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

Master Thesis

Finding the Symmetry of Geodata

Pilsen 2022 Bc. Ondřej Anděl

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Ondřej ANDĚL**
Osobní číslo:	**A20N0074P**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Téma práce:	**Hledání symetrie v geodatech**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s dostupnou literaturou k zadanému problému a s existujícím programovým vybavením pro hledání rovin symetrie vyvinutým na KIV.
2. V návaznosti na výsledky předchozího výzkumu navrhněte vhodné metody pro nalezení globálních a lokálních symetrií v zadaných datech.
3. Navržené metody implementujte jako webovskou aplikaci tak, aby bylo možné snadno přidávat další metody a jiné formáty vstupních dat, exportovat nalezené výsledky do souboru a nalezené symetrie vizualizovat. Pro vizualizaci můžete využít již implementovaných knihoven, dostupných pro nekomerční aplikace.
4. Implementované metody otestujte na reálných datech, získané výsledky popište v textu práce a 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:	**10. září 2021**
Termín odevzdání diplomové práce:	**19. května 2022**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 11. října 2021

Declaration

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

Pilsen, 19th May 2022

Bc. Ondřej Anděl

Abstract

This thesis is focused on finding the reflective symmetry in geodata using the existing software created by Ing. Lukáš Hruda. The mentioned software was optimized for finding symmetry in objects represented by its surface points, therefore, it does not perform well with geodetic data represented by point clouds where the symmetry is much less expressive. The goal of this work is to propose preprocessing methods to improve the quality of the found symmetry planes for this kind of data.

Abstrakt

Tato práce je zaměřena na nalezení reflexní symetrie v geodatech pomocí stávajícího softwaru vytvořeného Ing. Lukášem Hrudou. Zmíněný software byl optimalizován pro hledání symetrie v objektech reprezentovaných povrchovými body, proto si nevede dobře s geografickými daty reprezentovanými mračny bodů, kde je symetrie mnohem méně zjevná. Cílem této práce je pro tento typ dat navrhnout metody předzpracování pro zlepšení kvality nalezených rovin symetrie.

Acknowledgments

I would like to thank Prof. Dr. Ing. Ivana Kolingerova, Ph.D. for her valuable advice, factual comments and helpfulness in consulting and elaboration of this master's thesis. Furthermore, I would like to thank Ing. Lukáš Hruda for advice on working with the symmetry detection software and Mgr. Kateřina Uhrová for advice on weights of points. I would also like to thank Bc. Eliška Mourycová for the cooperation on integration of the segmentation algorithms. At last, I would like to thank Ing. Ivo Malý, Ph.D. and prof. Ing. Pavel Slavík, CSc. for their advice in the topic of user tests and symmetry evaluation.

Contents

1 Introduction **8**

2 Symmetry and Symmetry Detection **9**
2.1 Mathematical Algorithms 10
2.1.1 Combès et al. 10
2.1.2 Lipman et al. 11
2.1.3 Mitra et al. 12
2.1.4 Cicconet et al. 13
2.2 Neural Network Algorithms 15
2.2.1 PointNet 16
2.2.2 PRS-Net 17
2.2.3 SymmetryNet 17
2.3 Evaluation Metrics 18

3 LAS Format **20**
3.1 Format Specification 20
3.2 LAS Readers and Converters 23

4 Proposed Solution **25**
4.1 Preprocessing Module 25
4.2 Hruda et al. Detection Algorithm 27
4.2.1 Symmetry Measure 27
4.2.2 Candidate Election and Pruning 28
4.2.3 Weight Significance 29
4.3 Evaluation Web Application 30
4.4 Experiments Setting 31

5 Proposed Global Preprocessing **34**
5.1 Laplacian Operator 35
5.1.1 Height-based Laplacian 38
5.1.2 Color-based Laplacian 39
5.1.3 Intensity-based Laplacian 40
5.2 Classification Use 41
5.2.1 Majority Classification Use 42
5.2.2 Alternative Classification Use 43
5.3 Data Flattening 43

5.4 Preprocessing Evaluation 44

6 Proposed Local Preprocessing **47**
6.1 Manually Set Weights 47
6.1.1 Weights 1:0 48
6.1.2 Weights 5:1 50
6.2 K-Means Segmentation 52
6.2.1 Weights 1:0 53
6.2.2 Weights 5:1 55
6.3 Bounding Box Segmentation 57
6.3.1 Weights 1:0 58
6.3.2 Weights 5:1 58
6.4 Preprocessing Evaluation 61

7 User Evaluation of Symmetry **63**

8 Conclusion **66**

Bibliography **67**

1 Introduction

Many objects of our modern world show various grades and types of symmetry and automatic detection of symmetry is an important and difficult topic. For instance, symmetry is useful for scientists who want to understand and classify objects or for engineers who want to compress the data.

Due to the complexity of the problem, many algorithms for symmetry detection exist. The algorithms are optimized for various symmetry and data input types. This thesis follows up on the work of Ing. Lukáš Hruda in which he proposed and implemented an algorithm for detecting reflective symmetry in geometric objects represented by the set of points. The mentioned solution was proposed namely for geometric models depicted by surface points.

The goal of this thesis is to apply the implemented algorithm on a different type of data than it was originally proposed for. As the original and newly studied data types differ greatly, it was assumed that the algorithm would require additional adjustment and data preprocessing to improve the detected symmetry planes.

This thesis is part of international CSF project (the project 21-08009K, Generalized Symmetries and Equivalences of Geometric Data) being solved by the Faculty of Applied Science, University of West Bohemia in Pilsen and by the Faculty of Electrical Engineering and Computer Science of University of Maribor, the project partner, in Slovenia. Therefore, the implemented solution will be provided to the Maribor team members to perform additional experiments with other types of geodetic data. Furthermore, the results obtained by the implemented solutions are to be used for user tests at the Faculty of Electrical Engineering of the Czech Technical University in Prague. As such it was established that a web application aimed at such testing will be additionally implemented by the author of this thesis.

In Chapter 2, the problem of symmetry and symmetry plane detection will be explained. In Chapter 3, one of the most common geodetic formats (format LAS) will be described along with its interpreters. The Chapter 4 will focus on introducing the proposed solution. In Chapter 5 and Chapter 6, preprocessing methods (aimed on global and local symmetries, respectively) will be proposed and the results shown. Chapter 7 will inform about an experiment with user evaluation of reflective symmetries. Chapter 8 concludes the text.

2 Symmetry and Symmetry Detection

Before defining symmetry it is important to briefly introduce the data type used. All referenced data will be geodetic data represented by point clouds. The point clouds are defined as a set of data points in a space. The specifics of the used data types may vary depending on the actual data format. The most common geodetic format based on point cloud representation is LAS format which is described in more detail in Chapter 3.

In math, symmetry is described as an operation that leaves an object or a set of objects invariant. In other words, the object remains the same after applying a predetermined transformation. There are several divisions of symmetry based on this transformation, the range or the quality of the invariance.

Symmetry based on transformation can be divided into four major groups: reflective, rotational, translational and glide symmetries [5]. This thesis will focus only on the reflective symmetry and finding its planes. If we consider the range of symmetry, we talk about global or local symmetries. Global symmetry describes the case where the whole data set remains invariant. In contrast to this, local symmetry takes into account only a segment of the original data.

Symmetry can be divided also into perfect and approximate. Perfect symmetry describes objects capable of staying perfectly invariant, i.e., the original object will remain unchanged while all its points will be transformed into other points of the same object. On the other hand, approximate symmetry allows some points not to be represented in the transformed data, which allows symmetry to be detected even in partially damaged or noisy objects. Due to the specific type of the input data studied in this thesis, we are going to consider mainly approximate symmetries.

As stated in the introduction, this thesis is focused on detecting both global and local reflective symmetries in the geodata. The specific symmetry and data types put a additional restriction on detection algorithms. While the implemented solution will depend only on the algorithm described in Section 4.2, to familiarize reader with the topic, several other suitable algorithms will be introduced within Chapter 2. For greater comprehensibility, the algorithms will be divided into two groups. In Section 2.1 multiple algorithms based on mathematical rules and hypothesis will be introduced.

Algorithms based on machine learning will be described in Section 2.2. Section 2.3 is devoted to evaluation metrics.

2.1 Mathematical Algorithms

To familiarize the reader with the mathematically based algorithms and possible detection approaches, several algorithms with their base concepts will be named before delving deeper into the more significant ones. Due to the focus of this thesis, only the algorithms capable of performing in point clouds will be mentioned.

The algorithm by Kakarala et al. [12] uses the spherical harmonic domain in reflection symmetry detection. Korman et al. [13] propose a detection method using distortion measurement which quantifies the amount of volume mismatch between the original and the reflected object. The method proposed by Thrun and Wegbreit [36] performs hierarchical generate-and-test procedures based on taxonomies from different types of symmetries.

Schiebener et al. [29] base their algorithm on information about object surroundings and perform triangulation, which is used in uniform sampling to determine the symmetry plane candidates. Sun et al. [35] use mapping in a set of similar objects to determine symmetry plane candidates in a single object within the set. Simari et al. [33] propose symmetry detection method based on the distance of reflected points from the object surface and the weighted covariance matrix.

The approach developed by Podolak et al. [24] utilizes planar reflective symmetry transformation for volumetric functions and Monte Carlo sampling for surface transformation. Speciale et al. [22] suggest two detection algorithms, one using the RANSAC approach and the other using the Hough transform. The algorithm implemented by Cailliere et al. [2] enhances the solution of Mitra et al. [21] (described further in Section 2.1.3) by utilizing the Hough transform.

2.1.1 Combès et al.

This approach published in [4] is specifically focused on finding reflectional symmetry in point clouds. It is able to find both local and global symmetries in incomplete and damaged data. Due to its iterative closest point approach it is possible that the program will find only local symmetry which is not desirable.

The algorithm itself is rather simple. It starts with some initial plane within the data and progressively works to improve the count and the pre-

cision of points perfectly reflected through this plane. To achieve that, the method reflects the point set according to the initial plane and tries to minimize its distance from the closest point within the file set. This operation can be mathematically written as Eq. 2.1, where x_i represents the original point, y_i represents the closest point, S_p is a reflection transformation, $f(P)$ represents a quality evaluation of reflection over plane P, n represents the number of points of the sample. There is a new plane created during the function minimization. If it differs from the plane used in the previous iteration, the algorithm repeats itself over this plane. This algorithm is also proposed in a slightly adjusted version to minimize the negative influence of outliers which could lead to getting only local symmetry instead of the global one.[4]

$$f(P) = \sum_{i=1}^{n} \|y_i - S_P(x_i)\|^2 \tag{2.1}$$

The work presents the results of detection method on an incomplete scan of face, Stanford bunny and chair with a missing leg (see Fig. 2.1). The chair and the face examples show the quality of the algorithm on incomplete data while the bunny shows the algorithm's ability to work with approximate symmetries. In the presented examples the algorithm manages to find the symmetry planes. Nonetheless, it is important to note that all of the input data contain about 80 000 points which is rather low.

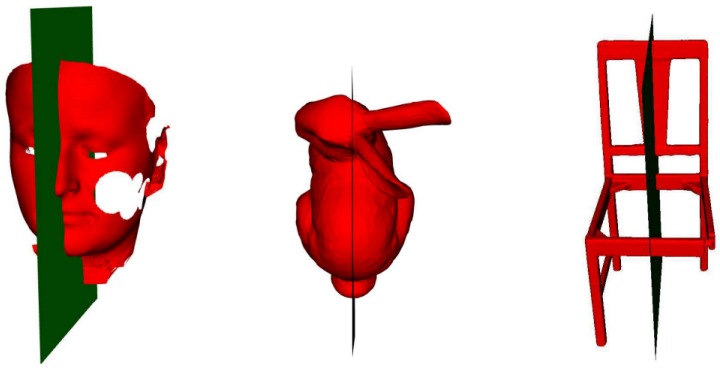


Figure 2.1: Results of detection algorithm implemented by Combès et al.[4]

2.1.2 Lipman et al.

The detection algorithm introduced by Lipman[19] manages to find global and local symmetries over various data, it supports even the approximate symmetry detection. Since it requires only the point coordinates, the method would be viable for use in cloud point formats.

The method is based on the symmetry correspondence matrix C, the elements of which quantify the points integration into the same orbit (i.e., the range of Symmetry Factored Distance). The points within the same orbit are usually linked by a short (ideally zero-length) edge but for point clouds the distance between the set of points has to be used instead.

The matrix C is computed from the dissimilarity matrix S using Eq. 2.2. $\widetilde{C}_{ij}$ is one matrix element of the symmetry matrix and S_{ij} is a cell of dissimilarity matrix which measures how well can the point set X be preserved by a rigid transformation that transforms the point x_i to x_j. σ is the localization parameter in the range 0.1% - 1%. Variable *diam* is the point set's diameter. After computing the symmetry correspondence matrix mentioned above, a spectral analysis can be performed, which will result in finding the symmetry plane.[19]

$$\widetilde{C}_{ij} = e^{-\left(\frac{S_{ij}}{\sigma diam}\right)^2} \tag{2.2}$$

While the method's main goal is not detecting symmetry planes specifically (the work focuses more on the rotational symmetry) several examples of the symmetry plane detection were presented in the work anyway. The algorithm manages to find the symmetry planes in the data which contain only approximate symmetry planes (see Fig. 2.2). None of the data have missing parts and similarly to the previously mentioned method the algorithm results are presented only on smaller examples. The algorithm while capable of finding symmetries is one of the slower methods.

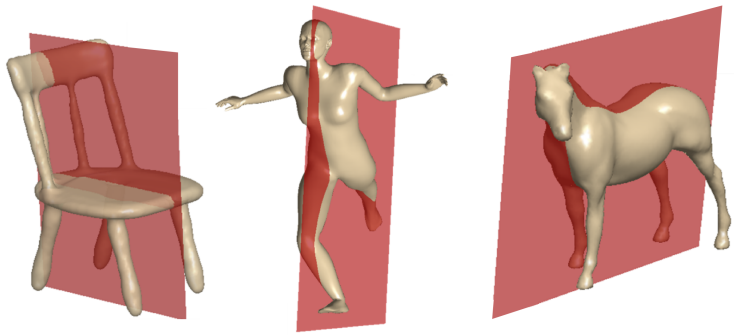


Figure 2.2: Results of detection algorithm implemented by Lipman et al.[19]

2.1.3 Mitra et al.

This method [21] is not specifically focused on cloud point models, but rather focuses on 3D mesh models. Since this method assumes the real world

objects being scanned to get sample points and describes the entire process of getting and approximating these samples, the initial phases can be skipped while working with cloud point formats. This method supports perfect and even approximate symmetries, it can perform well even over noised data. The method is suitable for finding not only reflective symmetries, but also rotational ones.

This method works with the Euclidean transformation group (group of transformations that preserve a distance between two points) and focuses on finding object segments which are invariant to the transformations defined within the system (translation, rotation, reflection and uniform scaling). The original paper states that only positional information is insufficient, therefore, the curvature tensors and integrated principal curvatures are obtained. These quantities are used to establish a 7-dimensional transformation space.

To establish the potential symmetries, the data are reduced and divided into pairs of points. The created pairs represent the symmetry relation at local sample spacing. To obtain object parts which are symmetrical, the above mentioned pairs have to be clustered with the other pairs having the same or similar transformation. The paper proposes to create a weighted sum and adjust it by a mean shift. The mean shift is an iterative clustering algorithm based on shifting points towards the highest density of the data in a region. The mean shift in Mitra detection algorithm is required as most of the real objects do not show perfect symmetries but only approximate ones.

The created clusters represent the object parts which exhibit the symmetrical behavior, however, as the pairs no longer contain positional information, it is possible that the cluster will consist of random points. The authors propose another verification, in which they pick random points from each cluster and add only neighbors that remain within the error threshold after applying the symmetry transformation.[21]

This method can be used to detect not only reflective symmetry but also other types. The method needs for its function to have densely sampled data. Similarly to previous solutions, the method was tested only on smaller data hence it is unknown how well it would perform on larger geodetic data inputs. The results in the paper are presented on models of three buildings, dragon statue and a horse. Due to the lower quality and readability of mentioned images only the sample of castle is shown in Fig. 2.3.

2.1.4 Cicconet et al.

The algorithm proposed by Cicconet et al. [3] is focused on finding symmetry through the registration and is focused mainly on medical data (namely

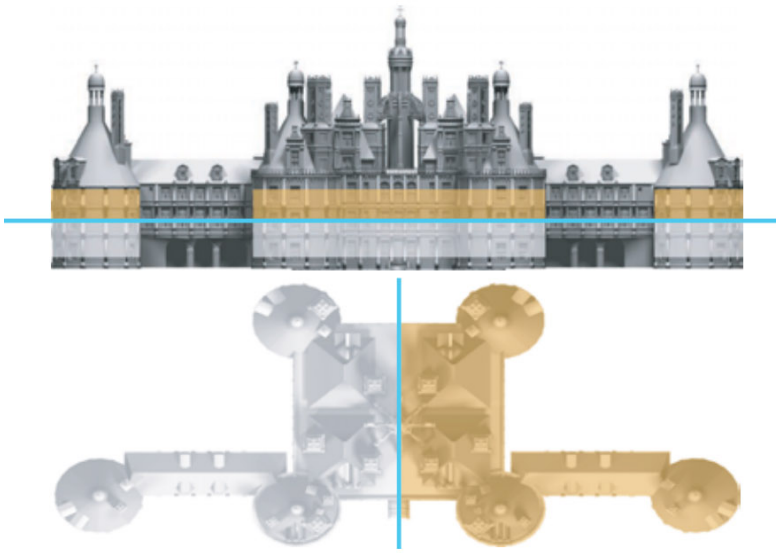


Figure 2.3: Results of detection algorithm implemented by Mitra et al.[21]

from the field of neuroscience). As such, the proposed approach is capable of detecting both the local and the global symmetries in 2D and 3D data formats. The accuracy of the algorithm is greatly influenced by what initial symmetry plane is used.

To start this method, first we must choose the initial reflection plane and represent it as a point and a perpendicular vector. As the chosen point does not require to be in the center of the points, this method can detect even planes of symmetry which do not contain the origin of the data. After establishing the initial plane, all the points are to be reflected in order to create a new set of points. The created set ought to be registered onto the original set via the rigid transformation.

Through the process of registration, the new rotation matrix and the transformation vector are obtained. They can be used in a further analysis. Finally, the symmetry plane can be computed by getting the eigenvector v from the rotation matrix and computing the point $\overline{p}$ in Eq. 2.3. R_0 is a rotation matrix, t is a translation vector, v is a perpendicular vector and d is a signed distance between the plane and the origin. The obtained vector v and the point $\overline{p}$ represent a new definition of the symmetry plane [3].

$$\overline{p} = \frac{1}{2}(R_0(2dv) + t) \tag{2.3}$$

The method was tested mainly on 2D images in which it consistently detects a satisfactory symmetry line. While the method was tested even on 3D data there is only one example in the paper. Due to the inadequate

testing set it is unsure whether the method would be suitable for handling larger point clouds. Results of the method are presented in Fig. 2.4. Authors admit that the method does not output the intersection of the computed symmetry plane with the symmetric object.

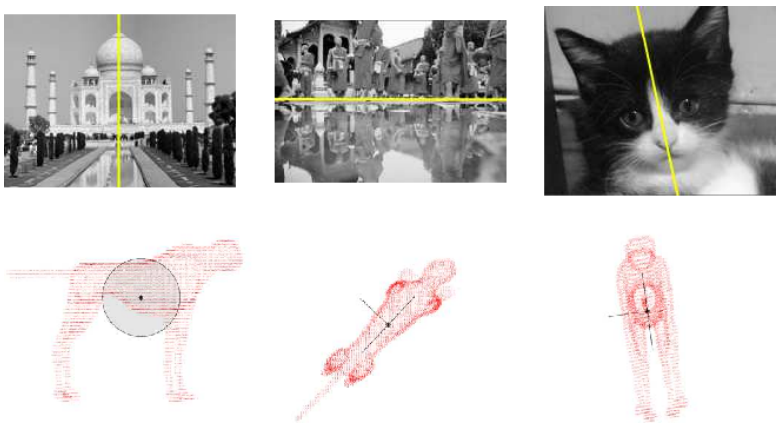


Figure 2.4: Results of detection algorithm implemented by Cicconet et al.[3]

2.2 Neural Network Algorithms

Neutral network algorithms work by detecting and learning a certain pattern within the learning data and later replicating this pattern within regular data inputs. Similarly, neural network programs can be used to learn and later detect symmetry planes within input data. This functionality is provided by mapping the numerical input into a probability function. These data are further modified by multiple transformation layers until they meet the required accuracy and are accepted as output data. Within the program the layers differ according to their purpose, however, the act of training adjusts weights to connections between each layer.

The data considered within the scope of this thesis are point clouds, which are generally unstructured and unordered. This format restriction requires to keep the permutation invariance (while predicting values for the whole set) and the equivariance (while computing values for one input point). Permutation invariance is defined as a function, which does not change if we permute the input values. The permutation equivariance is then defined as a function, the result value of which changes if its inputs are permuted.

Algorithm training is performed on data with a predetermined symmetry plane. The inner values within these data are set to 1 for points near the symmetry plane, the remaining points have their weights set to 0. The

inbalance of the weights mentioned above is solved by the weighted cross-entropy loss function. Most neural network algorithms produce several false-positive results, therefore, it is necessary to use a random sample consensus (RANSAC) to produce candidate planes. [34]

As stated above, neural network algorithms do not require any specific data formats, so they can be used even for the point clouds. Furthermore, the neural network can be used to separate feature and background partitions within the data. These features can then be used for data preprocessing in this thesis. However, due to the complexity of this solution and its dependence on appropriate test samples, after a discussion with the supervisor of this thesis, it was established not to use these algorithms. Therefore, the algorithms described within this section aim only to educate the reader about additional approaches to symmetry detection.

2.2.1 PointNet

PointNet is a neural network algorithm designed for manipulating input points in 3D. The individual input points are convoluted with a shared MLP (i.e., a multilayer perceptron) function which extracts a predetermined count of features per point. Equivariance in the data is achieved by sharing the MLP weights, while invariance is achieved through max pooling of maximum values of point features. This method is used to obtain significant informative points from an input whose attributes are used to perform classification or segmentation. The algorithm then applies the learned rigid transformation to the input point cloud [25].

Unfortunately, the mentioned method fails to contemplate the scaling factors within the point cloud format, therefore, it is not optional for LAS formats (which are important for the GIS data) in its base form. PointNet ideas are further extended in the method PointNet++. This method is defined by a hierarchical division of the input into overlapping regions (called farthest point sampling). After that, the algorithm extracts local features by the algorithms described above (i.e., PointNet approach). The obtained features are aggregated into global ones through multiple network layers [26].

In order to obtain reflectional symmetry over point cloud formats via this method, the data need to be processed by PointNet++ to determine which points belong to the symmetry plane. The results of this method are used in the RANSAC. This action is necessary to get rid of false-positive candidate planes. After that, the least squares method is used to obtain the initial plane equation. The generated plane is then further optimized by the iterative closest point (ICP). ICP's approach is to reflect original data

over this plane and minimize the distance between the original and reflected clouds. The described solution produces only global symmetry of complete objects, the symmetry plane is assumed to intersect the data origin.[11]

2.2.2 PRS-Net

PRS-Net is a neural network algorithm capable of detecting both reflection and rotation symmetry. It requires the input in a form of voxels. A voxel (also known as a volumetric pixel) is defined as a volume element representing some numerical quantity of a point in a three-dimensional space.

Neural network convolutions and max pooling layers of this algorithm produce the feature vector of the size 64 elements. Three separate branches of layers use this vector to predict up to three symmetry planes and three axes of rotation symmetry. After that, the unsupervised loss function is used as a symmetric metric to reflect or rotate (based on the required symmetry type) the geometry using generated symmetry plane or axis. Predictions which are poor local minima are removed.

The mentioned loss function consists of symmetry distance loss and regularization. At distant loss, the input is uniformly sampled and mapped via reflection or rotation. After that, the smallest distance of mapped points is computed. The sum of computed values of all points, planes and values is called the loss value. The regulation within this method is used to prevent duplicate results and uses the mentioned loss value to detect orthogonal samples [7].

2.2.3 SymmetryNet

SymmetryNet is able to detect reflection and rotation symmetry in 3D data with missing parts. The method is based on predicting multiple symmetries of a certain type via neural network. Each prediction is represented as a pair of points from the symmetry plane or the axis domain and its normal or direction vector. Network output predictions are clustered with corresponding ground truths through optimal assignment.

