

- ***Centre***

  Allows the setting of the centre of the sphere.

  - ***Place Surface between vertices***

    Select two vertices between which the surface of the sphere should pass. The centre of the sphere will be calculated in such a way that its surface passes through the midpoint between the selected vertices.

  - ***Custom Centre***

    Select the x,y and z coordinate of the sphere's centre.

- ***Radius***

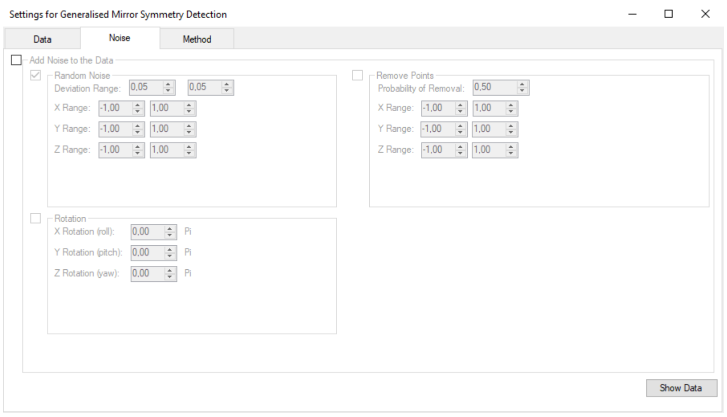
  Select the number by which the length of the AABB diagonal should be divided to acquire the radius of the sphere.

Real Data

To select existing data, click the *Select Existing Data* radio button and load a file using the *Load* button. The loaded point set will not be modified in any way. A file must be selected.

Noise Tab

To apply noise to the generated data, select the *Noise* tab (see Fig. A2) and check the *Add Noise to the Data* checkbox. The *Show Data* button located in the bottom right corner of this tab allows visualising the generated data (see the Visualisation section for more information). The description of the associated controls follows:



☐ Add Noise to the Data

✓ Random Noise
Deviation Range: 0,05 0,05
X Range: -1,00 1,00
Y Range: -1,00 1,00
Z Range: -1,00 1,00

☐ Remove Points
Probability of Removal: 0,50
X Range: -1,00 1,00
Y Range: -1,00 1,00
Z Range: -1,00 1,00

☐ Rotation
X Rotation (roll): 0,00 Pi
Y Rotation (pitch): 0,00 Pi
Z Rotation (yaw): 0,00 Pi

Figure A2: The window of the application switched to the *Noise* tab

- ***Random Noise***

  Select this option, if random noise should be added to the data.

  - ***Deviation Range***

    The two numbers specify the minimal (the first number) and maximal (the second number) deviation of a point from its correct position.
  - ***X Range***

    Specifies the interval on the x axis, which should be affected by this option.
  - ***Y Range***

    Specifies the interval on the y axis, which should be affected by this option.
  - ***Z Range***

    Specifies the interval on the z axis, which should be affected by this option.

- ***Remove Points***

  Select this option, if points should be removed from the set.

  - ***Probability of Removal***

    Specifies the probability of removing a point.
  - ***X Range***

    Specifies the interval on the x axis, which should be affected by this option.

  - ***Y Range***

    Specifies the interval on the y axis, which should be affected by this option.
  - ***Z Range***

    Specifies the interval on the z axis, which should be affected by this option.

- ***Rotation***

  Select this option, if the point set should be rotated.

  - ***X Rotation (roll)***

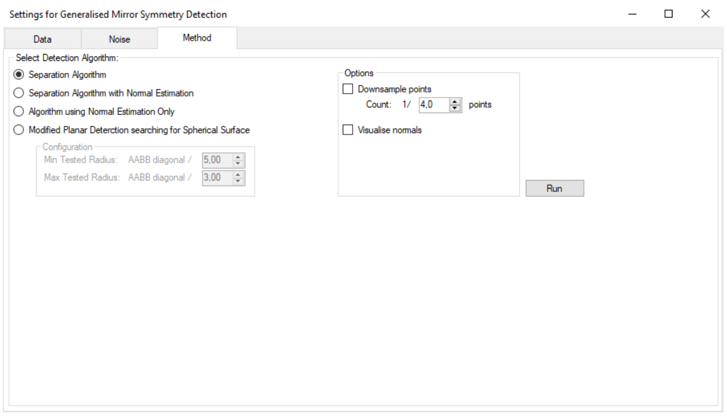
    Specifies the rotation about the x axis.
  - ***Y Rotation (yaw)***

    Specifies the rotation about the y axis.
  - ***Z Rotation (pitch)***

    Specifies the rotation about the z axis.

Method Tab

The *Method* tab (see Fig. A3) allows selecting the algorithm used for symmetry detection. The options are located on the left side of the window. If the algorithm modifying a method of planar symmetry detection is selected, a range of tested radii must be specified in the *Configuration* box.