This approach begins with the extraction of point-wise features from RGB-D (i.e., RGB image enhanced with depth information) using a convolution neural network and PointNet. Global features are acquired through spatially weighted pooling. The three-layer MLP then uses both of the mentioned feature types for prediction. DBSCAN (i.e., density-based spatial clustering) clusters predictions and centroids are returned as final predictions, which are weighted by a loss function. The results are then voxelized

to remove free and invisible voxels from the viewport, and the surface is transformed using the predicted symmetry. If the transformed surface points and free regions often overlap, they are then removed from the results as a significant error was detected [32].

2.3 Evaluation Metrics

As this thesis is focused on finding approximate reflective symmetries, it is necessary to specify evaluation metrics. The evaluation metrics aim to quantify the quality of found symmetries and, therefore, enabling comparisons of different detection algorithms or preprocessing approaches. This thesis will not use any specific evaluation metric and will quantify the quality of results by visual evaluation. The rest of this section will focus on introducing possible evaluation metrics.

Rassat [27] proposed a symmetry measure based on Hausdorff distance. Hausdorff distance measures how far two subsets of a metric space are from each other. Provided that the primary distance $d(x, y)$ between points x and y is defined, the Hausdorff distance between two subsets X and Y of this metric space is defined by Eq. 2.4. Buda and Mislow [1] expanded on the metric by applying it to mirror images and normalizing it to the diameter of the set (i.e., the upper distance between two points of the set).

$$H(X,Y) = \max\left\{\sup_{x\in X}(\inf_{y\in Y} d(x,y)), \sup_{y\in Y}(\inf_{x\in X} d(x,y))\right\} \tag{2.4}$$

Kuzmin and Stelmakh [14] introduced a chirality measure for 3D set of weighted points. The proposed method mirrors an image by one of the following symmetry operations: S_1, S_2, S_3 or S_4. In every reflection, each of n points from the set is associated with a point from mirror image. The sums of the lengths of the n pairs are denoted DA_x, DA_y and DA_z for the reflections through the planes yz, zx and xy, respectively. The degree of asymmetry for each symmetry operation is DA^{S_k}, with $k = 1, 2, 4, 6$ and is computed by Eq. 2.5.

$$DA^{S_k} = (DA_x^{S_k} * DA_y^{S_k} * DA_z^{S_k})^{\frac{1}{3}} \tag{2.5}$$

Harris, Kamien and Lubensky proposed a metric [8] describing a density in 3D space with spherical coordinates. Their solution uses tensor moments obtained by integration over the radial coordinate. Rasssat, Làszló and Fowler used the mean square topological strength to propose a chirality measure [28]. Marola [20] used an indirect symmetry measure to detect a

symmetry axis in a digitized 2D image. The measure is represented by Eq. 2.6, where w is the intensity function, (x, y) is point from the image and $(\bar{x}, \bar{y})$ is a point paired with (x, y) [23].

$$\beta = \frac{\int\int w(x,y)w(\bar{x},\bar{y})dxdy}{\int\int w^2(x,y)dxdy} \tag{2.6}$$

The presented metrics were chosen for the description in this thesis, as they support point clouds (the typical representation of geodetic data) and are also applicable for reflective symmetries. More detailed description of possible metrics is introduced in the review [23] written by Petitjean.

3 LAS Format

The LAS format is the most common format of geodata obtained by the technology LiDAR. The acronym LiDAR stands for Light Detection and Ranging and it is a terrain researching technology. The main concept of the technology is to scan the Earth surface using a pulsed laser. This method consists of three main parts: a laser emitter, a scanner and a GPS receiver.

LiDAR data are obtained by emitting a concentrated pulsed laser, and waiting for the reflected pulse. After the reflected light is detected, the laser range is computed and combined with the information obtained from GPS. Given that we know how fast the light travels and the strength of the original pulse, we can calculate the approximate distance from the object. By studying the strength of the reflected pulse, it is also possible to analyze the surface and the color of the object. This technology is used to scan large parts of terrain taking millions of entries, creating a structure known as a point cloud [18].

There are two types of LiDAR: terrestrial and airborne. While the terrestrial is performed by moving vehicles, the airborne one uses special helicopters or drones. Most LiDAR data sets were taken by airborne LiDAR [30].

Due to huge amount of data obtained by the LiDAR method, the data are usually saved in binary files such as LAS. Over the years, the LAS format has gone through several backward-compatible changes, the format described would apply to version 1.4, which was published in 2011 and has remained the last version till today.

3.1 Format Specification

The format LAS consists of four parts: a public header block, variable length records, point data records and extended variable length records. To ensure portability, if any item in any part is not defined, it needs to be set to the proper null representation (in most cases it is zero). While the length of each argument is important in the format itself, for this thesis it will be dismissed and only the major arguments and the purpose of each block will be described.

Each LAS header starts with the file signature consisting of four chars giving the string "LASF", this initial signature is to determine the file type and its validity. Another important piece of information in the header is the major and minor version of LAS, because some of the arguments and their

length vary between versions, it should be read and considered appropriately when handling this format. The LAS specification records both the day and the year of creation as well.

In addition to the version information, the files contain the size of the header itself, the offset to the point data, the point data format (depending on the version, there are up to eleven possible variants of point formats), the length and the count. In order to store a larger number of points, the record data are usually stored in a reduced format. To get the real values, it is required to apply the correct scale and offset factors, which are also saved in the header. Finally, the header contains both minimal and maximal coordinate values that can be further used for data processing.

Classification Value (bits 0:4)	**Meaning**
0	Created, never classified
1	Unclassified
2	Ground
3	Low Vegetation
4	Medium Vegetation
5	High Vegetation
6	Building
7	Low Point (noise)
8	Model Key-point (mass point)
9	Water
10	*Reserved for ASPRS Definition*
11	*Reserved for ASPRS Definition*
12	Overlap Points
13-31	*Reserved for ASPRS Definition*

Table 3.1: Point record classifications (formats 0-5)[17]

The Variable Length Records contain variable data types, including projection information, metadata, waveform packet information, and user application data. They are not important for the scope of this thesis, therefore, their format will not be described.

The most important part of the LAS format are the point data records. As stated above, records can be stored in eleven different formats, the used format is specified in the header block. All records within one file must have the same format. All formats share a similar core, which is represented by the first format with index zero. Other formats differ by having additional

arguments, such as GPS time, RGB color channels, near infrared channel or wave packets data.

Classification Value	**Meaning**
0	Created, never classified
1	Unclassified
2	Ground
3	Low Vegetation
4	Medium Vegetation
5	High Vegetation
6	Building
7	Low Point (noise)
8	*Reserved*
9	Water
10	Rail
11	Road Surface
12	*Reserved*
13	Wire - Guard (Shield)
14	Wire - Conductor (Phase)
15	Transmission Tower
16	Wire-structure Connector (e.g. Insulator)
17	Bridge Dock
18	High Noise
19-63	*Reserved*
64-255	*User Definable*

Table 3.2: Point record classifications (formats 6-10)[17]

All point formats contain information about the coordinates (x, y and z) and the intensity of the returned pulse. Additionally, the information about the scan direction, classification and scan angle can be found here. From all the information in the core format, the most important one (excluding coordinates and intensity itself) is the classification information. All data can contain their own clustering flags (see Table 3.1 and Table 3.2). It should be noted that the point format zero to ten use different length for this flag, therefore, it does not allow as specific classification as later formats.

The Extended Variable Length Records serve the same purpose as the VLR described above, their main advantage is that because they are at the end of the file, they can be appended and do not slow down reading point records. For further information rely on the official format specification [17].

3.2 LAS Readers and Converters

Due to the complexity of LAS format, in order to work with it, it is important to have the proper tools for its decoding. There are several programs both paid and free for working with this format, some allow only the visualisation of the information contained within it, other are used to directly extract it and convert it into human readable files. This section will introduce some of the most important. Since the main focus of this thesis is data preprocessing, the visualiser capable of displaying the format will be mentioned only marginally.

One of the best known applications for working with LAS is ArcGIS, which allows not only conversion and export of LAS data, but also visualisations and editing of this format. ArcGIS is also usable from code written in python. Unfortunately, this tool is not free and too complex for our needs, therefore, it is not directly suitable for this thesis. Another program capable of transforming point cloud formats into human readable files is LASUtility. Using this program for transforming data is viable, but for the scope of this thesis, an automatic solution (i.e., public code or library) is more preferable.

From the free public libraries, the most notable are libLAS and LAStools, both programs are capable to export data from the format LAS, and perform additional adjustment to this data. Due to the lower maintenance of these projects, the library PDAL is nowadays recommended to be used in new projects.

The figures in Appendix 1 are shown in contrast to the data previews obtained from the online application plas.io. This application is able to visualize LAS data in all available LAS point formats. It is able to show color information, classifications and intensity. Unfortunately, it does not focus on symmetry in this format, hence it does not support plane visualisation. It cannot be used for all images presented in this thesis.

Remaining images of geodata within this thesis are taken from the visualizing software implemented by doc. Ing. Libor Váša, Ph.D. and Ing. Lukáš Hruda. The software was provided as an additional module of the symmetry detection algorithm described in Section 4.2. This visualisation tool was not implemented for point cloud formats specifically, and since it is not publicly available, it will not be described further.

Due to the complicated use of the mentioned programs and libraries, a simple LAS reader was implemented in the preprocessing module using the programming language C#. The implemented reader is able to obtain all the information contained within the LAS format, but by default it reads only coordinates, intensity, color and classification information. Reading

additional information requires minor changes to the application code, which may slow down the performance of this software. If the input file contains more than 50,000,000 points (the number was chosen to provide a suitable size for our experiments), only each seventh point will be read (i.e., the point cloud is simplified).

4 Proposed Solution

The proposed solution consist of preprocessing module, Hruda's symmetry detection algorithm, visualization of the results and web application for results evaluation. The mentioned parts will be described in more detail in order of their execution. The solution is also described in user and programmer documentations. For greater readability, mentioned documents were written separately for the symmetry detection solution and the symmetry evaluation web application.

4.1 Preprocessing Module

Due to the complexity of the original detection algorithm (see Section 4.2), the preprocessing solution was implemented as a separate module that modifies data from input files and transforms them into files with supported extensions. The module is specifically focused on geodata, which need additional modification before providing satisfactory results.

The preprocessing approaches are dividable into two groups: preprocessing aimed to improve global symmetry detection (described in Chapter 5) and preprocessing aimed to enable local symmetry detection (see Chapter 6). The module workflow and classes responsible for individual functionalities are depicted in Fig. 4.1.

The module accepts input files with extensions LAS and DAT. Extension DAT consists purely from point coordinates and as such is read directly in preprocessing methods. Extension LAS requires data interpreter. As stated in Section 3.2, due to the complexity of library solutions and their support, an original LAS interpreter was implemented.

After the data are successfully loaded (and the preprocessing method is determined), the application launches the required preprocessing method. Within this thesis six methods for global preprocessing and three methods for local preprocessing were implemented. The supported global preprocessing include three methods based on the Laplacian operator (height-based, color-based and intensity-based) and two methods for LAS classification uses (majority-based and based on the object origin). In order to determine significance of the outline points, outline separations (an experiment possible only on triangulated data) was implemented. Aside from the mentioned approaches, the data can be also interpreted without prepro-

cessing which leads to the modification of the input files into files supported by the symmetry detection application. All mentioned global methods and experiments were implemented with an option to flatten the data (i.e., set all z coordinates to zero).

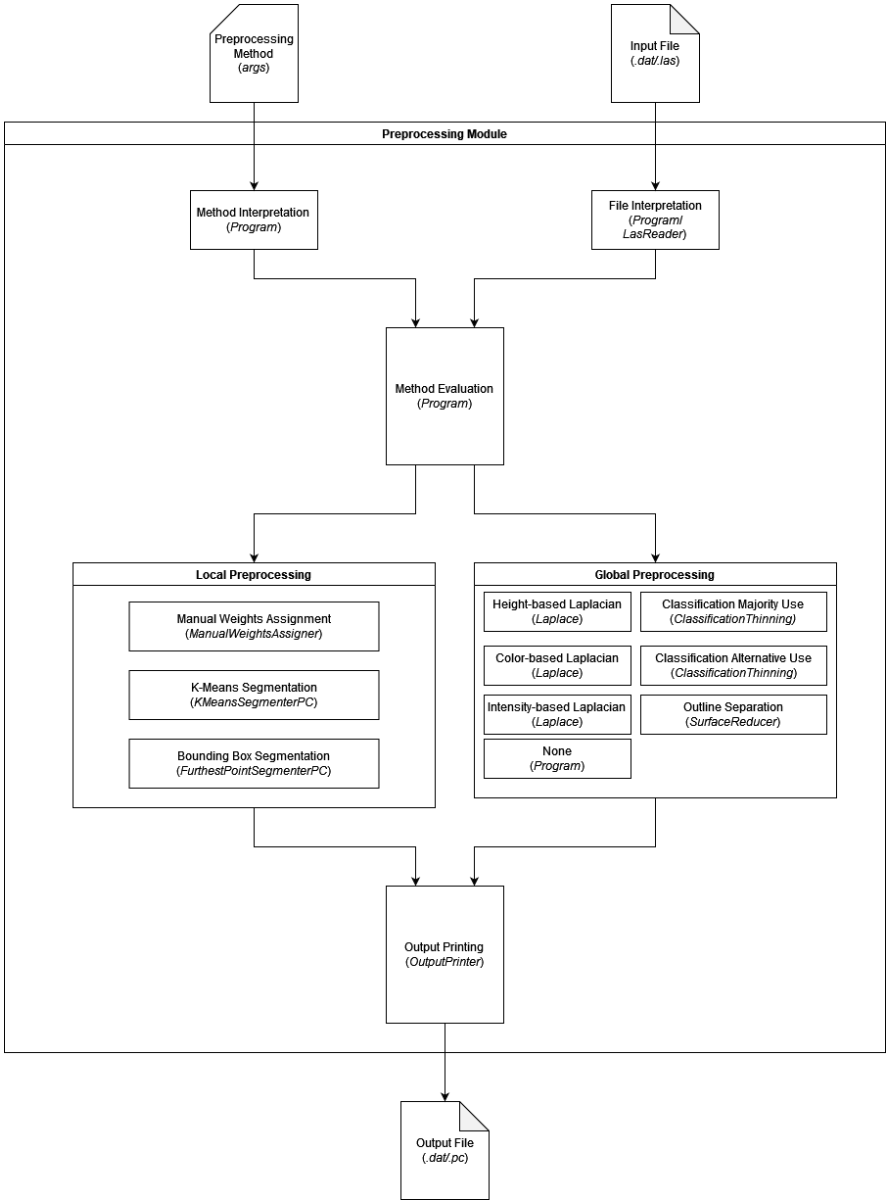


Figure 4.1: Workflow of the preprocessing module

For the local preprocessing, only manual weight assignment was implemented by the author of this thesis, the remaining two approaches (k-means segmentation and bounding box segmentation) were implemented by Bc. Eliška Mourycová and were only incorporated into the weight assignment.

All mentioned local preprocessing methods use weights, whose significance is described in Section 4.2.3. In order to allow local symmetry preprocessing, the original symmetry detection algorithm had to be adjusted and the support for reading weights from the files had to be implemented. For detailed information about the implementation see implementation documentation which is bundled along with the solution. The proposed methods are described with their results in Chapter 5 and Chapter 6.

4.2 Hruda et al. Detection Algorithm

While there are numerous algorithms for symmetry plane detection, this thesis is focused on using the detection algorithm created and implemented by Ing. Lukáš Hruda [10]. The proposed algorithm supports finding global symmetries in a set of points. As the method does not require presence of triangles, this method is viable not only for data formats such as the file format OBJ, but also point cloud formats such as the format LAS. This method is capable of finding both single and multiple symmetry planes in complete, incomplete and even noised data. While the algorithm is capable of working even with the volumetric data it was used mainly for scans which contain only surface points.

4.2.1 Symmetry Measure

The proposed method detects approximate reflective symmetries in the sense described in the Chapter 2. Authors in their study first define a general plane P by its implicit equation $ax + by + cz + d = 0$ and denote the vector $[a, b, c, d]^T$ as p. A vector function $r(p, x)$, which reflects the point $x = [x, y, x]^T$ is computed by Eq. 4.1, where $n_p = [a, b, c]^T$ is the normal vector of the plane P. All components of the function are continuous and differentiable except of the configuration $p = [0, 0, 0, d]^T$, which does not represent a valid plane.

$$r(p, x) = x - 2\frac{n_p^T x + d}{n_p^T n_p} n_p \tag{4.1}$$

In a perfectly symmetrical object X, all points can be reflected over the plane P into another (or the same) point of X by the transformation function $r(p, x)$. This trait can be mathematically written as $r(p, x_i) = x_j$. The existence of approximate symmetries leads to the need of evaluation of the point set symmetry via the plane P. The created metric is called the

symmetry measure and is denoted as $s_x(p)$. The symmetry measure can be computed by Eq. 4.2, where φ represents the similarity function.

$$s_x(p) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \varphi \left(\| r(p, x_i) - x_j \| \right) \tag{4.2}$$

in The similarity function is a decreasing radial function which converges to the zero and $\varphi(0) = 1$. Weights w_{ij} are set to 1 by default and influence of different values will be described later in Section 4.2.3.

All the point pairs x_i, x_j have x_i reflected over the plane P. The distance between the reflected point and x_j is measured and transformed into the similarity using φ. Since the majority of the point pairs have similarity 0, computing it for all the values is highly inefficient. In order to optimize the algorithm, the similarity is computed only for pairs where $||r(p, x_i) - x_j|| \leq \frac{2.6}{\alpha}$.

The uniform grid with the cell size $\frac{2.6}{\alpha}$ is utilized for the computation of the symmetry measure. For every point only values from the given cell and its direct neighbors are used. The obtained symmetry measure values are summed in order to create the cumulative symmetry measure which can be used to quantify the quality of every symmetry plane. After establishing this quantifier, the problem of finding the symmetry plane in the point cloud can be reduced into maximizing the symmetry measure over the data. This can be understood as minimizing the distance between the reflected point and another point in the same point cloud.

4.2.2 Candidate Election and Pruning

More candidate planes ought to be created in order to detect the symmetry plane which is the best according to the given specification. Candidates can be generated by finding the symmetry plane between pairs of points, but since such operation would be time consuming for all points of the cloud, data are first simplified. The candidate plane is then obtained by computing the normal vector and the coefficient d using Eq. 4.3. x_i and x_j in mentioned equation represent individual points of single point pair.

$$\begin{aligned} n_p &= x_i - x_j \\ d &= -n_p^T \left(\frac{x_i + x_j}{2} \right) \end{aligned} \tag{4.3}$$

To detect a satisfactory symmetry plane, the candidate with the highest symmetry measure must be determined. Unfortunately, the count of the candidates is usually high and some planes within the selected sample are

similar. This leads to the implementation of the candidate pruning through utilizing the distance function $D(p_u, p_v)$, see Eq. 4.4. Values a, b, c and d are coefficients of the implicit plane equation. Variables n_{pu} and n_{pv} are normal vectors of planes. The value l_{avrg} is the average distance of the point cloud point from its cetroid.

$$\begin{aligned} \widehat{p} &= \frac{1}{\|n_p\|}\left[a, b, c, \frac{d}{l_{avrg}}\right]^T \\ D(p_u, p_v) &= \begin{cases} \|\widehat{p}_u - \widehat{p}_v\| & n_{pu}^T n_{pv} \geq 0 \\ \|\widehat{p}_u + \widehat{p}_v\| & n_{pu}^T n_{pv} < 0 \end{cases} \end{aligned} \tag{4.4}$$

To prevent multiple identical planes, if the newly created symmetry plane candidate p_v is closer then $\delta = 0.1$ to p_u, the pair is replaced by the averaged plane. Since the algorithm takes a long time for large point clouds, the data are simplified to contain approximately 1000 points. The symmetry measure is computed from the reduced sample and the best candidate is elected. Since the candidates are created from the simplified cloud and the elected plane is usually only close to the best symmetry plane (not identical), the result optimization is required.

Before applying the optimization the point set should be translated together with the initial plane into the origin. This change is necessary mainly for files with larger d (i.e., the distance from the origin). In such files even a small change of the normal vector of the candidate plane can greatly affect the resulting plane. The optimization of candidates is performed by the quasi-Newton optimisation method L-BFGS, which uses the gradient of the symmetry measure.

4.2.3 Weight Significance

As stated above, the mentioned symmetry detection algorithm uses weights which are set to 1 by default. Each weight w_{ij} consists of two separate parts. The dynamic part is dependent on the plane and can represent for example the symmetry of normal vectors or the direction of principle curvatures in corresponding pairs of points with respect to the given plane. Dynamic part of weight is not modified in the solution proposed by the author of this thesis.

Static weights can be set to magnify probability of finding symmetry planes intersecting important parts of data. In our case this enables the local symmetry by empowering the significant data segments. The higher static weight is, the smaller distance between the reflected point and the nearest

point from the data is allowed. This weight can be set automatically by specifying a mathematical function or manually. This thesis will utilize the later option. Support for reading weights from the geodata input files was implemented to allow weight assignment from the proposed preprocessing module.[10] The weights and their possible assignment are described in more detail in [9] and [37].

4.3 Evaluation Web Application

After the symmetry detection is performed, additional evaluation of results may be required. Since humans and machines understand approximate symmetries differently, it is necessary to compare results manually by humans. In order to automatize the mentioned process, web application for comparing two images (with different symmetry planes) was created.

The application was implemented using Java Spring Boot and React frameworks which use REST API to communicate. The program uses PostgreSQL database to persistently save its data. In order to guarantee platform independence, the project was dockerized. The application user interface was implemented using Bootstrap technology to make it responsive to different viewport sizes.

All users are required to log in before performing any action within the experiment. The voting application distinguishes between two types of users: admins and regular users. In every application instance there is one admin user which is created during the first run of the application. The functionality to create additional admins was not implemented but regular users can be created by the admin in a specifically dedicated page. Users are generated in batches and have general name `user_i` where `i` is an order number to keep names unique. The passwords for each user are generated randomly and all passwords are hashed before saving in the database.

All regular users are required to give consent to data analyzing. Users who agree to the experiment rules are redirected into the training section in which they answer predetermined questions on training samples. This section is used to familiarize users with the problem and to tune their ability to distinguish between correctly and incorrectly detected symmetry plane within a data sample. After answering each question, the correct solution is shown. If the question was answered incorrectly, it is kept within the testing poll. The training section is completed only after all questions are answered correctly. The answers within the training sections are not saved in the database.

The competition of the training section is announced via the specialized modal window. The experimental section is almost identical to the training section, the major difference is the lack of the reply validation. The section is completed after answering every question of the given poll. After completing the experiment section, the user is redirected to the site with a consent revoking in case the user no longer wishes to have his replies analyzed. Only answers of the users who granted consent are to be used by the application.

Individual questions can be added into application by the admin using a special form. Each question requires to specify two images. Questions can have their correct option assigned in that case a training question is created. If no correct option is specified, experiment question is created. The user with admin rights can download answers from the experiment section as an Excel file.

4.4 Experiments Setting

The hardware used for implementation and testing was a computer with a dual core processor Intel® Core™ i5-7200U CPU with frequency 2.50GHz. The device has 8GB RAM and the operating system Windows 10.

There were two data sets used for testing. The former contained 5 data files, listed in Table 4.1 and were used namely for global symmetry experiments. The latter contained 8 data files, listed in Table 4.2, and were used for global and local symmetry experiments. Data sets are shown in Figs. 4.2 and 4.3, respectively.

Data sample	Point count	Point format
data1	19,654,636	0
data2	559,802	1
data3	6,541,983	2
data4	12,051,584	2
data5	99,264,449	3

Table 4.1: Global input data statistics

Figures of data sets and their results shown in the main text are in a small scale to keep the text compact. To enable their more detailed inspection, they are included in a bigger scale in Appendices 1,2. All the examples presented in Appendix 1 are demonstrated in contrast to the original set displayed with the online application plas.io, which has been further expanded by adding the expected symmetry plane.

Data sample	Point count
geo_data1	60,244
geo_data2	15,820
geo_data3	13,829
geo_data4	4,897
geo_data5	100,001
geo_data6	20,014
geo_data7	41,853
geo_data8	70,433

Table 4.2: Local data samples statistics

Data from Table 4.1 are saved in LAS format. Besides the point count, the table shows the point format in the LAS file (see Chapter 3). All data (except geo_data5) from Table 4.2 have only coordinates x and y.

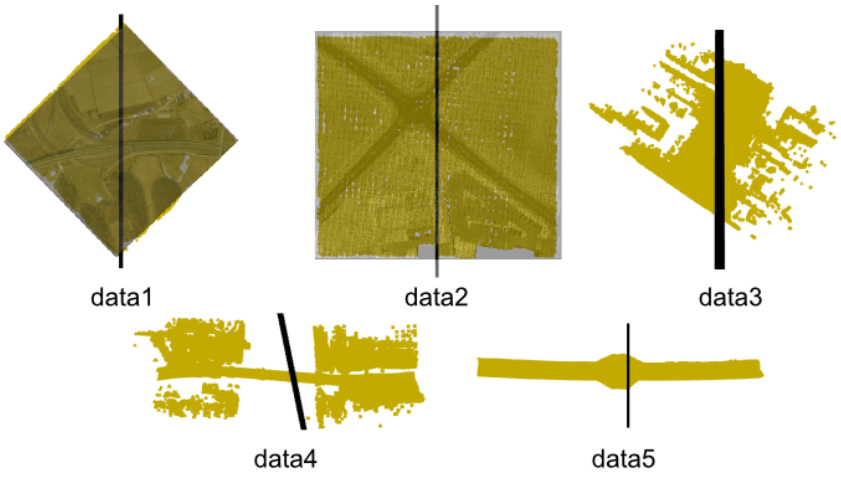


Figure 4.2: Global symmetry input data

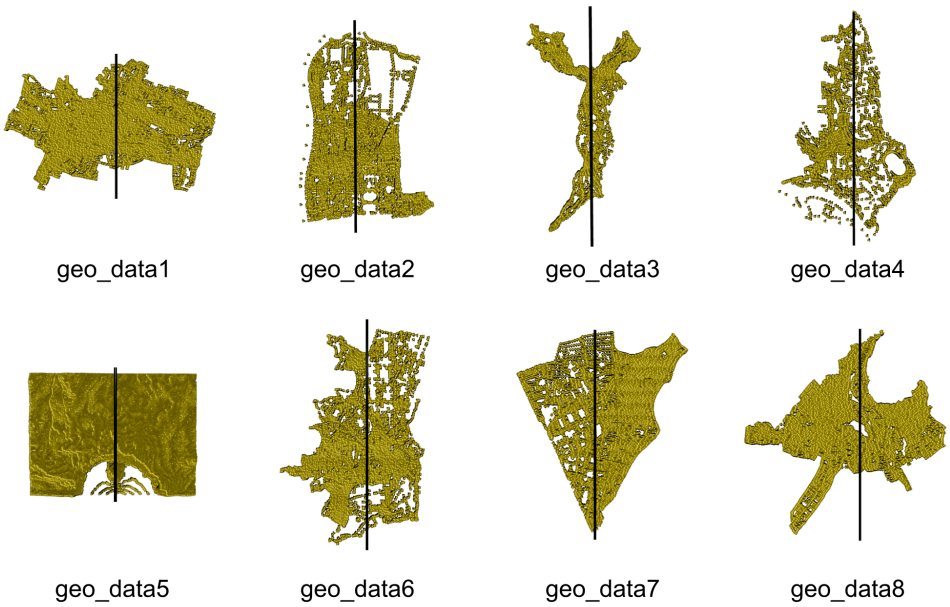


Figure 4.3: Local symmetry input data

5 Proposed Global Preprocessing

The symmetry algorithm described in Section 4.2 has already been proven to work well on regular three dimensional and two dimensional objects. It yielded great results in finding global symmetries in both complete and incomplete data. However, previous work has shown that to perform well over geodetic data, the algorithm needs the additional data preprocessing.

In order to understand the significance of preprocessing, refer to Fig. 4.3, which shows geodetic data that contains only the dominant feature points. The data examples present distinctive shapes with little to no noise. The symmetry detection algorithm introduced in Section 4.2 managed to find relatively suitable symmetry plane in all input data, except the last one (geo_data8). If we compare the results with samples in Fig. 4.2 where the data sets contain also some background points, we have two observations: 1. the symmetry detection is mainly affected by the outline points, 2. the background points in data have a negative influence.

To verify that the symmetry detection method used is mainly affected by the outline points, an experiment was performed for 2D data from the sample collection specified in Table 4.2, where all inner points were removed and only outline points remained. This experiment was performed with the data shown in Fig. 4.3 and in Appendix 2 (pages 1, 9 and 17) extended with additional information. The enhanced files contain not only a cloud of points, but also a triangulation information. Triangulation is saved within the data as the index of vertices of each triangle. Certain triangles are flagged as outline triangles. After listings with all the triangles, the source files present the information about the neighborhood of triangles in the triangulation.

Then, for the given experiment, the program reads all vertices of the outline triangles and removes the duplicate points. As seen in Fig. 5.1 and in Appendix 2 (pages 2, 10 and 18), this method still manages to find acceptable symmetry planes, however, the planes found differ from the planes obtained from the original data. This shows that the inner points do have some influence on the detection algorithm. The detection with only outline points is more prone to abnormalities within the data. The algorithm is then more likely to assume that part of data outline is missing and therefore it is affected by ghost points. Ghost points are points the detection algorithm assumes for perfect symmetry.

One of the main reasons, why the method described in Section 4.2 does not perform for geodetic data as well as for other data types is because of the technique of the scanning described in Chapter 3. The result of this scanning procedure is a square or a rectangular data sample, with the Earth surface as a background. The symmetry detection algorithm in that case tries to find the symmetry plane over the whole set, which leads to finding symmetry planes of the outer data shape instead of considering significant parts such as roads and buildings withing the set. Therefore, one idea how to improve the results is to extract significant points and to apply the symmetry detection on them.

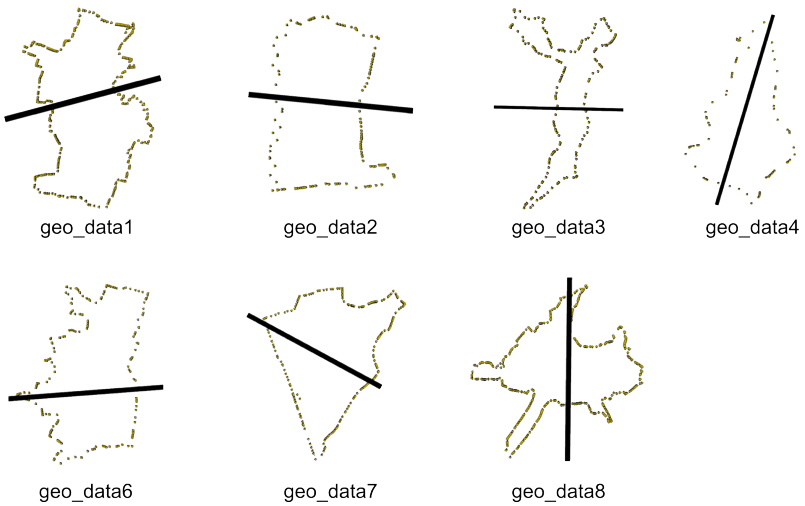


Figure 5.1: Outline geodetic data example

5.1 Laplacian Operator

The Laplacian operator is a derivative operator used to highlight regions of rapid changes in data, therefore, it is often used for edge detection. The difference between this edge detector and others is that it utilizes the second derivative instead of the first one. If we consider the input as a function, the application of the Laplacian operator to a function results in zero values at the places where the edges are suspected (see Fig. 5.2).[15]

It classifies edges based on their inward or outward orientation. Unfortunately, this can sometimes lead to edge duplication in regular images [16]. Since this thesis is mainly focused on point cloud data, this problem does not concern us.

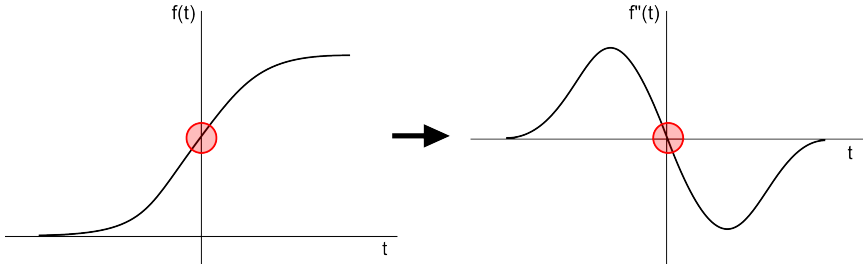


Figure 5.2: Impact of Laplacian on function graph

The Laplacian operator can be applied in two major ways. Assuming we know a function which describes (or approximates) our data set, we might use it directly by performing the divergence of the gradient of the mentioned function (see Eq. 5.1). After computing the equation, we would set the values to be used in a further data filtering.

Laplacian operator in Eq. 5.1 is denoted by symbol $\bigtriangledown^2$. The function f is a continuous function. Expressions $\frac{\partial^2 f}{\partial x^2}$, $\frac{\partial^2 f}{\partial y^2}$ and $\frac{\partial^2 f}{\partial z^2}$ are the second-order partial derivatives of f.

$$\bigtriangledown^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} \tag{5.1}$$

The algorithm described above is used, if the function is known, which is not the case, while handling the point clouds. Therefore, we must use the second option of computing. We can use one of four convolution filters (see Fig. 5.3) which are commonly used for the approximate Laplacian filter.

0	1	0	0	-1	0	1	1	1	-1	-1	-1
1	-4	1	-1	4	-1	1	-8	1	-1	8	-1
0	1	0	0	-1	0	1	1	1	-1	-1	-1
a)			b)			c)			d)		

Figure 5.3: Laplacian operator masks

The mentioned filters differ in being positive (options a and c) or negative (options b and d), each targeting a different type of edges. The positive operator is oriented mainly on the outward edges, while the negative Laplacian performs better while detecting inward edges. Those masks can be also divided whether they evaluate pixels based on four (options a and b) or eight (options c and d) neighboring pixels[16]. In this thesis we used a positive Laplacian operator computed on eight neighboring pixels (option c).

An earlier established mask is applied in the so-called convolution, which is a mathematical operation used to multiply two matrices of the same dimension but different size. The first matrix represents the original image, while the second, usually smaller, matrix represents the convolution filter. The operation itself is performed by sliding the smaller matrix over the bigger one and adding up the products of multiplying at each position of the smaller matrix (see Fig. 5.4 and Eq. 5.2). The resulting number is a new value for the pixel in the middle of the masked section of the original picture. The values at the edges of the original matrix can be either ignored (set to zero) or counted only with neighboring pixels (ignoring the overflowing mask).[6]

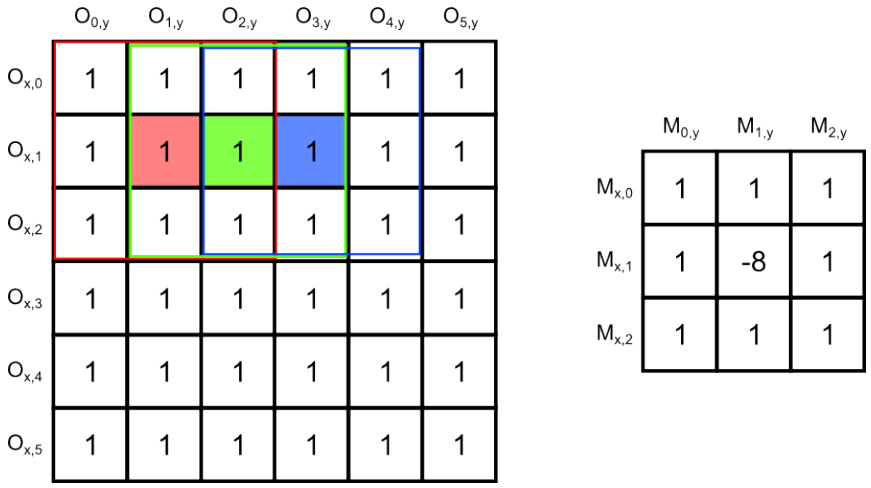


Figure 5.4: Applying convolution filter

In Eq. 5.2 $Lap(x, y)$ is a resulting value of Laplacian operator over indices x and y from original matrix O. Indices n and m are height and width of the convolution filter M, respectively (with convolution mask from Fig. 5.3 $m = n = 3$).

$$Lap(x, y) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} M_{i,j} * O_{(x-1+i),(y-1+j)} \tag{5.2}$$

The problem in our representation is the fact that the point clouds are not ordered in any way, therefore, in order to perform the Laplacian described by the second algorithm, we need to project them into a two-dimensional matrix O. If we consider a point cloud L which consists of l points, then every dimension of O should contain $\sqrt{l}$ matrix elements. The segment ranges s_x and s_y are computed by Eq. 5.3, where x_{max} and x_{min} are maximal and

minimal x coordinate from the point cloud L.

$$\begin{aligned} s_x &= \frac{x_{max} - x_{min}}{\sqrt{l}} \\ s_y &= \frac{y_{max} - y_{min}}{\sqrt{l}} \end{aligned} \tag{5.3}$$

After ranges are computed, every point is projected into the 2D matrix M by Eq. 5.4. Variables x and y are coordinates of points from point cloud L. i and j are indices into M.

$$\begin{aligned} i &= \frac{x - x_{min}}{s_x} \\ j &= \frac{y - y_{min}}{s_y} \end{aligned} \tag{5.4}$$

5.1.1 Height-based Laplacian

This Laplace option had been created for data that contain only coordinates. It is based on the z coordinate, assuming that most of the tested models look for the symmetry plane orthogonal with the ground. After computing the Laplacian operator described above, the resulting height is compared with a predetermined threshold. The threshold is determined based on the maximum possible height and average height (to minimize inaccuracy in case of solitary extremes). Only the points, the absolute value of which is below the threshold and is non-zero, are added to the final points.

Results can be seen in Fig. 5.5. It was discovered that because of the great number of points, projected into the same x and y coordinates (i.e., the points differ only in z coordinate), this method is still viable but it does not produce the best possible results. The best result of this method can be observed in the data5, where the method leads to finding the symmetry along the road.

Some improvements (compare to the original data in Fig. 4.3) can be seen also in data3 and data4, but in those sets the method leads to finding planes orthogonal to the roads. The sets of data1 and data2 showed no changes because all points in the data sets have nearly the same z coordinate. This results in the height-based Laplacian not being able to subtract significant features.

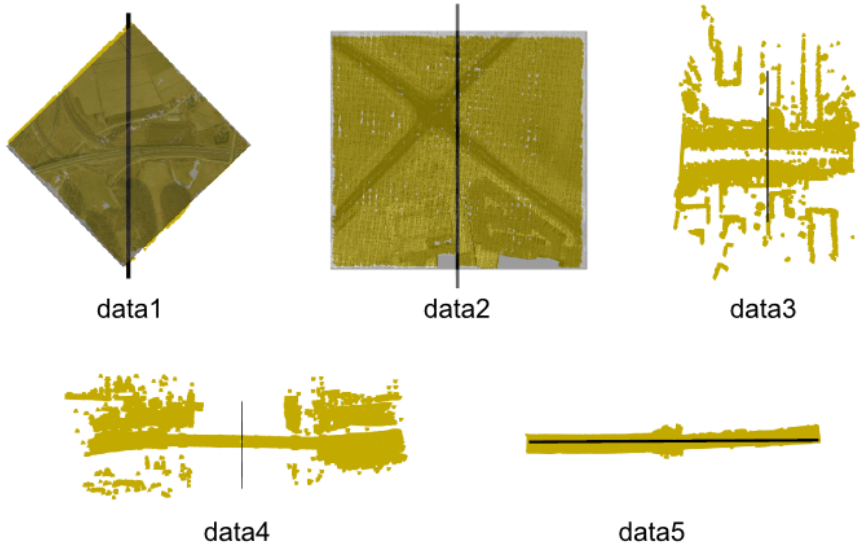


Figure 5.5: Results of height-based Laplacian

5.1.2 Color-based Laplacian

After having identified the problems with the flat samples described in Section 5.1.1, it was decided to implement a more traditional Laplace representation focusing on subtraction data based on color channels. Up to this point, all operations were performed on the LAS data obtained by external application las2txt, which is a part of the liblas library. This software allows one to export the LAS data in a format with pure coordinates, omitting most of the additional information encoded in the mentioned format. Therefore, it was necessary to implement an original LAS reader capable of obtaining all possible records.

Some of the point record formats described in Chapter 3 support a color, which is saved as separate red, green and blue channels. As opposed to a regular 8-bit representation with a range of 0 to 255, LAS saves its data as 16-bit representation of the same. Those two representations can be easily converted from the 8-bit to the 16-bit version, or vice versa, simply by multiplying or dividing the original value by 257, respectively. As the range of the color channels has already been specified (0 - 255 in 8b representation or 0 - 65536 for the 16b representation), we can use the range mentioned above to filter the points after their color values have been modified by Laplace.

It is important to note that not all of point record formats support color, and even if so, there is no guarantee that the color channels contain reas-

onable values (all values might be still set to zero). Although the color information was set in some of the testing data (data sets presented in Fig. 5.6), sometimes the resulting values failed to highlight the edge points due to the changes in the color channels being too subtle to recognize.

As it is clear from the presented results, this method did not improve any detected planes, in cases of data1 and data5 it did not affect the plane at all. In the case of data3 the colors in the set were represented as shades of gray with small changes between each point. This leads to misleading the algorithm and rotating the plane closer to brighter buildings with more noise in color channels.

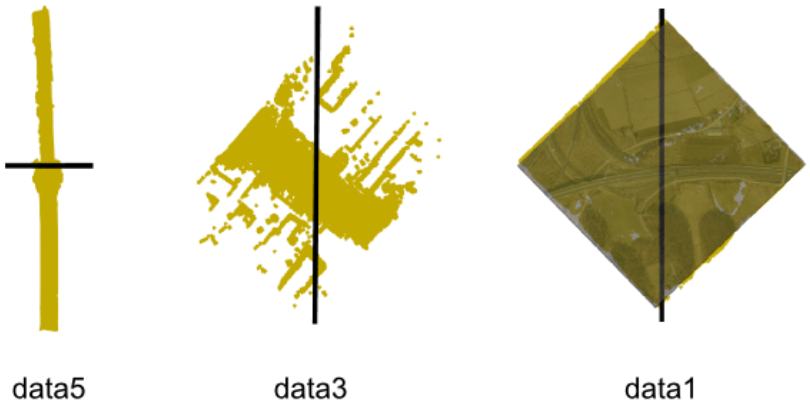


Figure 5.6: Results of color-based Laplacian

5.1.3 Intensity-based Laplacian

At the end of the testing of the color-based Laplacian, I noticed that some data sets, while having no color information, appeared to be in shades of gray (the so called grayscale). This information is caused by the parameter intensity which is present in every point record format. Unlike colors, it often has a height contrast between different parts of data, therefore, it should be a prime candidate for the Laplace filtering.

The intensity parameter is once again represented as a 16-bit unsigned number with the same conversion ratio as stated above. As such, it has a predefined range which can be used to exclude unnecessary points. As the points sharing the same x and y coordinate have similar, if not the same intensity value, this method is less likely to suffer from inaccuracy caused by averaging values as described in Section 5.1.

Unlike its predecessors, this method has proven to be able to detect different subsets of data as seen in Fig. 5.7 in the set data1. Unfortunately, the figure shows that this method does not improve the symmetry detection in an overly complicated set (data3). This method is able to improve the found symmetry (mainly observable in data4) and to remove background points (data1). However, if the sample represents data without any background points and with nearly the same intensity at each point, the detected symmetry plane remains unchanged (data5).

There is another problem which can be observed in data2. As the intensity within background points of this data example changes rapidly, Laplacian fails to remove background segments. Lastly, this method might be influenced by random intensity shifts, as seen in data1, where noise created by Laplacian mislead a potentially great result.

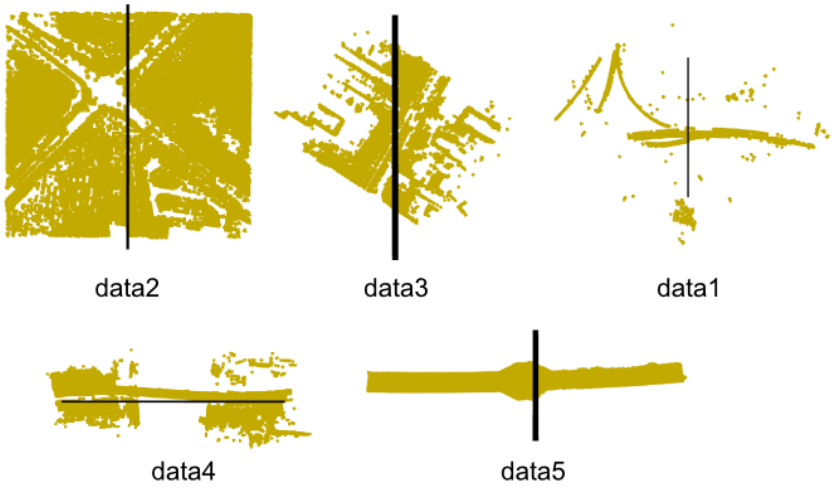


Figure 5.7: Results of intensity-based Laplacian

5.2 Classification Use

As described in Chapter 3, the data in LAS format natively supports classification which is a required parameter in all point formats. The possible classification values are observable in Table 3.1 and Table 3.2. The proposed solution utilizes mentioned parameter to cluster data into two subsets which will be saved in separate files. For data with bigger point counts, the data will require additional filtering by removal of random points.

Two methods of classification use were proposed in this thesis, each useful for a different situation. Mentioned methods are viable only if the LAS data

contains multiple classification values; otherwise, the application performs only removal of random points.

5.2.1 Majority Classification Use

This approach counts the individual classifications within the input file. It is assumed that the most occurring classification has special meaning for the data, therefore, all points with the most often represented classification are separated into a new file. The remaining points are used to create a complementary file.

As the method separates point clouds into two files based on LAS classifications, it requires the input data to contain at least two classification types. From input files specified in Table 4.2 only files data5, data3 and data1 meet the specified requirements. The results of the method can be seen in Fig. 5.8. The method managed to separate feature points from remaining data in files data5 and data3 and symmetry algorithm managed to find satisfactory symmetry planes in mentioned files. Unfortunately, the method did not lead to good results in file data1. Since the points with most represented classification type within the mentioned file were not the feature points of the point cloud, the separation was not successful and, therefore, even the detected symmetry planes were not good.

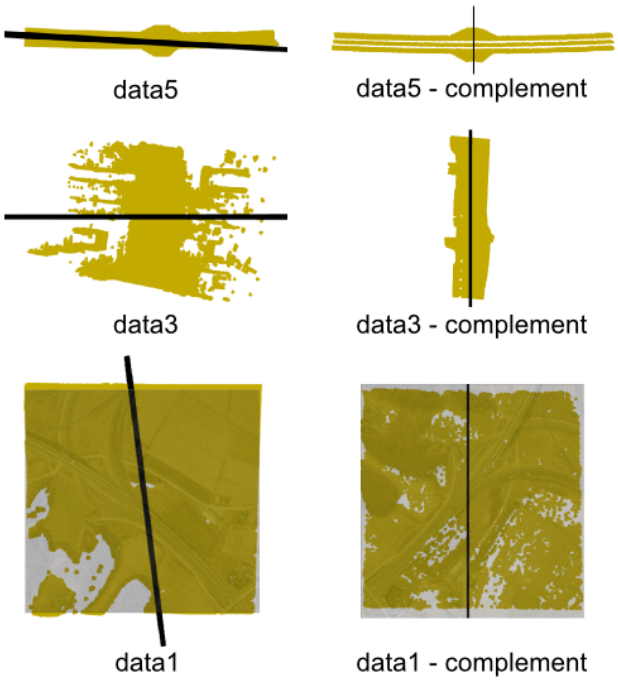


Figure 5.8: Results of classification use

5.2.2 Alternative Classification Use

The classification values defined within the LAS format (see Table 3.1 and Table 3.2) aim to be as specific as possible (e.g. there are 3 classification types for vegetation). As our research is not focused on specific classification value, we can group points with similar classification meaning into new clusters. The aim is to create two data sets, one containing all possible man-made structures and the other one containing vegetation, water planes and other natural phenomena.

This approach is focused on input data containing multiple classification values. Furthermore, the method requires a complete classification, i.e., data consisting of points classified only as types 0 and 1 are not supported. Both of these types represent the "unclassified" option, the difference being that value 1 was evaluated by some clustering algorithm, but it was further impossible to indicate the correct classification.

From the provided input files, only data1 is suitable for this solution. As seen in Fig. 5.9 the method did not manage to separate the feature points from the remaining data. The reason for the failure is that the data while classified does not contain the expected (correct) classification values.