Settings for Generalised Mirror Symmetry Detection

Data | Noise | Method

Select Detection Algorithm:
- [✓ Separation Algorithm]
- Separation Algorithm with Normal Estimation
- Algorithm using Normal Estimation Only
- Modified Planar Deterction searching for Spherical Surface

Configuration
- Min Tested Radius: AABB diagonal / 5,00
- Max Tested Radius: AABB diagonal / 3,00

Options
- Downsample points
  Count: 1/ 4,0 points
- Visualise normals

Run

Figure A3: The window of the application switched to the *Method* tab

- ***Configuration***
  - ***Min Tested Radius***

    The minimal tested radius when detecting a sphere of symmetry. Enter a number, by which the AABB diagonal of the generated point cloud should be divided to acquire the minimal tested radius.
  - ***Max Tested Radius***

    The maximal tested radius when detecting a sphere of symmetry. Enter a number, by which the AABB diagonal of the generated point cloud should be divided to acquire the maximal tested radius.

Additional configuration options are available in the *Options* box.

- ***Options***
  - ***Downsample points***

    Check this option, if the point cloud should be simplified. Enter the number by which to divide the number of points in the original point cloud to acquire the new approximate point count.
  - ***Visualise normals***

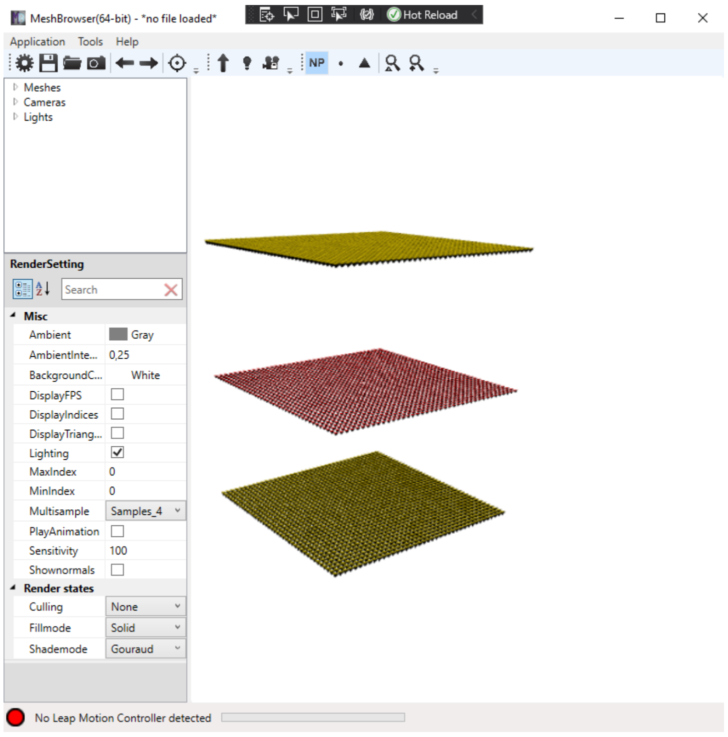
    Check this option if the approximated normals should be visualised. The normals will not be estimated and, therefore, will not be visualised, if the *Modified Planar Detection searching for Spherical Surface* algorithm is selected.

To launch the symmetry detection procedure, use the *Run* button. After the procedure is finished, the visualisation window will open automatically (see the Visualisation section for more information).

Visualisation

A new window will open when the generated point set is to be displayed. There will be multiple point sets ready for visualisation. Their specific type and order depends on the configuration. The yellow set is the input set, the green set represents the correct surface (the one used for data generation) and the red set is the detected set (output from the symmetry detection algorithm). If the visualisation of normals was selected, the approximated normals will be coloured in brown. See Fig. A4 for an example of data visualisation.

Please note that the program may sometimes crash when opening the visualisation tool, especially when attempting to open it repeatedly. This issue is caused by the SlimDX library, which is used for the visualisation.



RenderSetting

Misc

Ambient	Gray
AmbientInte...	0,25
BackgroundC...	White
DisplayFPS	☐
DisplayIndices	☐
DisplayTriang...	☐
Lighting	☑
MaxIndex	0
MinIndex	0
Multisample	Samples_4
PlayAnimation	☐
Sensitivity	100
Shownormals	☐

Render states

Culling	None
Fillmode	Solid
Shademode	Gouraud

No Leap Motion Controller detected

Figure A4: The visualisation tool window

Controls of the Visualisation Tool

You can rotate the object by holding down the left mouse button and dragging it. If you simultaneously hold down the "L" key, the light will be adjusted. To pan the object, hold down the right mouse button and drag it. To list through the point sets, use the "," and "." keys.

Output

The symmetry detection procedure produces two outputs. The first one is the points lying on the surface of symmetry, or the surface itself, while the second output is the computed errors. These outputs are saved to a file named `outXtime` in the working directory, where `X` represents the used algorithm and `time` represents the time the file was saved in the format of day-month--hour-minute-second.

The format of the content of this file is the following: The beginning of the file contains the calculated errors. These lines begin with the `#` character. The rest of the file contains the calculated points lying on the surface of symmetry. Each point is saved in the format of `x,y,z`, where x, y, and z denote the respective coordinates. The coordinates are separated by a comma. If the result of the symmetry detection is a sphere, the centre of the sphere is saved in the same format. The radius is saved on the following line.

Errors are only calculated, when the correct result is known, which is the case when artificial data is generated using the method that corresponds to the selected algorithm. If the input set is rotated, the correct result is unknown.