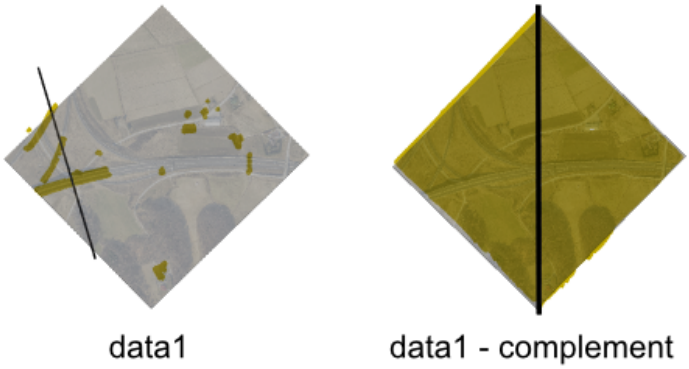


Figure 5.9: Results of alternative classification use

5.3 Data Flattening

In addition to methods proposed above, it is possible to apply flattening to all preprocessing. To demonstrate possible results, an example of flattening over height-based Laplace was presented in Fig. 5.10. If enabled, flattening only sets the z coordinate to 0 after the the required filter has been applied. In most cases, this operation will not worsen the results, it either gives the

same results as before flattening, or improves them, alternatively in few cases helps to find the second symmetry plane.

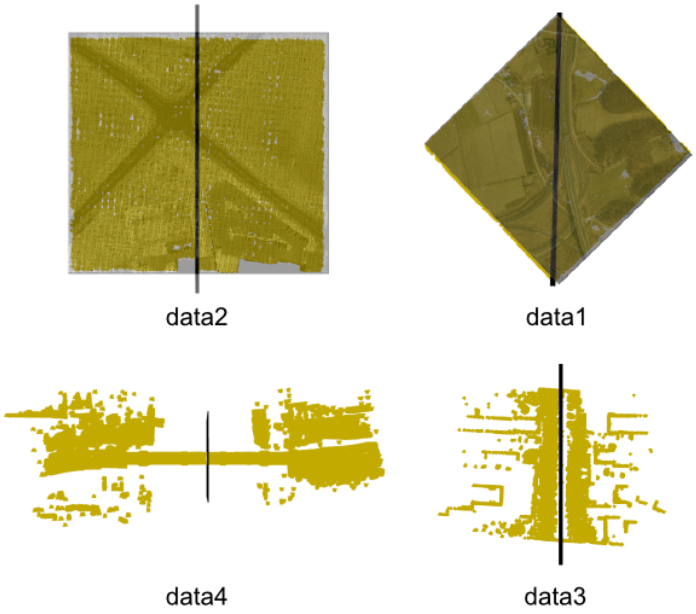


Figure 5.10: Results of flattening over height-based Laplace

Since this method transforms three-dimensional models into two dimensional projection, it is best used with the help of external software to display the data in their original non-flat format. As the data are reduced to a ground plan, it might not be ideal for the data which is expected to have a symmetry plane non-orthogonal to the horizontal axes. To test this option in full extend, the option not to perform any additional preprocessing has been added.

5.4 Preprocessing Evaluation

All the mentioned methods were implemented as a separate module, their use was not required for the overall running of the detection algorithm, however, several methods have proved to improve the found symmetry planes. Unfortunately, no universal global preprocessing was discovered in this thesis, each method had its good and bad results. In addition to preprocessing methods, an innate LAS reader has been implemented, therefore, the external software is no longer required to transform this format into formats supported by provided detection application.

The best results were brought by the first implemented method, the height-based Laplacian. This method works relatively well on sampled data

without background points (with no readable outer shape), however, it is weak against sets with the background information. In such cases, the method usually finds symmetries in the shape of a sample (such as a rectangle or a square). It is also speculated that it would not perform well over the files with a lot of points with the same coordinates in x and y axis. For the previously mentioned reason, the algorithm would be insufficient for the data with caves or chasms. Since this solution uses the height as its argument, it is not possible to find any symmetry planes that are horizontal. If such planes were required, it would be necessary to swap the x or y coordinate with the height and perform a projection in a different direction. All of these changes would have to be made within the application code, as these cases were not expected to be common in geodata.

The second best results were obtained using the majority classification thinning. The problem with this approach is that not all data are correctly classified, and even with classified data, one depends on the data interpretation from unknown programs. On the other hand, unlike the previous method, this solution separates original data into two subsets, therefore, it offers two solutions. As each separated set is considered a stand-alone element, the effect of the noise data, whether it is real noise or unwanted interference from differently classified data, is also reduced.

All the points mentioned about the majority classification thinning also apply in the case of alternate classification thinning, as the main principle remains the same. Both methods differ only in the approach to file division. The first method separated data based on majority classification, while the second one separates data based on the origin of represented objects (man-made objects or natural objects). As the points are generally unclassified in all available test sets, this method performs worse then its majority counterpart.

The worst result was obtained from the color-based Laplace. The main failure of this method was caused by rather sparse representation in the tested sets, where most sets did not contain color channels at all. Those sets, which did, had the problem due to subtle color changes in the whole set and monochrome tones. In one set, the plane was also influenced by noise of the color values, which is this algorithm more vulnerable to.

As with the previous method, the intensity-based Laplace performed rather poorly in most sets. Unlike the color-based version, it managed to subtract the background points, but unfortunately, the method still produced enough noise points to negate possibly good results. The intensity-based and color-based versions do not significantly help to find planes in data without any background points.

The last mentioned solution is the data flattening, which can be used in combination with all the previous methods. This method is able to improve the results in cases where the symmetry detection algorithm is misled by the z coordinate. Similarly to the first proposed method (the height-based Laplace), it is not able to improve the results in data with background points and in cases with an overlapping terrain (caves and cavities). It is important to note that due to the projection of all points to the same z coordinate, especially for samples with a high number of count, the created projection may be difficult to read. It is impossible to find the symmetry planes which are horizontal. This method performs well, although the concept is relatively simple, its importance is mainly in combination with the other preprocessing mentioned above.

6 Proposed Local Preprocessing

In Section 4.2 it was stated that the detected symmetry plane can be influenced by changing the weights of important points. More precisely, the program attempts to find several planes by using weights assigned to pairs of points. Based on this property, we can assume that finding local symmetry planes is only a matter of applying different weights to points within the data. In order to allow effective search and accurate results, it was necessary to make several changes within the implementation of the algorithm described in Section 4.2.

For faster detection, the original algorithm expects the input weights to be symmetrical (i.e., in the point cloud L the weight value of the point L_{xy} is the same as the weight of the point L_{yx}). In order to be able to work even with asymmetrical values, the application has been expanded by adding a variable for switching between modes (symmetrical/asymmetrical weights). Furthermore, the implementation now allows the weights to be read from the original data. In order to retain the weight information within the reduced set, the simplification method had to be adapted as well. A visualiser now highlights points with higher weights (i.e., more important points).

The methods described in this section will be performed with the data shown in Fig. 4.3 and in Appendix 2 (pages 1, 9 and 17). Files geo_data2 and geo_data5 will not be be presented in any of the proposed methods, because they have little to no interesting parts with respect to local symmetries. The data in this chapter are not in the format LAS described in Chapter 3, but in a auxiliary format consisting only of the number of points (the first line) and the point information (coordinates separated by spaces).

6.1 Manually Set Weights

This approach expects the user first to manually separate part of the point cloud in any external software. Points from the separated file are then assigned a different weight value than points from the rest of the original point cloud.

Due to the complexity of the detection algorithm, it is recommended that the subset is not less than one tenth of the original file (based on the

point count). The reason for that is that the detection algorithm performs point simplification, which leads to a weight approximation. If the important points are a small subset compared to the original data, the simplification algorithm is more likely to ignore the weighted points and result in assigning the same weights to all points. This can lead to finding global symmetries instead of local ones, in the extreme case where all points have their weights assigned to zero, the program will no longer be able to detect any plane at all. The implemented prepocessing is able to assign arbitrary weights to important (local) points as well as unimportant (global) points. Several weights were tested, the best results were found in scenarios described below.

6.1.1 Weights 1:0

In this method, the weight 1 is applied to a small section of points, which is considered more important. Remaining points are assigned the weight 0. Points with the weight zero are completely ignored, therefore, this approach should find the symmetry plane only on the basis of points from the secondary file (points with non-zero values).

The problem with this approach is that due to the point simplification described above (and subsequent averaging of weights), it is possible that all points are evaluated as having weight value equal to 0, leading to inability to produce any symmetry plane candidates, which further leads to the failure of the detection algorithm. This phenomenon rarely occurs if a significant point sample is smaller then one tenth of the original data file. However, it is important to note that this problem can be solved either by increasing the number of important points, or by reducing the number of unimportant points. While the same problem persisted even at different weight values, in this case it is the most noticeable, due to inability to produce any results.

In this approach only points with a non-zero weight values influence the detected symmetry plane. This method usually produces planes that do not intersect any mass of the data, i.e., are being collateral with the data outlines. While not necessarily wrong, such results might be less beneficial, therefore, the results obtained by other weight ratios might be preferable.

The measured results can be seen in Fig. 6.1 and in Appendix 2 (pages 3, 11 and 19). In this figure, important data segments (represented by the red section in the data) were selected not to cause the problem described within the first paragraph of this subsection. However, the second described phenomenon concerning the direction of the detected symmetry plane can be seen in all variants in geo_data3 and geo_data8. Furthermore, the described behaviour can be found in geo_data1 (all variants excluding c), geo_data4

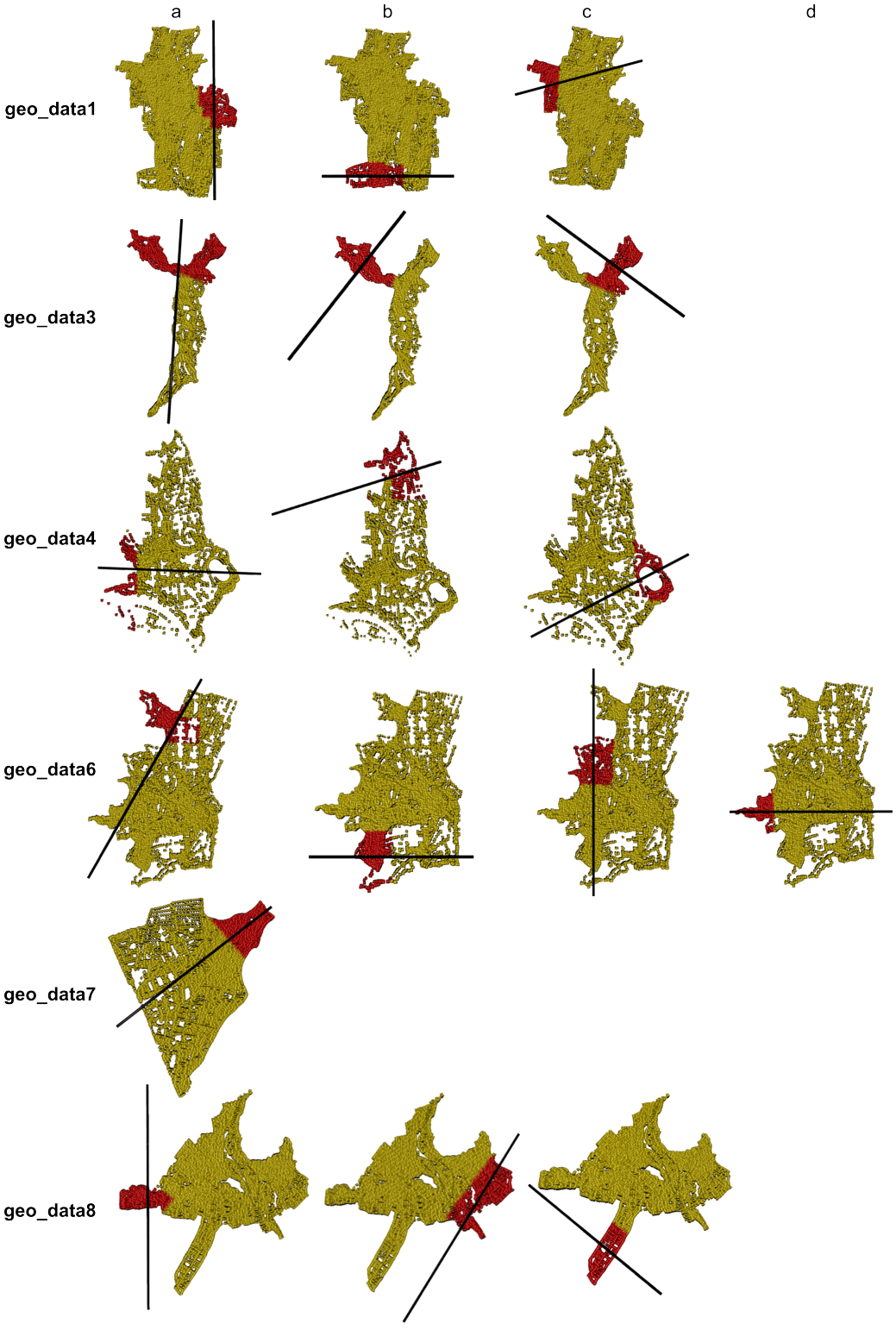


Figure 6.1: Local symmetry results with 1:0 weights

(variant b) and geo_data6 (variants a and b).

6.1.2 Weights 5:1

During the testing, it was observed that if most of the point weights were set to 1, increasing the weights of the remaining points (the significant points from the second file) would change the found symmetry plane. If the weight values of all points are close to each other, the resulting symmetry plane is tilted (opposed to good symmetry planes) and generally not ideal. By approaching the weight ration 5 for important points and 1 for the rest, the results are significantly improved and stabilized on the symmetry plane orthogonal to the data outline. Further increasing the importance of points from the secondary file usually does not lead to further changes. If the significant subset is too small, the detected symmetry plane often coincides with the global symmetry.

The results of this method are shown in Fig. 6.2 and in Appendix 2 (pages 4, 12 and 20). It can be observed that all examples except the variant a in geo_data4 give satisfactory results, which usually differ from the results found by the method described in 6.1.1. As this method considers not only significant points, but even their surrounding, the symmetry detection method succeeds in finding planes, which are not only symmetry planes within the red sections (i.e., local symmetry), but also within the whole data sample (i.e., incomplete global symmetry).

The data file geo_data4 variant a fails to find the local symmetry due to the smaller size of the significant section. This is the only variant of all the tested examples which can benefit from further increasing the ratio between the weights. The method is stabilized on the same result as file geo_data4 variant a in Fig. 6.1 for weights with the ratio 10:1.

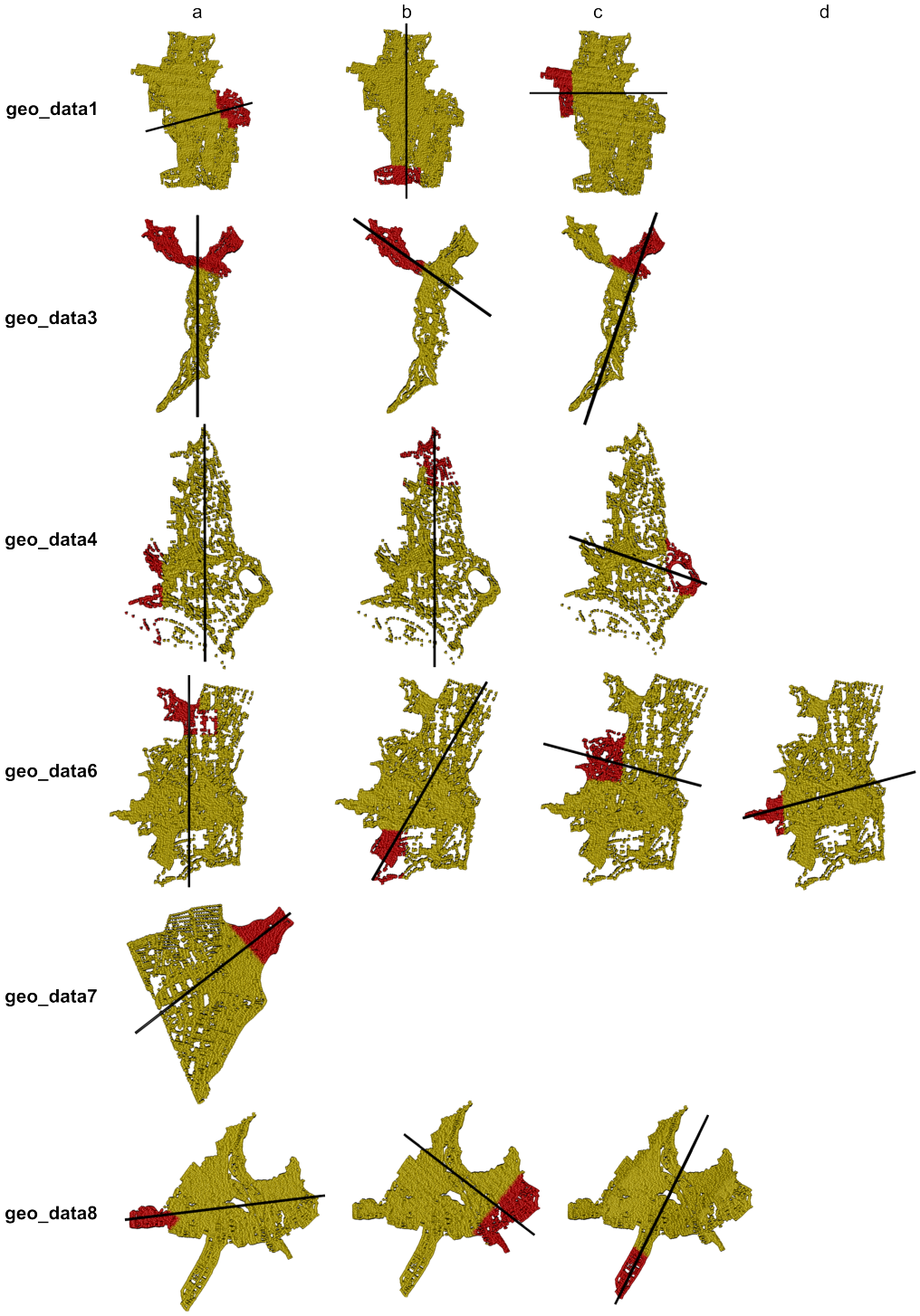


Figure 6.2: Local symmetry results with 5:1 weights

6.2 K-Means Segmentation

This preprocessing approach is heavily based on automatic segmentation using the k-means algorithm, which was implemented by Bc. Eliška Mourycová. Her proposed clustering algorithm distributes points into a predetermined number of clusters.

K-means is a clustering algorithm focused on partitioning the data space in a way that data points within the same cluster are as similar as possible. It aims to minimize the distance to the cluster centroid. This algorithm selects random points as its initial centroids (the same quantity as the number of clusters), while the remaining points are assigned to the cluster of the closest centroid. When all points are clustered, the centroid ought to be recomputed and the process repeated. The algorithm stops when the centroids do not change, the points remain within the same clusters or after reaching a predetermined iteration count [31].

After the data is divided into clusters (with k-means algorithm described above), each cluster is evaluated by a quality quantifier. This quantifier is computed by constructing a line between the cluster centroid and the furthest point within the cluster. All points from the data segment are orthogonally projected onto this line. Distances between the original and projected points are summed and normalized by dividing by the number of points within the cluster. The clusters are then sorted according to the computed value (from highest to lowest) and only the required number of clusters is kept. This described solution leads to an effective and fast data segmentation.

The proposed preprocessing method creates multiple files (the same number as the number of clusters kept), all of which contain the entire original point cloud. The method then assigns weights to each point of the cloud points based on the user input and cluster affinity. While the program allows inputting individual weight ratios, within this thesis only the ratios 1:0 and 5:1 will be described, as they were found to produce results of greater quality.

The evaluation of the results quality for the given weights depends on the quality of the significant segments obtained by the k-means based segmentation algorithm described above. In the examples presented in Fig. 6.3 and Fig. 6.4 we can see segments obtained with default cluster count and default count of perserved clusters. The approach defaultly creates 5 clusters, from which only 3 best clusters are saved. Mentioned default values were established based on experiments (values result in suitable size of clusters). Most of the segments obtained by this configuration are satisfactory, they repres-

ent significant points in a manner similar to manually picked segments in Section 6.1. The problematic segments returned by this method can be seen on the mentioned pictures in variants b and c for geo_data1, variant c for geo_data3 and geo_data6. Variants b and c in geo_data7 and variant a in geo_data8 are not optimal either. In addition, in geo_data7 the algorithm fails to retrieve archipelago in the upper right corner, which is a dominant feature of the file.

6.2.1 Weights 1:0

This method sets the weight 1 to all points contained within one cluster obtained by k-means segmentation. The remaining points have their weight set to 0. In other words, only the points from the mentioned cluster influence the symmetry detection algorithm.

The results of this approach can be seen in Fig. 6.3 and in Appendix 2 (pages 5, 13 and 21). If we consider only the quality of the detected symmetry within the selected segments, the results of this approach are mostly good, the worst detected symmetry planes are generated in the variant b in the file geo_data3 and the variant c in the file geo_data4. The bad result from the first mentioned data is directly due to robustness of the detection algorithm described in Section 4.2, which leads to the assumption of missing parts within significant points. The second mentioned case is caused by the uneven distribution of the points within the segment, where majority of the points is present at the bottom of the cluster.

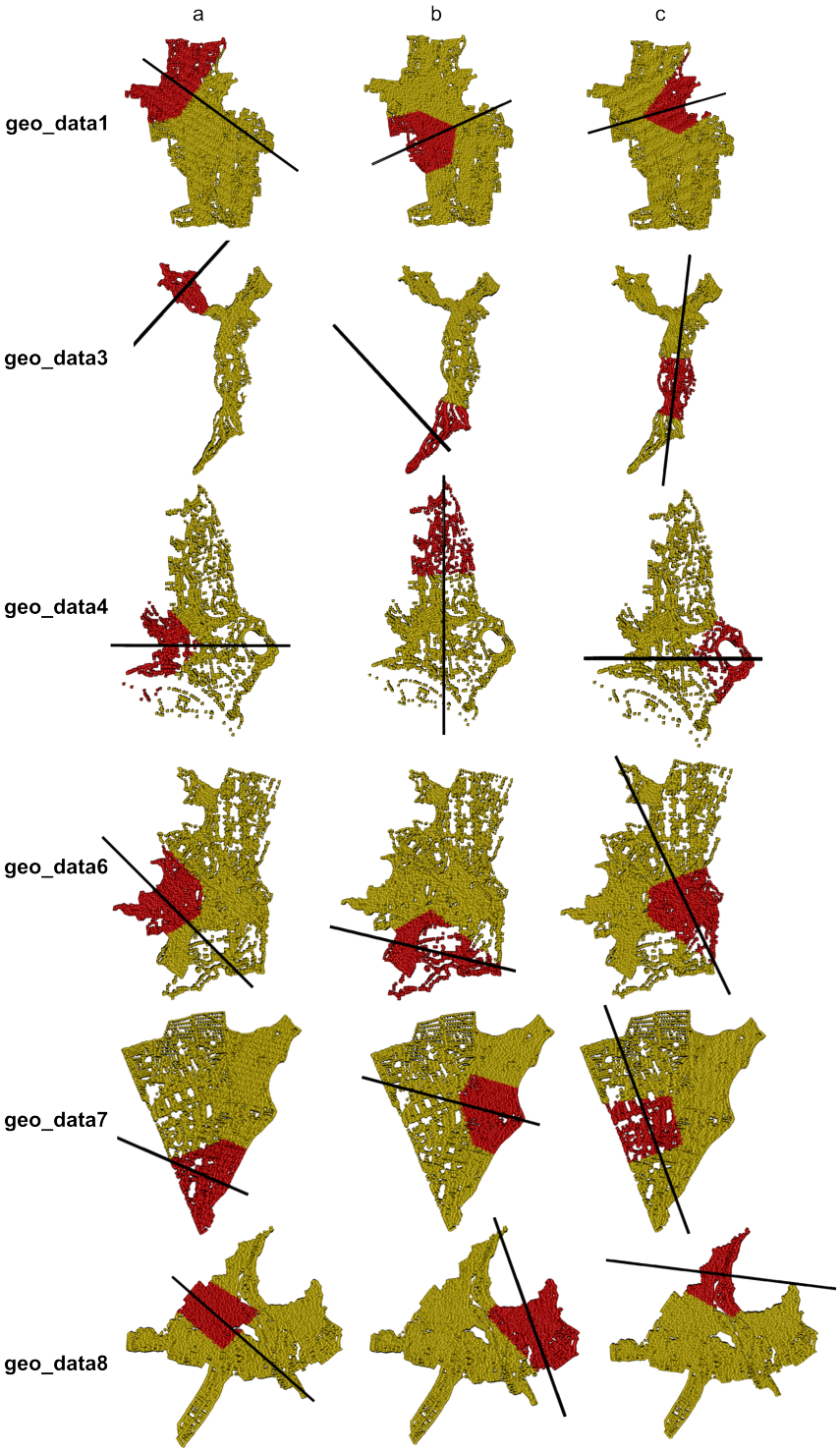


Figure 6.3: Local symmetry over k-means segmentation with 1:0 weights

6.2.2 Weights 5:1

The clustered data from this experiment have their weights set to 5, while the remaining points of the original point cloud have their weights set to 1. This results in a greater influence of selected significant points, while the remaining points still have some significance. Therefore, the detected symmetry plane is more likely to intersect entire point clouds, not just selected sections. In smaller clusters, the accumulation of secondary weights (i.e., weight 1) can exceed the primary weights of significant segments, leading to finding the global symmetry instead.

The detected symmetry with the weight ratio 5:1 to the data segmented by k-means is in many cases similar to the results observed in 6.2.1. In the results displayed in Fig. 6.4 and in Appendix 2 (pages 6, 14 and 22), a decrease in the quality of the detected symmetry can be observed in the variant c in geo_data1. This change is caused by the mentioned phenomenon of overpowering of the primary weights. The variants b and c in geo_data6 and the variant a in geo_data7 show an improvement in the detected symmetries. The options a in geo_data3, geo_data6 and variants b, c in geo_data8 present different planes with a quality comparable to the above ratio in the same segmentation method.

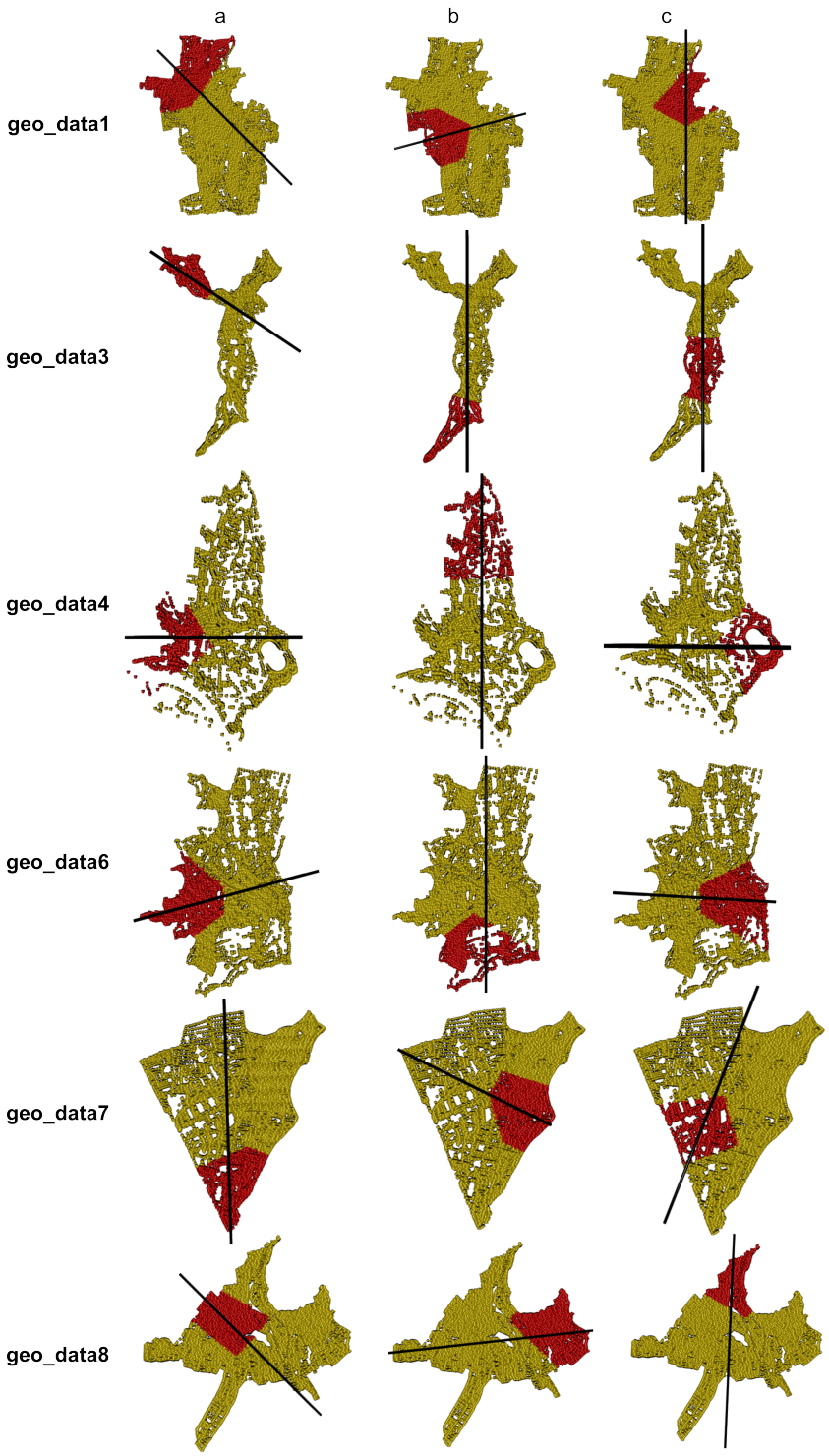


Figure 6.4: Local symmetry over k-means segmentation with 5:1 weights

6.3 Bounding Box Segmentation

The method described in this section is once again based on automatic segmentation provided by Bc. Eliška Mourycová. The proposed method works with the information of the bounding box, used by the original symmetry algorithm described in Section 4.2.

The method starts by retrieving points touching the bounding box of the point cloud and marking the one furthest from the center of the geometry as the starting point. Additional points are then added to the newly created cluster in the breadth-first (BFS) manner. Each point has its neighbors determined by comparing the distance between a pair of points with an adjustable delta. Delta is defaultly set to one tenth of the shortest side of the bounding box.

Points are not enqueued during the BFS, if their distance is less then the average distance to the center of the geometry, or if the required number of points in the cluster has already been reached. This approach guarantees each cluster to have approximately the required number of points. The clusters obtained in this way have their weights within the data file increased on the basis of the user input. As with the previous methods within this chapter, only the weight ratios 1:0 and 5:1 will be presented. Other ratios within this interval provide a tilted symmetry plane that is usually unsatisfactory. Weight ratios higher then 5:1 no longer change the detected symmetry plane.

The bounding box segmentation algorithm depends on the secondary arguments, namely the cluster count and the cluster count kept, which default on the values 10 and 3, respectively. The segments created with these values are presented in Fig. 6.5 and Fig. 6.6. The implemented method is able to detect significant segments in all tested data, furthermore, the returned segments have smoother transitions, their edges are of circular shape. This minimizes errors in detection caused by non-existing missing parts which is the algorithm described in Section 4.2 prone to.

In the presented examples, variant c in geo_data7 is missing, because it is the same as variant b in the same data. This duplicity is caused by data not having any additional interesting segments. A similar phenomenon can be observed in the option c in geo_data3. The later sample was kept in preview because it resembles variant b from the same file, but not was not identical.

6.3.1 Weights 1:0

Within this weight ratio, only segmented points affect the detected symmetry plane. The results of this method are presented in Fig. 6.5 and in Appendix 2 (pages 7, 15 and 23). Due to the circular edge of the segments in all variants, this solution is less likely to lead to the symmetry plane collateral with the outline of the data. Such results can be seen only in the variant c in geo_data1, geo_data6 and in the variant b in geo_data3. The detected planes in the variant b in geo_data3 and variant a in geo_data6 can again be explained by the strong robustness of the detection algorithm (described in Chapter 4.2), which assumes the section shape to be of the damaged object. The symmetry detection over other examples managed to find satisfactory symmetry planes.

6.3.2 Weights 5:1

Most points within this approach have their weights set to 1, only the points contained within individual clusters obtained from segmentation have their significance magnified by setting their weights to 5. As seen in Fig. 6.6 in Appendix 2 (pages 8, 16 and 24), the results from this weight ratio are similar to those from 6.3.1. In the data we can observe that in variants a and c in geo_data1 and all variants in geo_data6, the use of this ratio leads to finding different symmetry planes of the same quality as the previous ratio.

Variants b and c in geo_data3 had their weights in significant parts overpowered by the weights of majority of the points (i.e, due to the point simplification described in 6.1, all the points share same weight). As a result, algorithm found global symmetries instead of local ones. The variant a in geo_data8 suffers from false-positive detection due to the circular edge of the segment which unfortunately strongly resembles the shape of the coast within this section of the data.

It is important to note that this ratio is likely to benefit from changing the secondary arguments in the segmentation method for file geo_data3. Reducing the number of clusters for this file can lead to larger segments, which would be more resistant to the weight overpowering. This phenomenon might lead to better symmetry plane detection for this file in particular, but it is more likely to degrade the results for the remaining files as their segments are already relatively large.

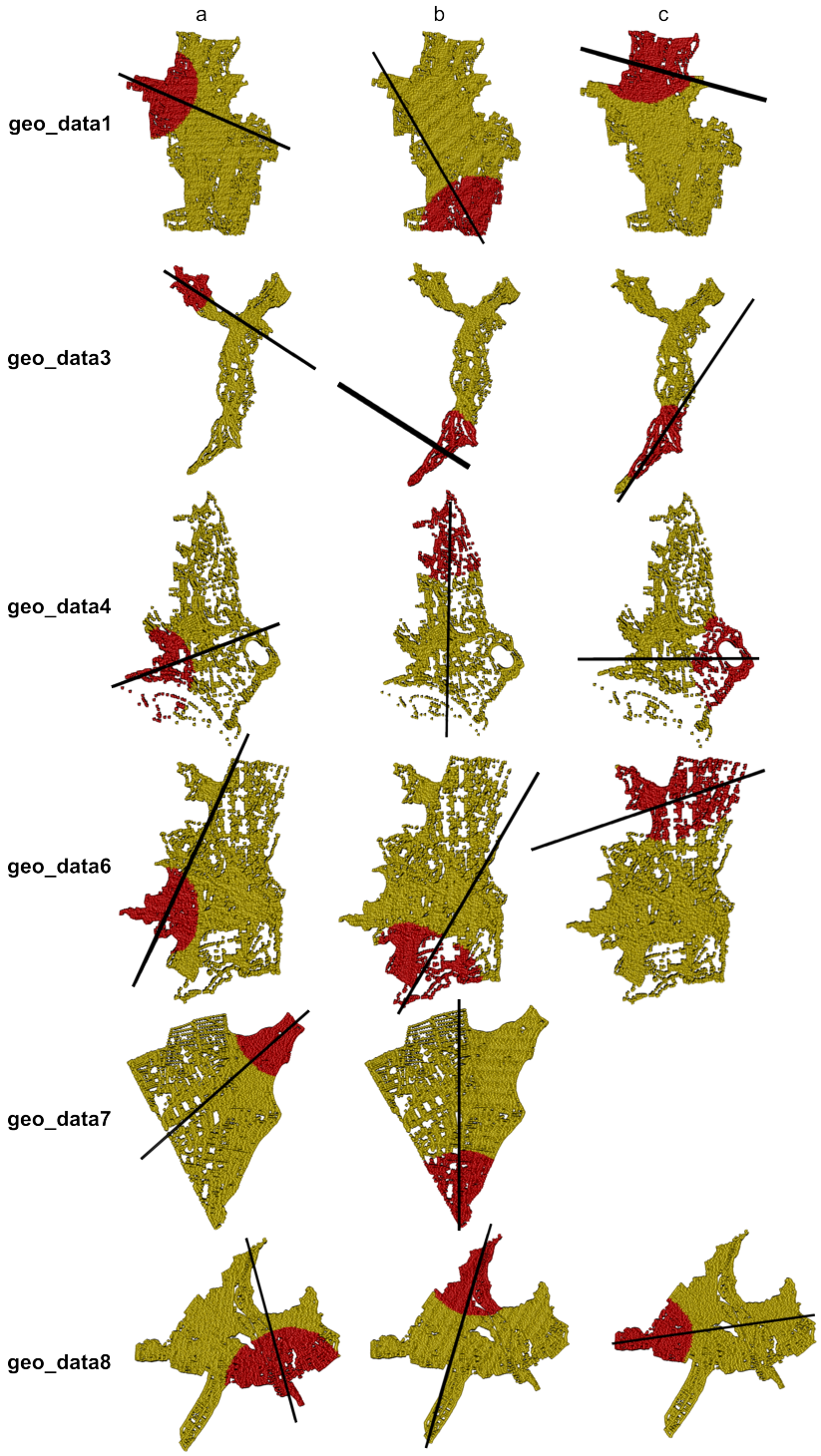


Figure 6.5: Local symmetry over bounding box segmentation with 1:0 weights

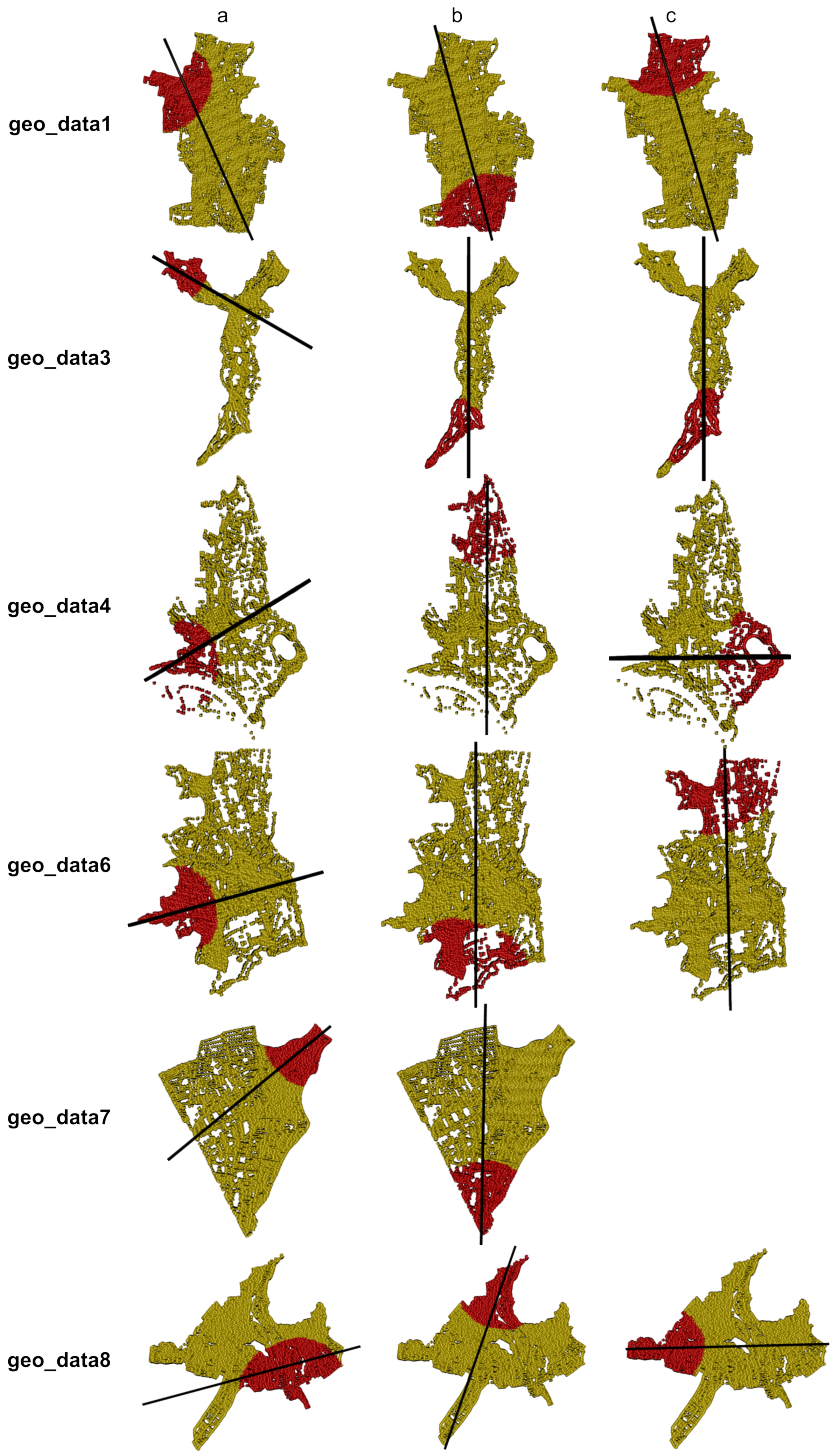


Figure 6.6: Local symmetry over bounding box segmentation with 5:1 weights

6.4 Preprocessing Evaluation

Similar to global methods described in Chapter 5, all methods for local symmetry were implemented in a module separated from the original symmetry detection application. In addition to the implementation of those preprocessing methods it was necessary to slightly modify the original detection algorithm, specifically it was necessary to enable reading symmetry weights from input files and allow work with asymmetric weight inputs (to speed up the original algorithm, symmetrical weights were suspected). To increase the quality of the user experience, the color assignment based on weights has been implemented, resulting in visually separate sections of data with different significance.

Within this chapter, three major methods were proposed along with two weight ratios. In the following text separation methods will be evaluated based on their clustering capability, not on the quality of the symmetry plane found within them. Similarly, weight ratios will be judged only on their quality of the detection, not on the accuracy of the separation to which they were applied.

Manual preprocessing ought to be mentioned separately when evaluating segmentation algorithms, as the quality of the segments depends only on the user's intention and skill to select the appropriate points. Therefore, this approach allows more freedom at a price of higher demands on the user and his time (points have to be separated in an external software). The use of manual input is especially useful in situations where other methods do not provide satisfactory results. In this thesis, the mentioned method was also used as a proof of weights' usability for local symmetry detection.

The remaining two segmentation algorithms offer similar results, which can be observed mainly in geo_data4, geo_data3 and geo_data6. Unfortunately, both methods fail to detect an intrusion in the file geo_data3 in upper right-hand corner, which would be manually declared as significant. Similarly, both methods were unable to offer in geo_data8 archipelago located in the lower left-hand corner, which is a prominent feature of the file. Furthermore, k-means sometimes presents segments which would be unlikely picked manually, these segments can be observed mainly in the variant c in geo_data6, variants b and c in geo_data7 and the variant a in geo_data8 in Fig. 6.3 and Fig. 6.4. Thanks to these segments, the bounding box method worked better in the tested files. Additionally, the bounding box approach results in circular edges of segments, which are less likely to adversely affect the symmetry detection algorithm.

While the bounding box algorithm is more preferable in the presented

data examples, it would perform poorly in non-convex data if a significant segment was located in the non-convex part. This problem can be derived directly from the algorithm specification described in Section 6.3. Similarly, if a significant part is located in the mass of the data, k-means is likely to provide more satisfactory results. The bounding box approach is also more likely to be affected by the rotation of the coordinate system.

After testing weight ratios 1:0 and 5:1 on all files and preprocessing methods described within this chapter, it was concluded that in most cases the ratio 1:0 offers better results. The second proposed weight ratio sometimes leads to major weight being overpowered, which leads to finding the global symmetry plane instead of local one within a significant segment. Using the weight ratio 1:0 sometimes detects the symmetry plane that would not be picked manually, because the human brain tends to consider even surroundings of a significant section (much more like a 5:1 weight ratio).

7 User Evaluation of Symmetry

The web evaluation application described in section 4.3 was deployed on a computer from the university domain and tested on sequence of 18 questions. The mentioned questions were focused on comparing pairs of detected symmetry planes. In order to evaluate detected symmetries, each of the rows from Fig. 7.1, which represents a single geodetic file, was transformed into three comparison questions.

In each question the order of images has been randomized (i.e., the image, which is assumed to contain better symmetry plane, is not always on the same position). Each question set (i.e., 3 questions based on the same input data) contain one image of unprocessed data (i.e., the data, which has not been modified by preprocessing module) and two images of data with modified weights. The input data with modified weights has been manually selected from results obtained from approaches described in Section 6.1, Section 6.2 and Section 6.3. It was decided that only global symmetries will be evaluated. Based on an advice from HCI experts from the Czech Technical University in Prague, all images contain vertical symmetry axis, to prevent axis rotation to influence user input. As none of the data express perfect symmetry, only approximate symmetries would be evaluated.

The question sequence was completed by 36 anonymous users from the University of West Bohemia (11 students and 9 employees), the Czech Technical University in Prague (2 employees) and the University of Maribor (5 students and 9 employees). Their results were analyzed and a graph in Fig. 7.2 created. The bar chart illustrates the number of votes for each question option. As each three questions are focused on one data file (with three different symmetry planes), the computed votes have been aggregated.

It can be observed that in case of data with 'strong' symmetry in one direction, the users voted for, e.g., geo_data8. In case of two 'dominant' symmetry directions, the answers were divided between them, see geo_data1. However, the results show that the users have problems detecting and evaluating weaker approximate symmetries. The mentioned deduction is based on higher evaluation value for the data option b in geo_data3 than the option c. Similarly option a in geo_data8 was evaluated as less symmetrical then option b. The results also show that the users are not unanimous in their toleration to extra points in the data sets which 'spoil' the symmetry.

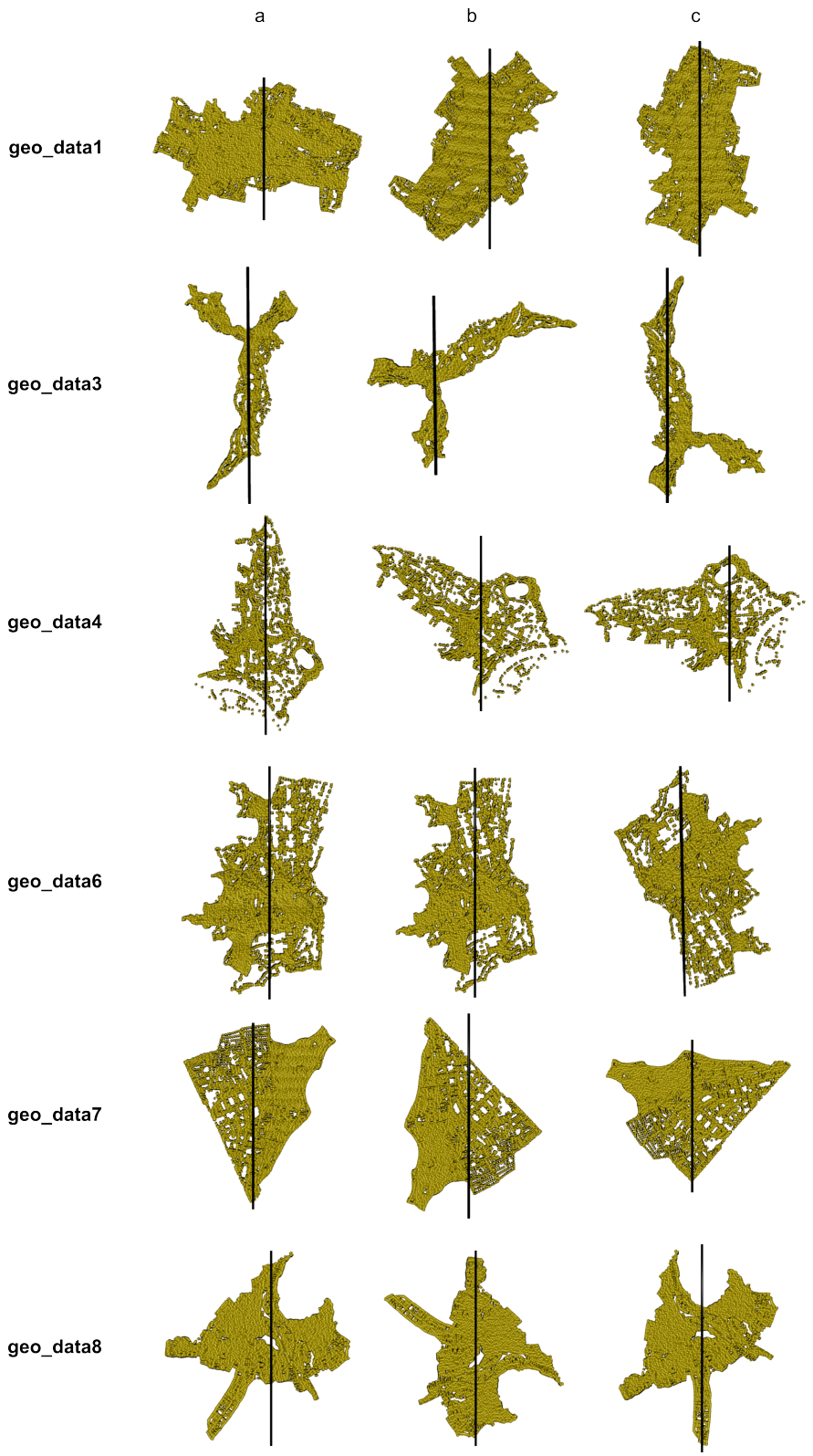


Figure 7.1: Evaluation data

The obtained results were analyzed also according to the information about user's status (a student/an employee) and university (Pilsen/Maribor/Prague). It was observed, that all users share similar understanding of symmetry, because the differences were only minor. The greatest difference was observed between empoyees and students in questions related to geo_data7, where students evaluated option a as more symmetrical and employees prioritized option c. The remaining questions were evaluated the same by both categories.

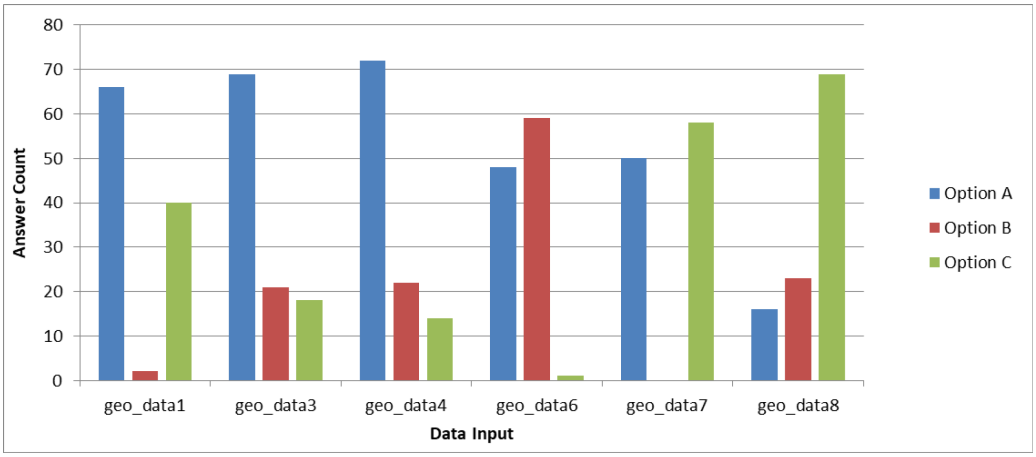


Figure 7.2: Evaluation results

The measured results also prove that the human and machine understanding of symmetry differs. While the humans evaluate symmetry even based on rotation, size and context of data, machine approach aims to maximize some symmetry metric with the data. Deeper insight into the human symmetry perception is a question for more detailed future research.

8 Conclusion

In this thesis, a preprocessing module for symmetry in geodata was implemented. This module contains a reader capable of interpreting geodetic data (LAS format and regular point clouds) and implementation of methods to improve plane symmetry detection in geodata. The preprocessing methods implemented within the module can be divided into approaches improving detected global symmetries and approaches enabling or improving local symmetries.

The proposed methods for global symmetry detection were able to refine the results of uncomplicated data, but no satisfactory solution was developed in files with background points. An approach using LAS classification was proposed to remove background points, but due to the unclassified data, the method did not lead to any refinements. Several solutions based on the Laplacian operator were suggested, but none offered a noticeable improvement in all input examples.

In order to apply methods to local symmetry detection, the original detection algorithm had to be slightly modified. The approach for local symmetry detection based on weights modification was first tested on manual inputs, later automatic segmentation algorithms were provided. There were two possible weight settings proposed for local symmetry. It was proved that the suggested methods provide satisfactory results for most of the tested data, the modification of the weights in the original detection algorithm successfully allowed local symmetry detection.

Due to a cooperation with the Faculty of Electrical Engineering of Czech Technical University in Prague on user tests it was established that instead of implementing the preprocessing module as a web application, a web application for evaluating results of this thesis would be created. The web application for evaluation was implemented as separate software.

The proposed solution can be expanded by implementing additional segmentation methods for local symmetry detection. Furthermore, the local symmetry solution could be applied to data of different types (i.e., data that is not geodetic). The suitability of individual approaches could be automatically evaluated and the best solution for each data proposed.

Bibliography

[1] Buda, A. B. – Mislow, K. A Hausdorff chirality measure. *Journal of the American Chemical Society*. 1992, 114, 15, s. 6006–6012. doi: 10.1021/ja00041a016. Available at: https://doi.org/10.1021/ja00041a016.

[2] Cailliere, D. et al. 3D mirror symmetry detection using Hough transform. *2008 15th IEEE International Conference on Image Processing*. 2008, s. 1772–1775.

[3] Cicconet, M. – Hildebrand, D. – Elliott, H. Finding Mirror Symmetry via Registration and Optimal Symmetric Pairwise Assignment of Curves. s. 1749–1758, 10 2017. doi: 10.1109/ICCVW.2017.206.

[4] Combès, B. et al. *Automatic symmetry plane estimation of bilateral objects in point clouds* [online]. IEEE Conference on Computer Vision and Pattern Recognition, 2008. [cit. 2021/10/21]. Available at: https://hal.inria.fr/inria-00331758.

[5] Eck, D. *About Symmetries of the Plane* [online]. Math, 2008. [cit. 2021/10/4]. Available at: https://math.hws.edu/eck/math110_s08/symmetries/index.html.

[6] Fisher, R. et al. *Convolution* [online]. HIPR2, 2003. [cit. 2021/09/27]. Available at: https://homepages.inf.ed.ac.uk/rbf/HIPR2/convolve.htm.

[7] Gao, L. et al. PRS-Net: Planar Reflective Symmetry Detection Net for 3D Models. *IEEE Transactions on Visualization and Computer Graphics*. 2021, 27, s. 3007–3018.

[8] Harris, A. B. – Kamien, R. D. – Lubensky, T. C. Molecular chirality and chiral parameters. *Rev. Mod. Phys.* Oct 1999, 71, s. 1745–1757. doi: 10.1103/RevModPhys.71.1745. Available at: https://link.aps.org/doi/10.1103/RevModPhys.71.1745.

[9] Hruda, L. – Kolingerová, I. – Váša, L. *Robust, fast and flexible symmetry plane detection based on differentiable symmetry measure: supplementary material* [online]. [cit. 2021/09/30]. Available at: http://meshcompression.org/tvcj-2020.

[10] Hruda, L. – Kolingerová, I. – Váša, L. *Robust, fast and flexible symmetry plane detection based on differentiable symmetry measure*

[online]. The Visual Computer, 2011. [cit. 2021/09/30]. Available at: https://doi.org/10.1007/s00371-020-02034-w.

[11] Ji, P. – Liu, X. A fast and efficient 3D reflection symmetry detector based on neural networks. *Multimedia Tools and Applications*. 2019, 78, s. 35471 – 35492.

[12] Kakarala, R. – Kaliamoorthi, P. – Premachandran, V. Three-Dimensional Bilateral Symmetry Plane Estimation in the Phase Domain. s. 249–256, 06 2013. doi: 10.1109/CVPR.2013.39.

[13] Korman, S. et al. Probably Approximately Symmetric: Fast Rigid Symmetry Detection With Global Guarantees. *Computer Graphics Forum*. Aug 2014, 34, 1, s. 2–13. ISSN 0167-7055. doi: 10.1111/cgf.12454. Available at: http://dx.doi.org/10.1111/cgf.12454.

[14] Kuz'min, V. – Stel'makh, I. Possible approach to the quantitative evaluation of the asymmetry of molecules. *Journal of Structural Chemistry - J STRUCT CHEM-ENGL TR*. 07 1988, 28, s. 498–503. doi: 10.1007/BF00749581.

[15] *OpenCv – Laplace Operator* [online]. OpenCV, 2019. [cit. 2021/09/27]. Available at: http://man.hubwiz.com/docset/OpenCV.docset/Contents/Resources/Documents/d5/db5/tutorial_laplace_operator.html.

[16] *tutorialspoint – Laplacian Operator* [online]. tutorialspoint, 2021. [cit. 2021/09/27]. Available at: https://www.tutorialspoint.com/dip/laplacian_operator.htm.

[17] *LAS SPECIFICATION* [online]. ASPRS, 2011. [cit. 2021/09/28]. Available at: https://www.asprs.org/wp-content/uploads/2010/12/LAS_1_4_r13.pdf.

[18] *What is lidar?* [online]. National Oceanic and Atmospheric Administration, 2021. [cit. 2021/09/28]. Available at: https://oceanservice.noaa.gov/facts/lidar.html.

[19] Lipman, Y. et al. *Symmetry Factored Embedding and distance* [online]. ACM Transactions on Graphics, 2010. [cit. 2021/10/21]. Available at: https://collaborate.princeton.edu/en/publications/symmetry-factored-embedding-and-distance.

[20] Marola, G. On the Detection of the Axes of Symmetry of Symmetric and Almost Symmetric Planar Images. jan 1989, 11, 1, s. 104–108. ISSN 0162-8828. doi: 10.1109/34.23119. Available at: https://doi.org/10.1109/34.23119.

[21] Mitra, N. J. – Guibas, L. – Pauly, M. Partial and Approximate Symmetry Detection for 3D Geometry. *ACM Transactions on Graphics (SIGGRAPH)*. 2006, 25, 3, s. 560–568.

[22] Pablo Speciale, A. C. M. R. O. – Pollefeys, M. A Symmetry Prior for Convex Variational 3D Reconstruction. In *European Conference on Computer Vision (ECCV)*, 2016.

[23] Petitjean, M. Chirality and Symmetry Measures: A Transdisciplinary Review. *Entropy*. 2003, 5, 3, s. 271–312. ISSN 1099-4300. doi: 10.3390/e5030271. Available at: https://www.mdpi.com/1099-4300/5/3/271.

[24] Podolak, J. et al. A Planar-Reflective Symmetry Transform for 3D Shapes. *ACM Trans. Graph.* jul 2006, 25, 3, s. 549–559. ISSN 0730-0301. doi: 10.1145/1141911.1141923. Available at: https://doi.org/10.1145/1141911.1141923.

[25] Qi, C. R. et al. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, 2016. Available at: http://arxiv.org/abs/1612.00593. cite arxiv:1612.00593 Comment: CVPR 2017.

[26] Qi, C. R. et al. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In Guyon, I. et al. (Ed.) *Advances in Neural Information Processing Systems*, 30. Curran Associates, Inc., 2017. Available at: https://proceedings.neurips.cc/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf.

[27] Rassat, A. Un critère de classement des systèmes chiraux de points à partir de la distance au sens de Haussdorf. *Comptes-rendus des séances de l'Académie des sciences. Série 2, Mécanique-physique, chimie, sciences de l'univers, sciences de la terre*. 1984, 299, 2, s. 53–55.

[28] Rassat, A. – Làszló, I. – Fowler, P. W. Topological Rotational Strengths as Chirality Descriptors for Fullerenes. *Chemistry – A European Journal*. 2003, 9, 3, s. 644–651. doi: https://doi.org/10.1002/chem.200390072. Available at: https://chemistry-europe.onlinelibrary.wiley.com/doi/abs/10.1002/chem.200390072.

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

[30] Sharma, B. *What is LiDAR technology and how does it work?* [online]. Geospatial Media and Communications, 2021. [cit. 2021/09/28]. Available at: https://oceanservice.noaa.gov/facts/lidar.html.

[31] Sharma, P. *The Most Comprehensive Guide to K-Means Clustering You'll Ever Need* [online]. Analytics Vidhya, 2019. [cit. 2022/02/24]. Available at: https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/.

[32] Shi, Y. et al. SymmetryNet: Learning to Predict Reflectional and Rotational Symmetries of 3D Shapes from Single-View RGB-D Images. *ACM Trans. Graph.* nov 2020, 39, 6. ISSN 0730-0301. doi: 10.1145/3414685.3417775. Available at: https://doi.org/10.1145/3414685.3417775.

[33] Simari, P. D. – Kalogerakis, E. – Singh, K. Folding meshes: hierarchical mesh segmentation based on planar symmetry. In *SGP '06*, 2006.

[34] Strnad, D. Symmetry detection using neural networks. 2022.

[35] Sun, Y. et al. Joint Map and Symmetry Synchronization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[36] Thrun, S. – Wegbreit, B. Shape from Symmetry. 2, s. 1824– 1831 Vol. 2, 11 2005. doi: 10.1109/ICCV.2005.221. ISBN 0-7695-2334-X.

[37] Uhrová, K. – Kolingerová, I. *Generalized symmetry of geometric data* [online]. [cit. 2021/09/30]. Available at: http://meshcompression.org/tvcj-2020.

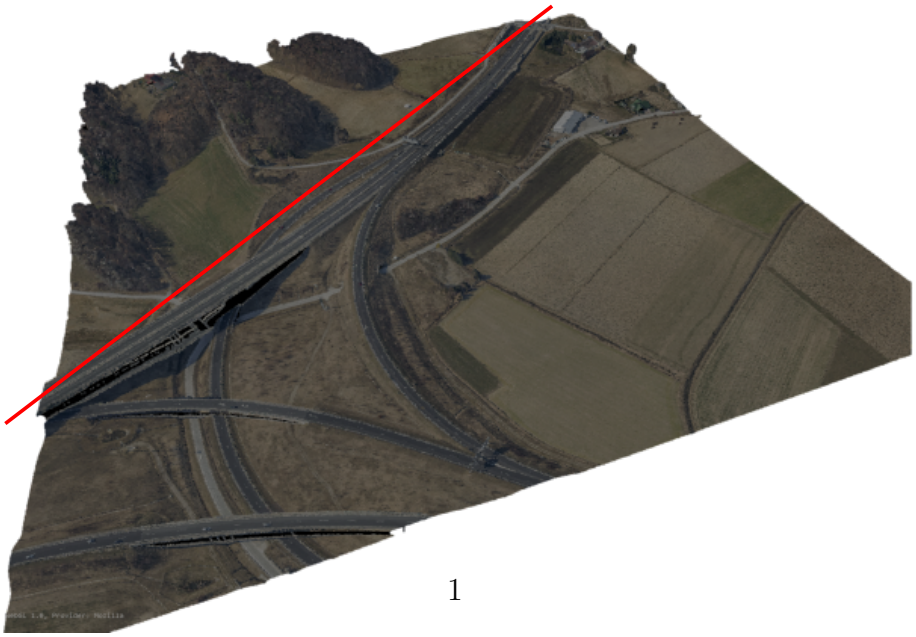
Appendix 1

data1

Original data visualized in plas.io with expected symmetry

top-view

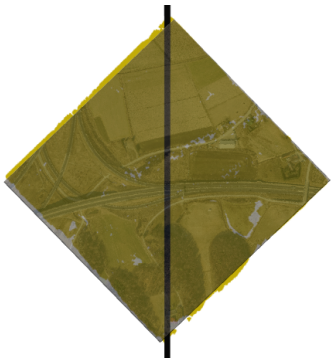
3/4-view



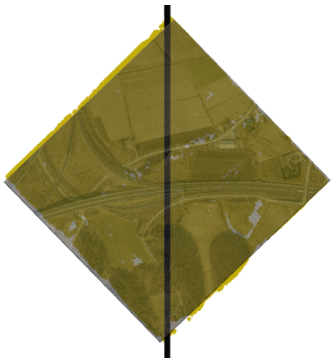
data1

Preprocessed data

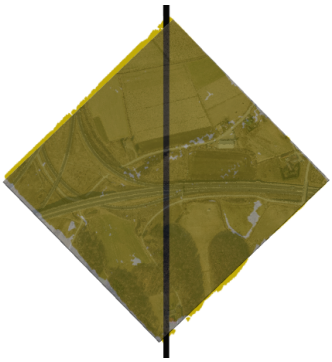
Original data



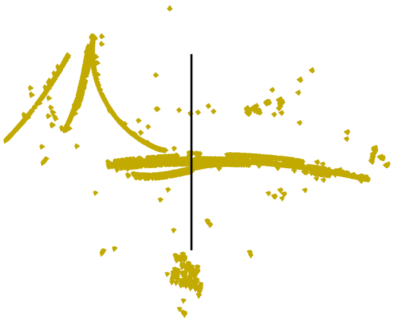
Height-based Laplace



Color-based Laplace

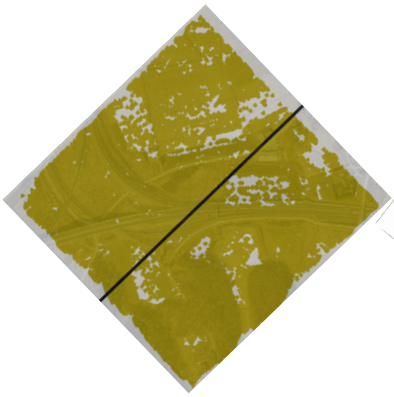


Intensity-based Laplace

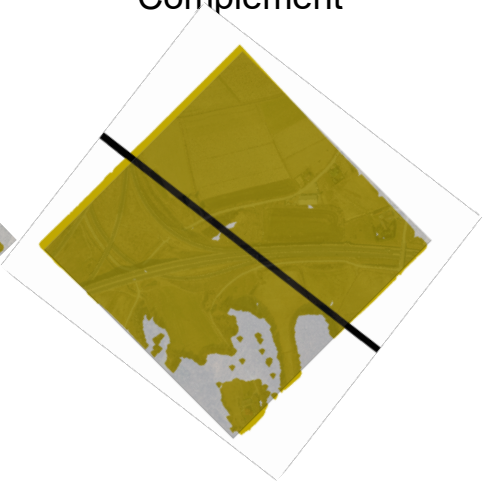


Majority classification

Majority

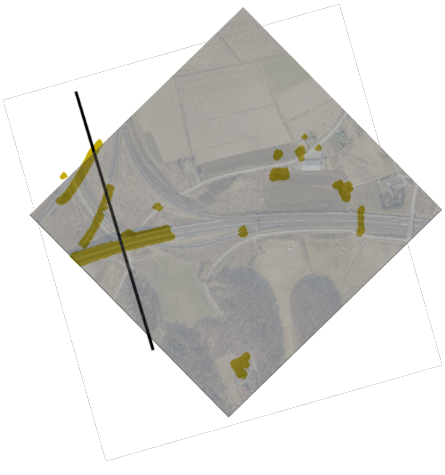


Complement

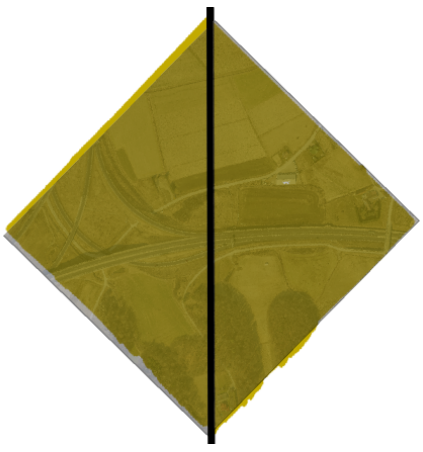


Alternative classification

Man-made set



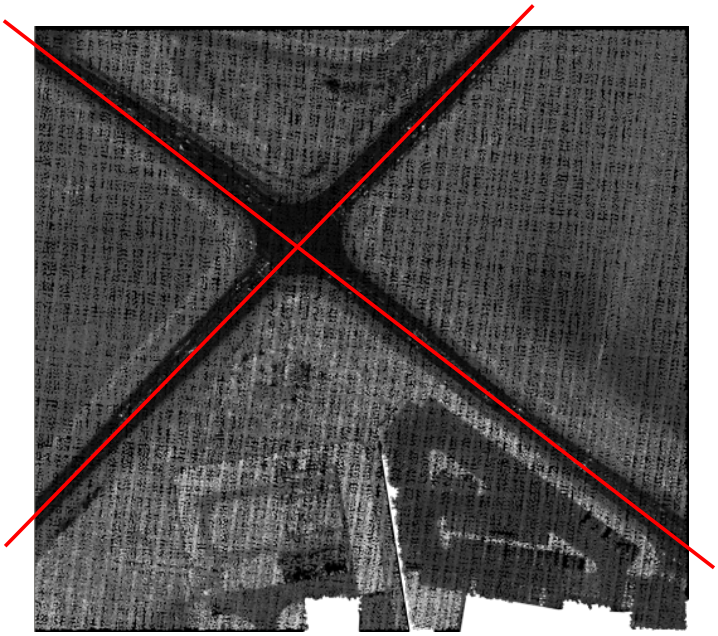
Natural set



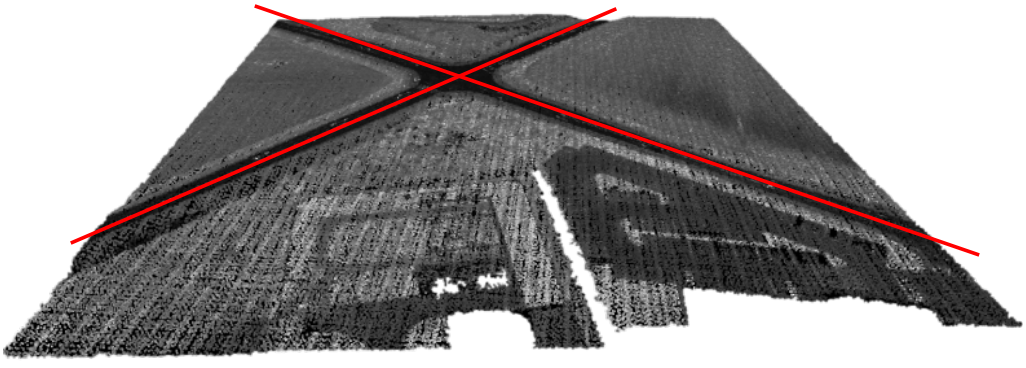
data2

Original data visualized in plas.io with expected symmetry

top-view



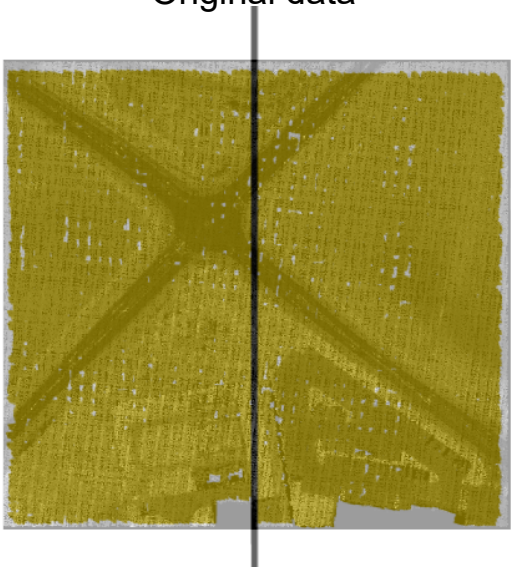
3/4-view



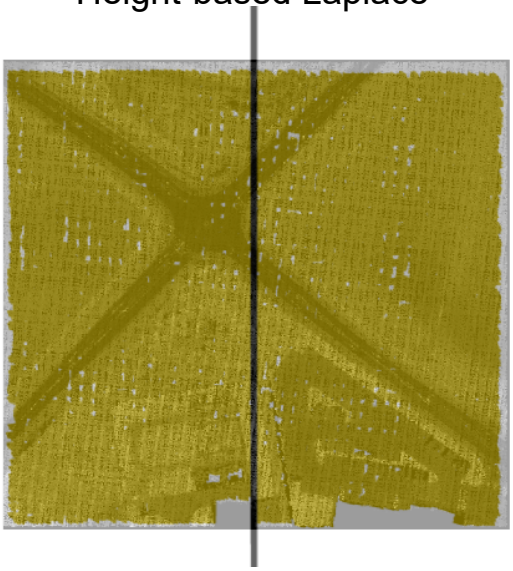
data2

Preprocessed data

Original data



Height-based Laplace

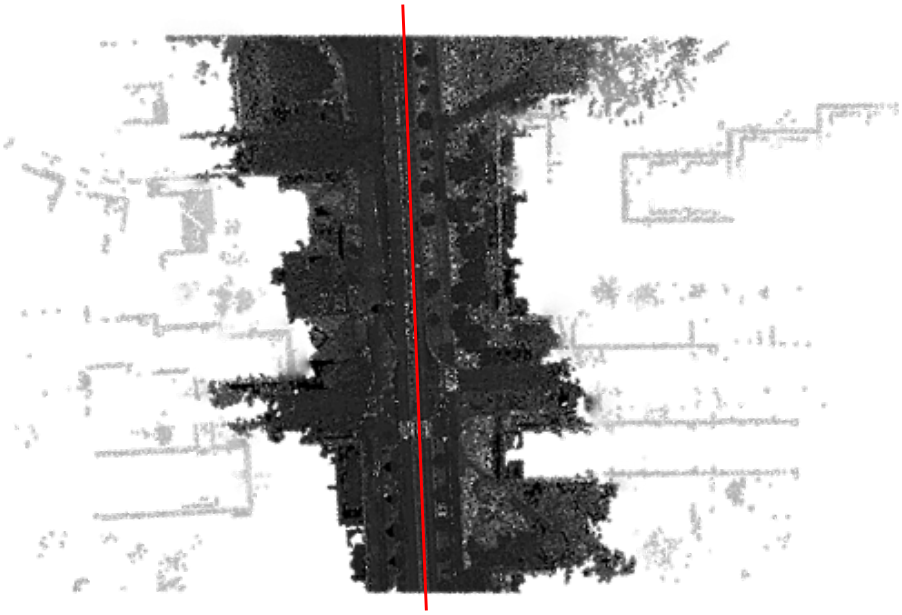


Intensity-based Laplace

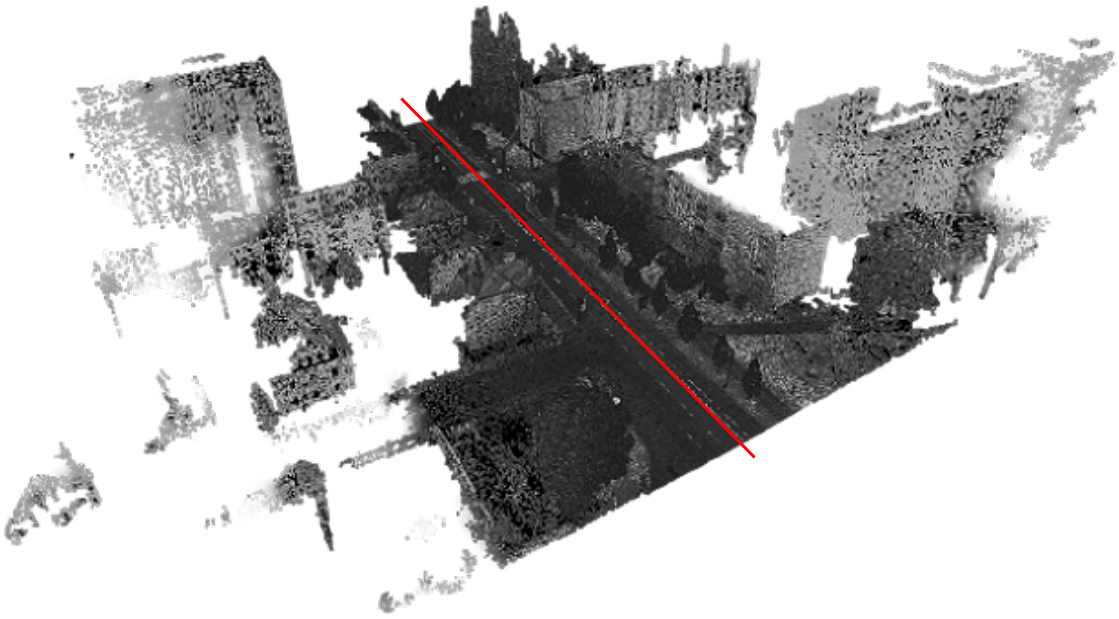
data3

Original data visualized in plas.io with expected symmetry

top-view



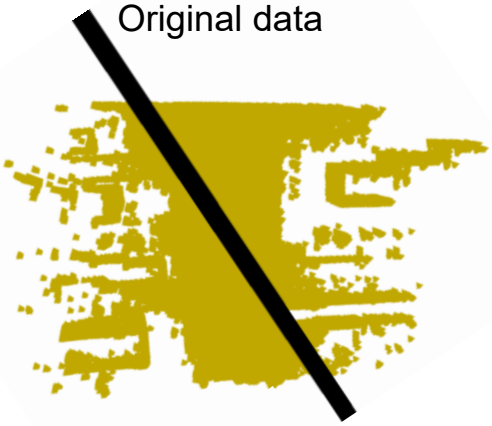
3/4-view



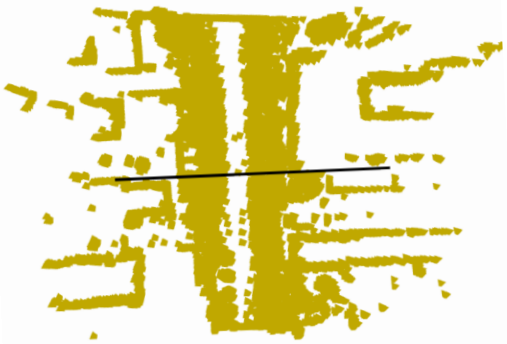
data3

Preprocessed data

Original data



Height-based Laplace



Color-based Laplace

Intensity-based Laplace

Majority classification

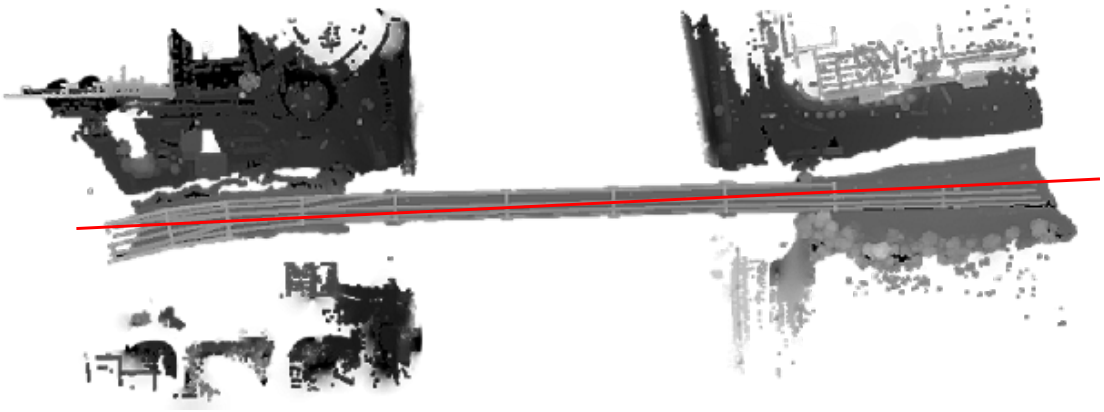
Majority

Complement

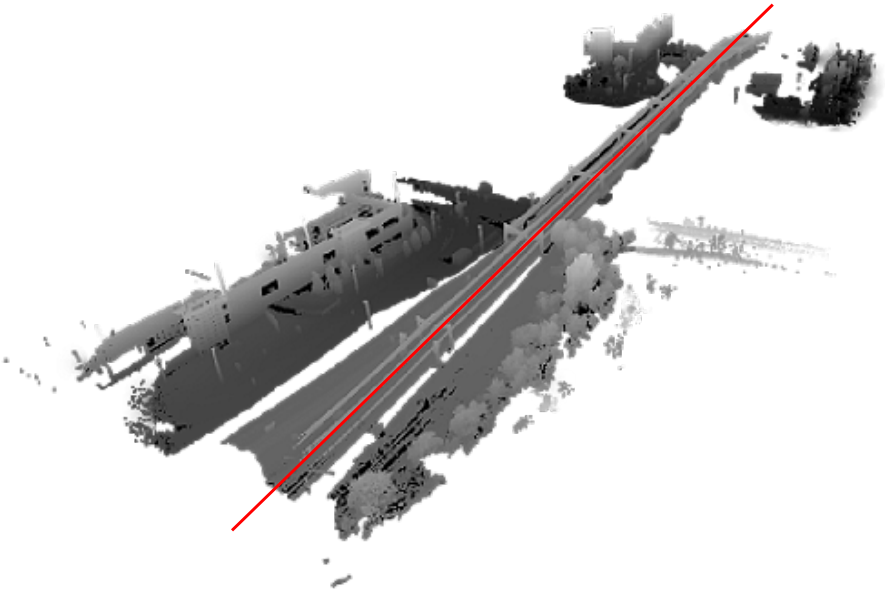
data4

Original data visualized in plas.io with expected symmetry

top-view



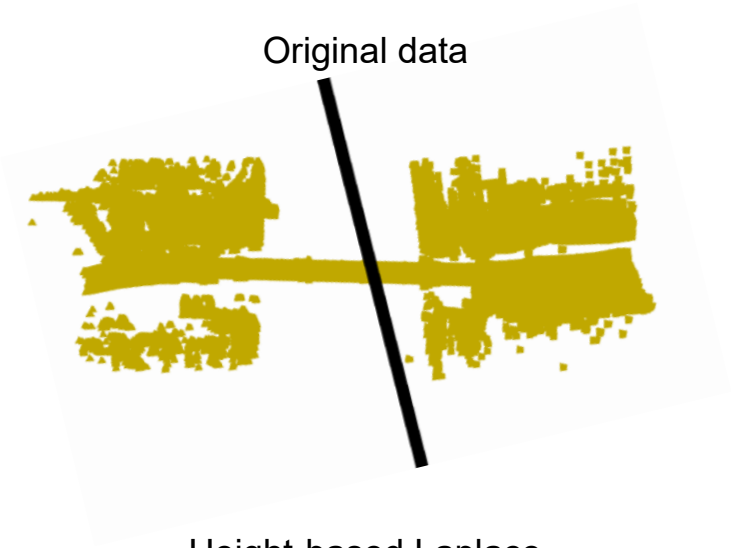
3/4-view



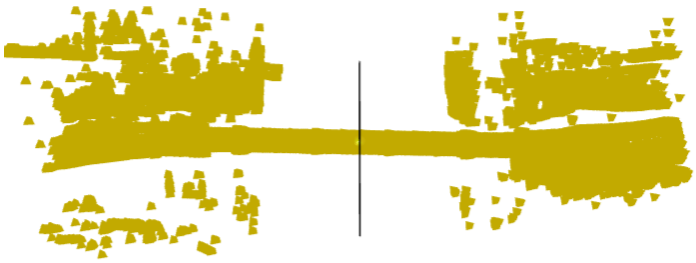
data4

Preprocessed data

Original data



Height-based Laplace



Intensity-based Laplace

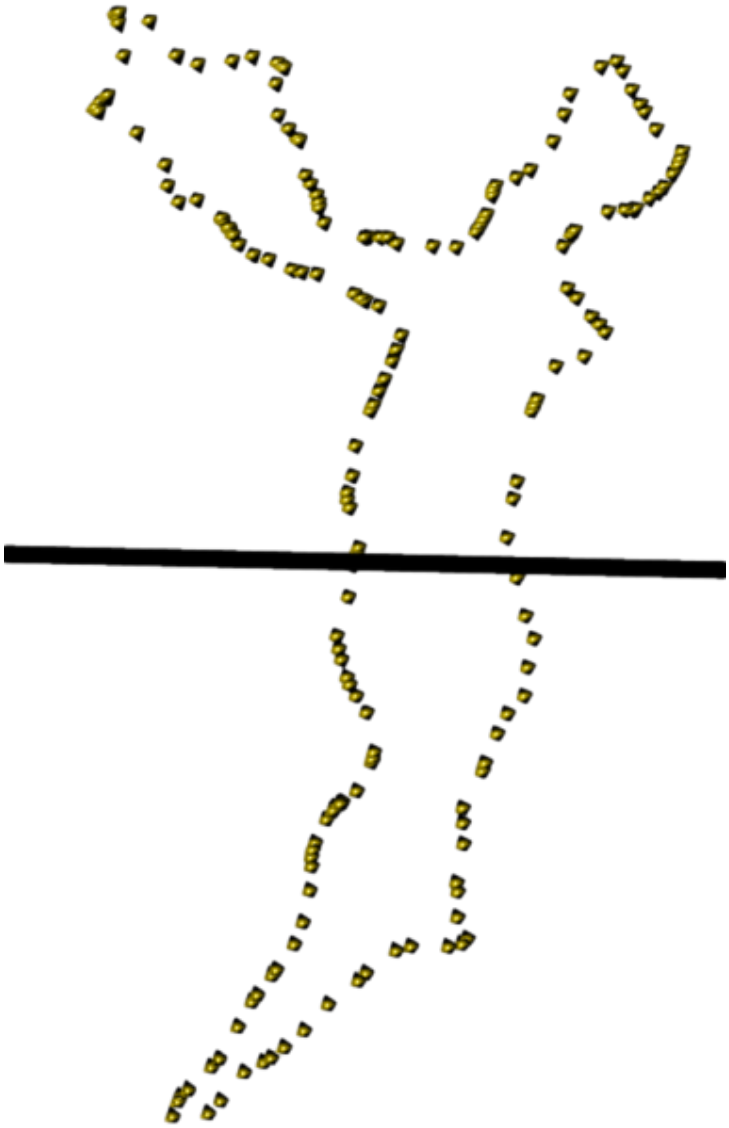
Appendix 2

geo_data3

Global symmetry

geo_data3

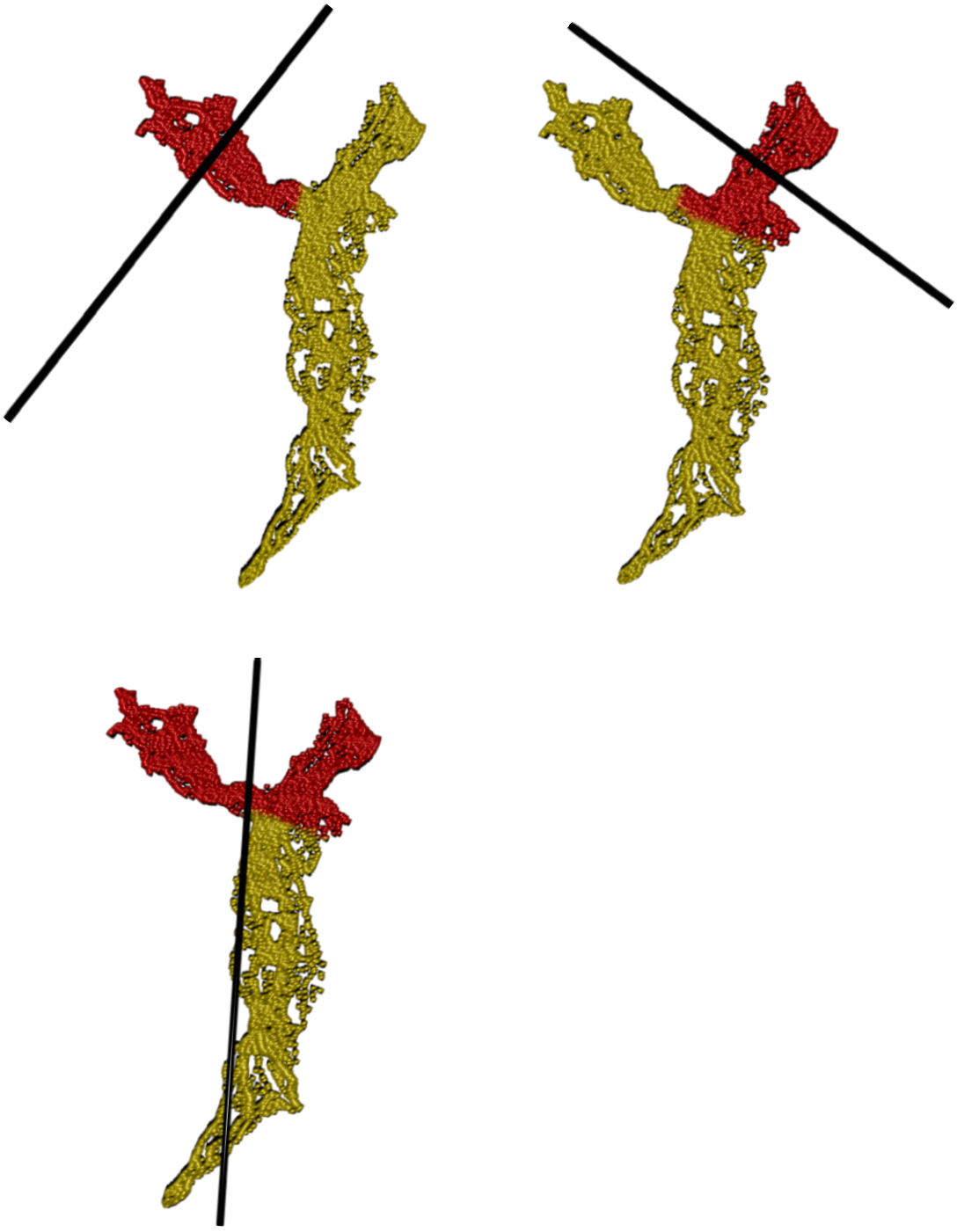
Global symmetry - outline



geo_data3

Local symmetry

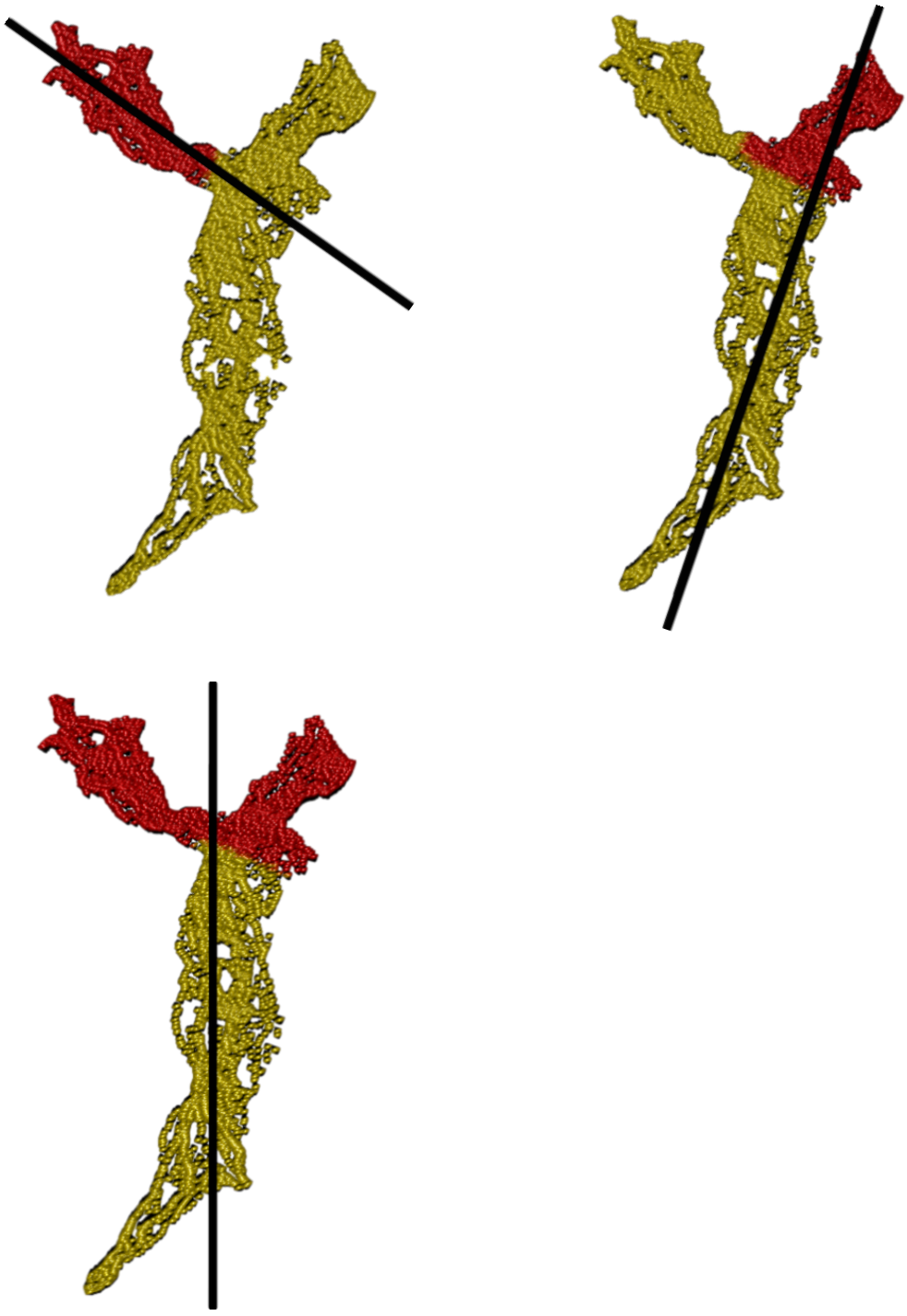
Manual weights (1, 0)



geo_data3

Local symmetry

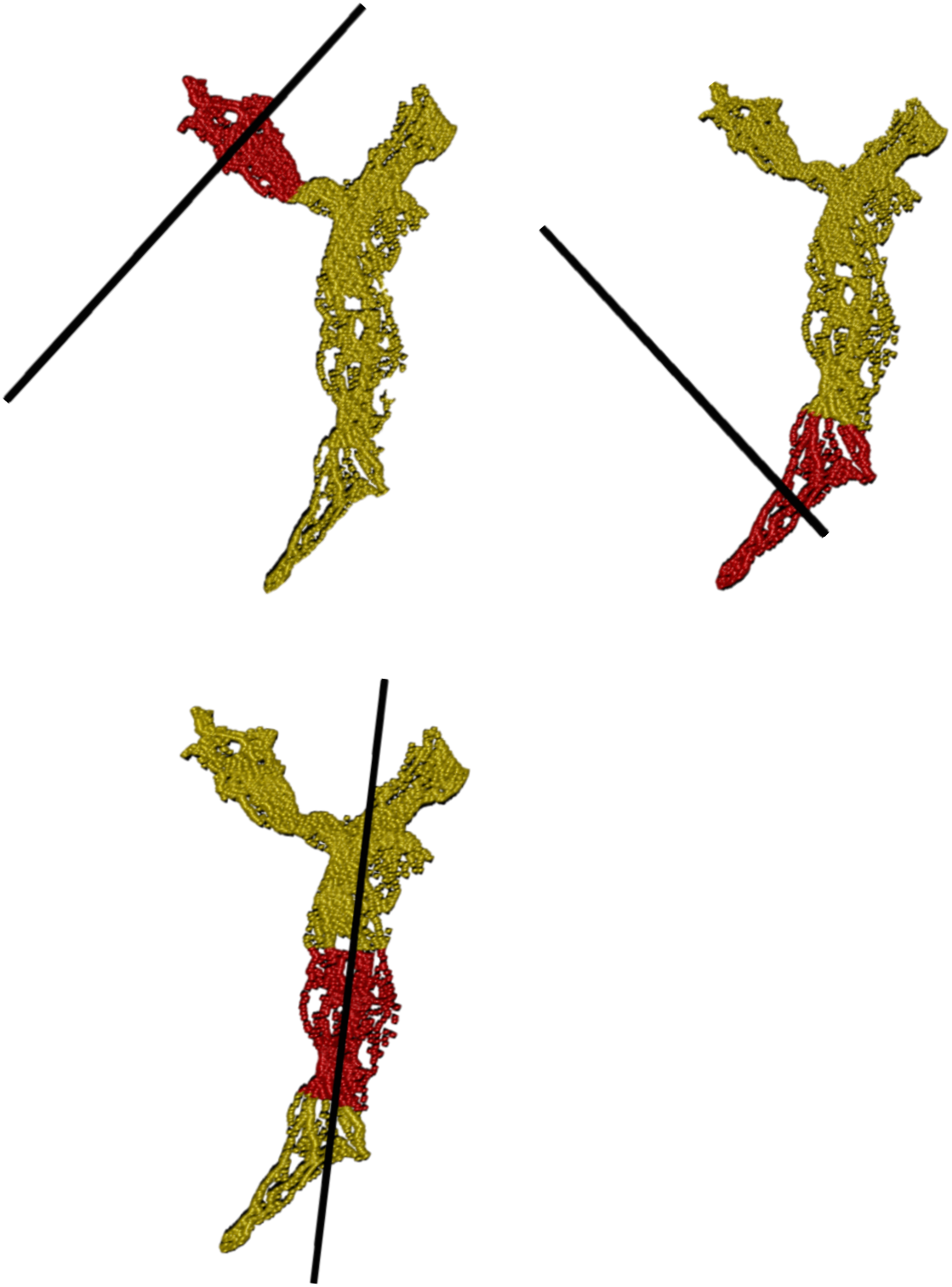
Manual weights (5, 1)



geo_data3

Local symmetry

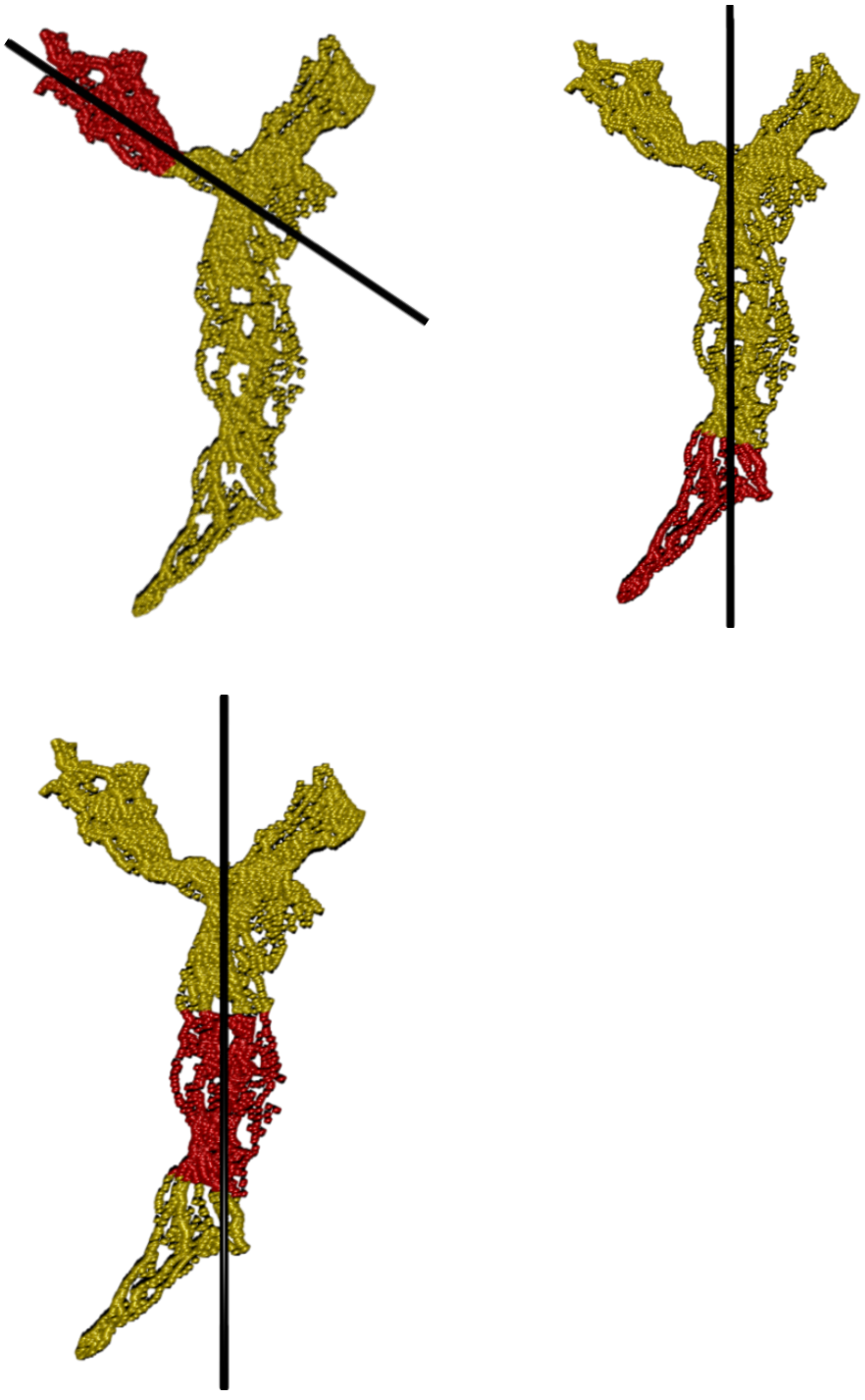
K-means (1, 0)



geo_data3

Local symmetry

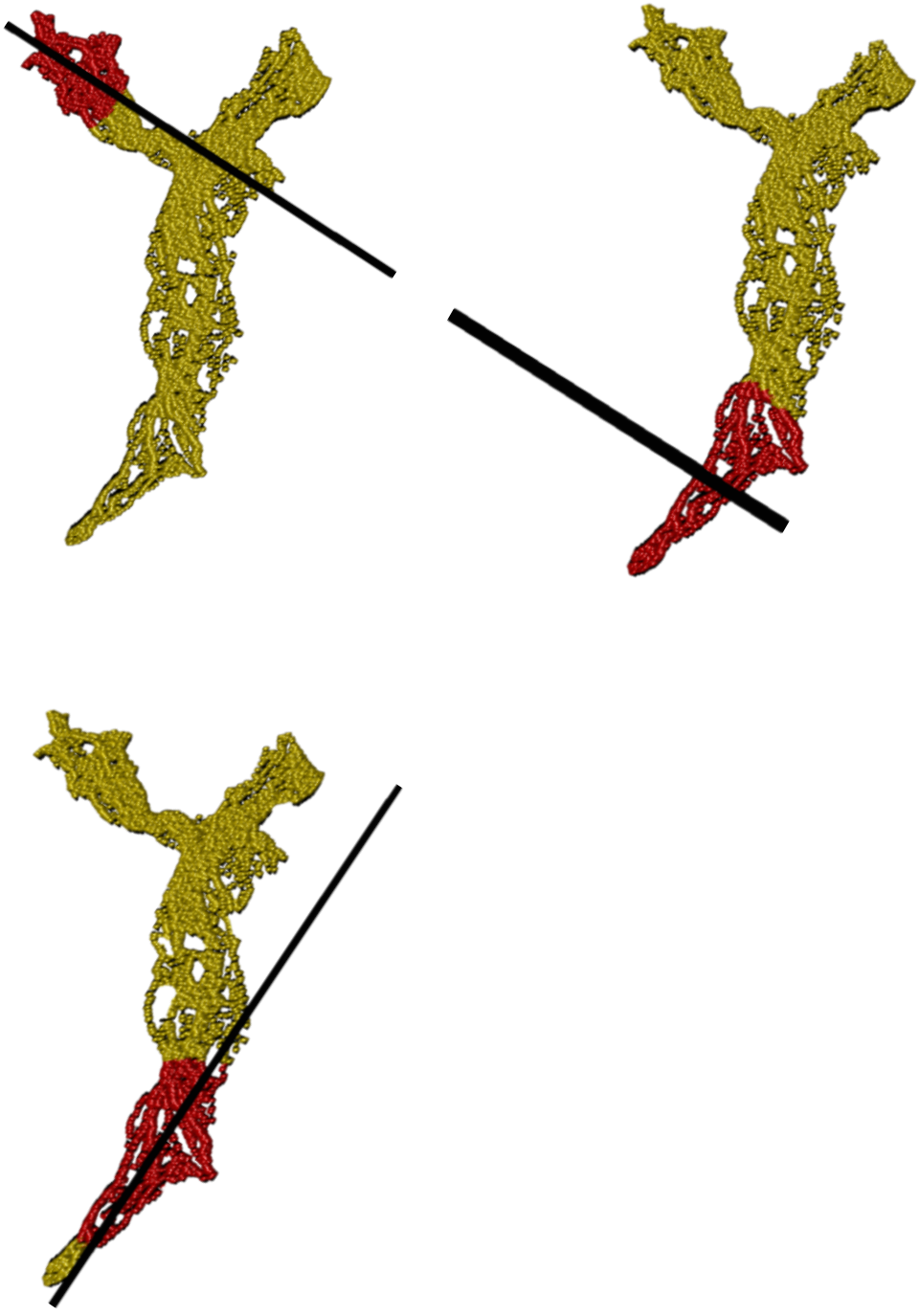
K-means (5, 1)



geo_data3

Local symmetry

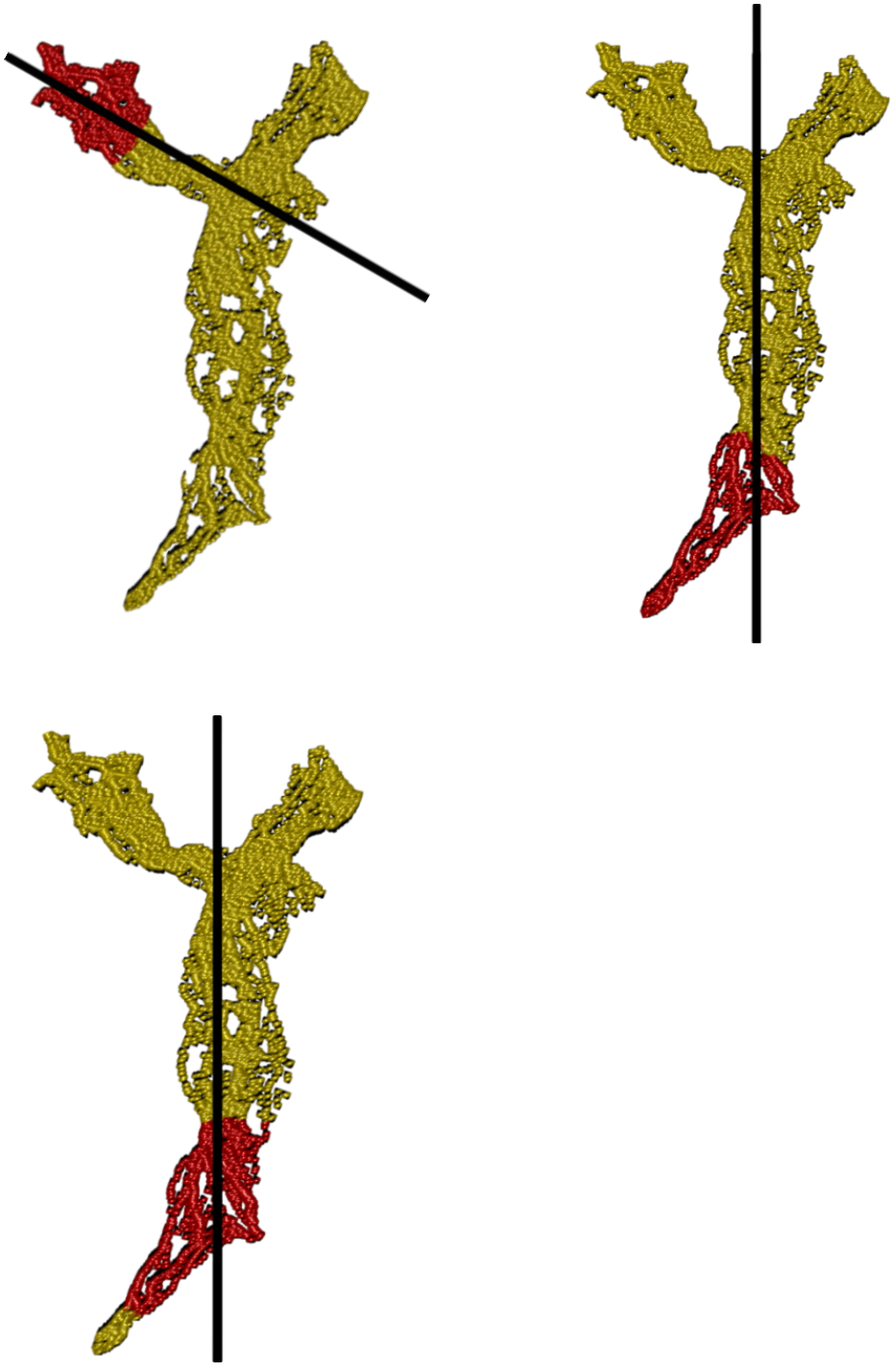
Bounding box (1, 0)



geo_data3

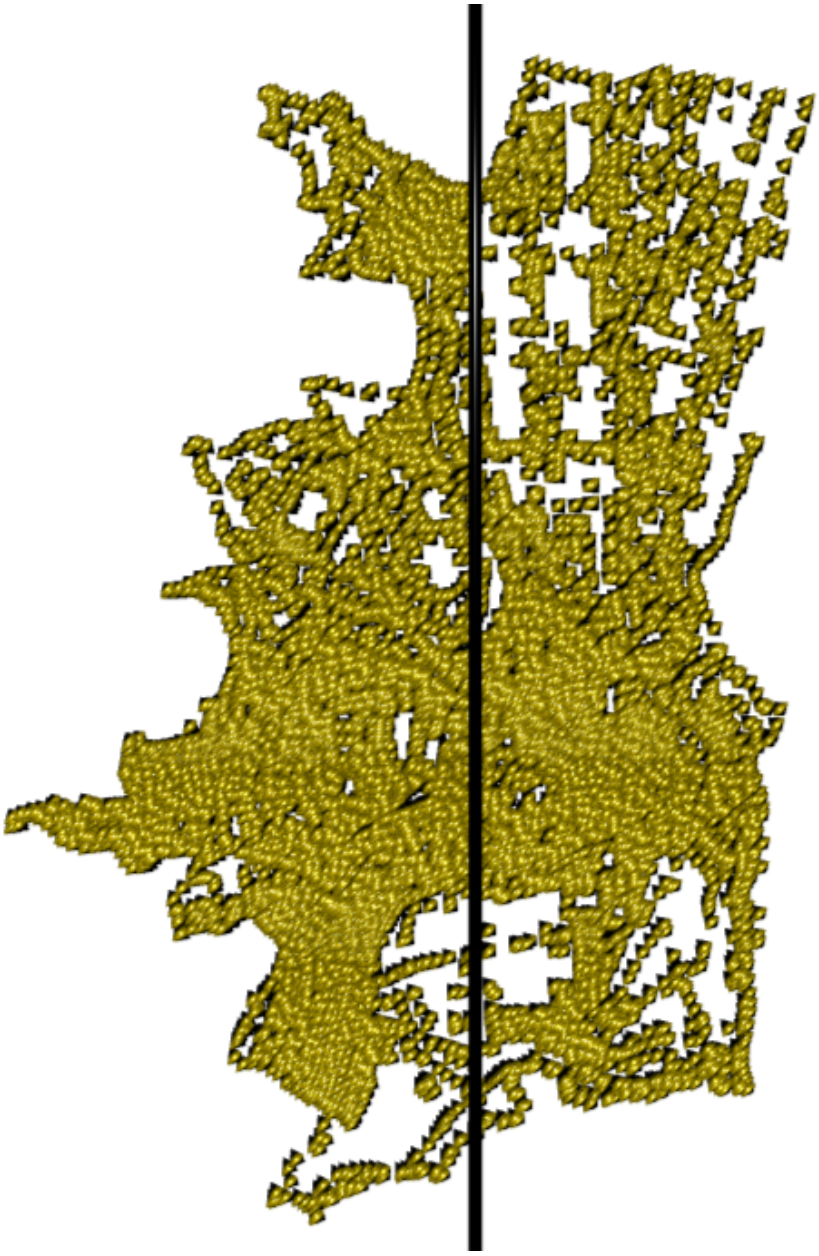
Local symmetry

Bounding box (5, 1)



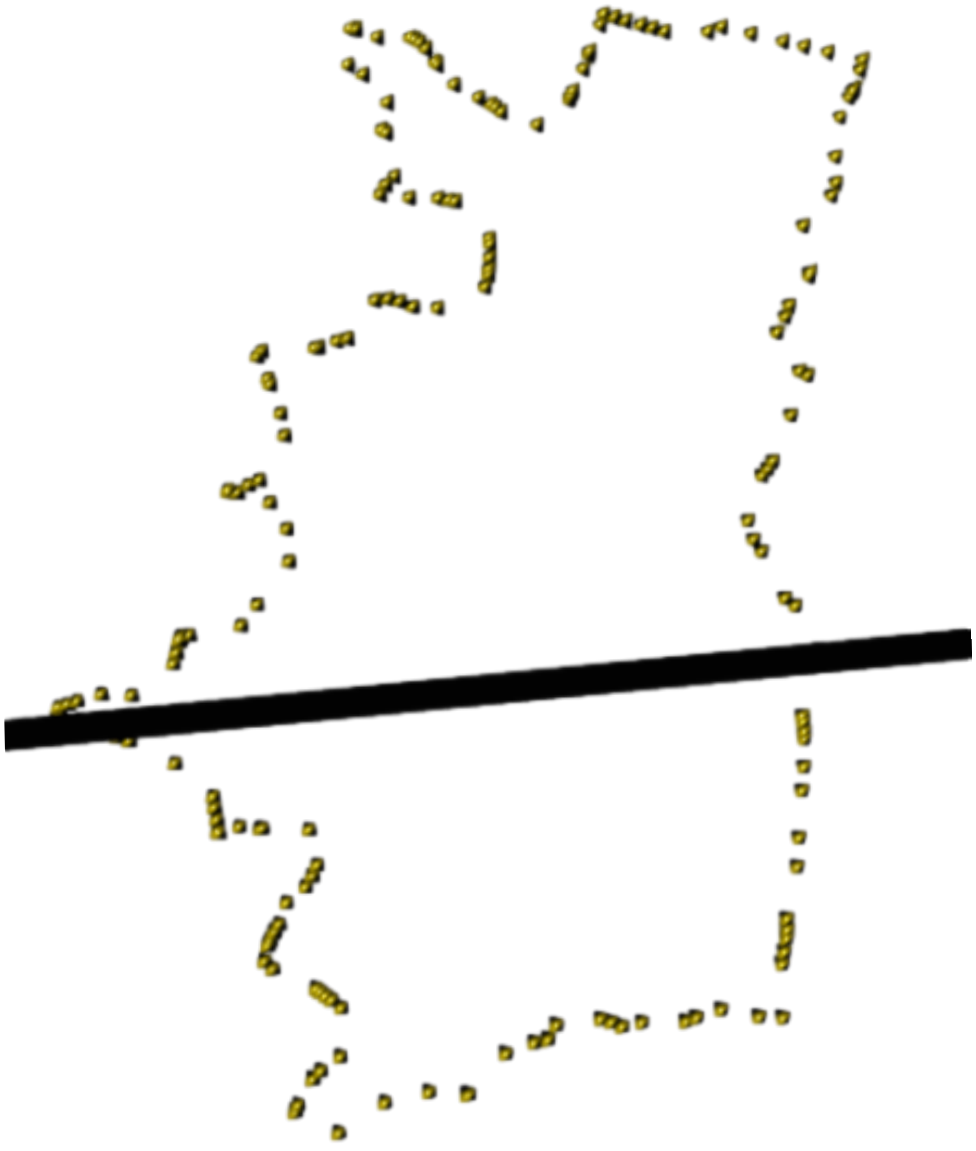
geo_data6

Global symmetry



geo_data6

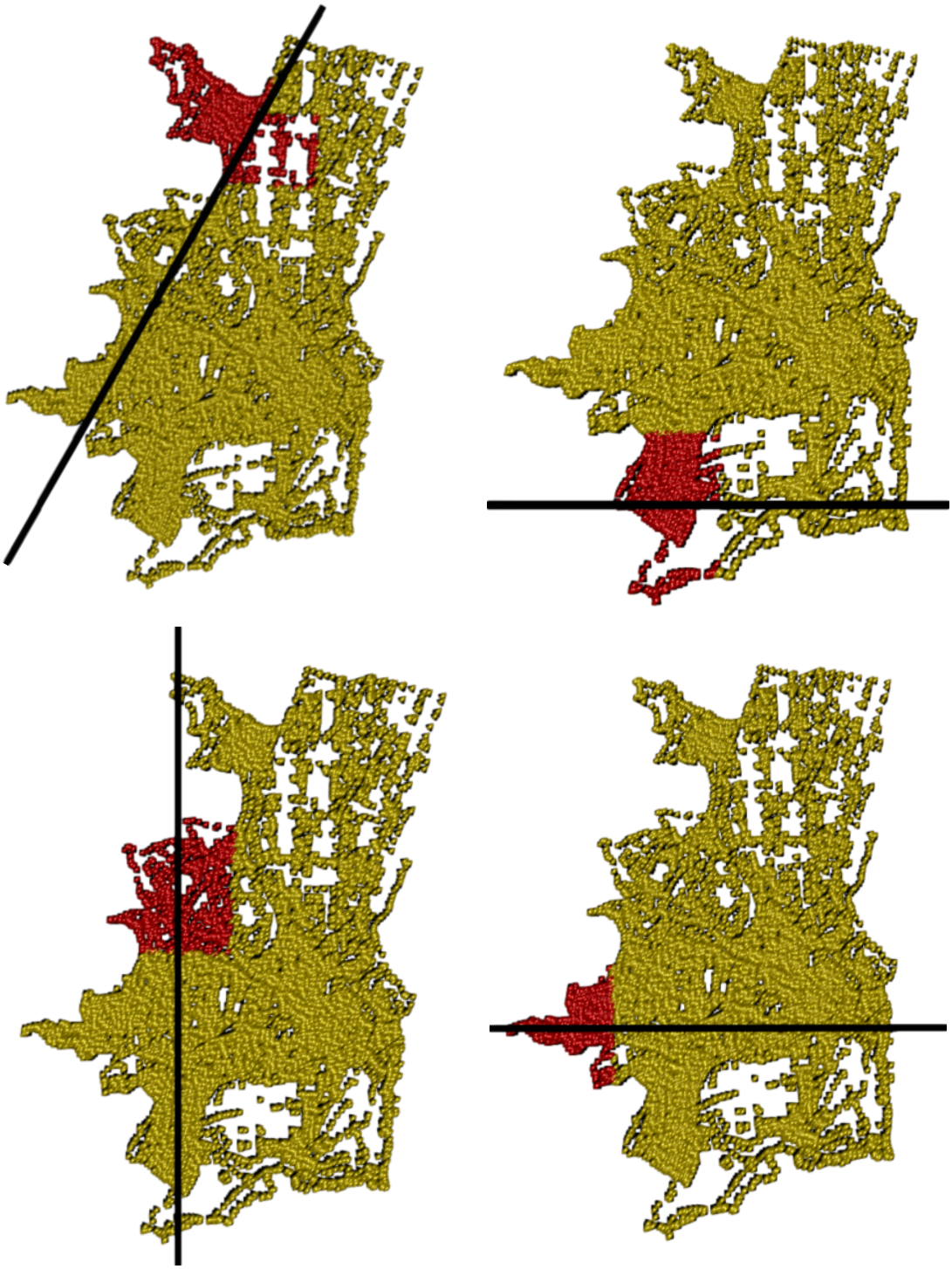
Global symmetry - outline



geo_data6

Local symmetry

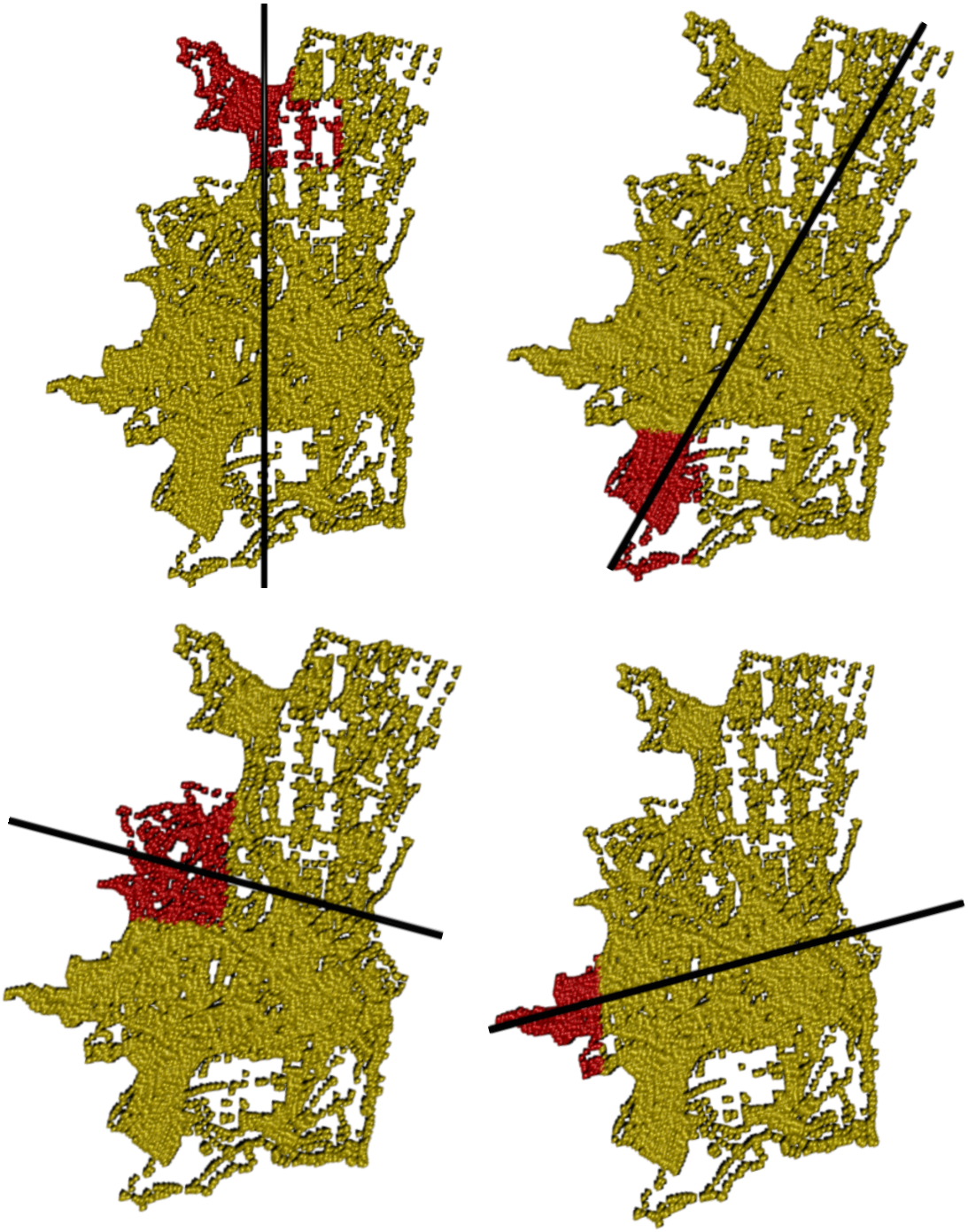
Manual weights (1, 0)



geo_data6

Local symmetry

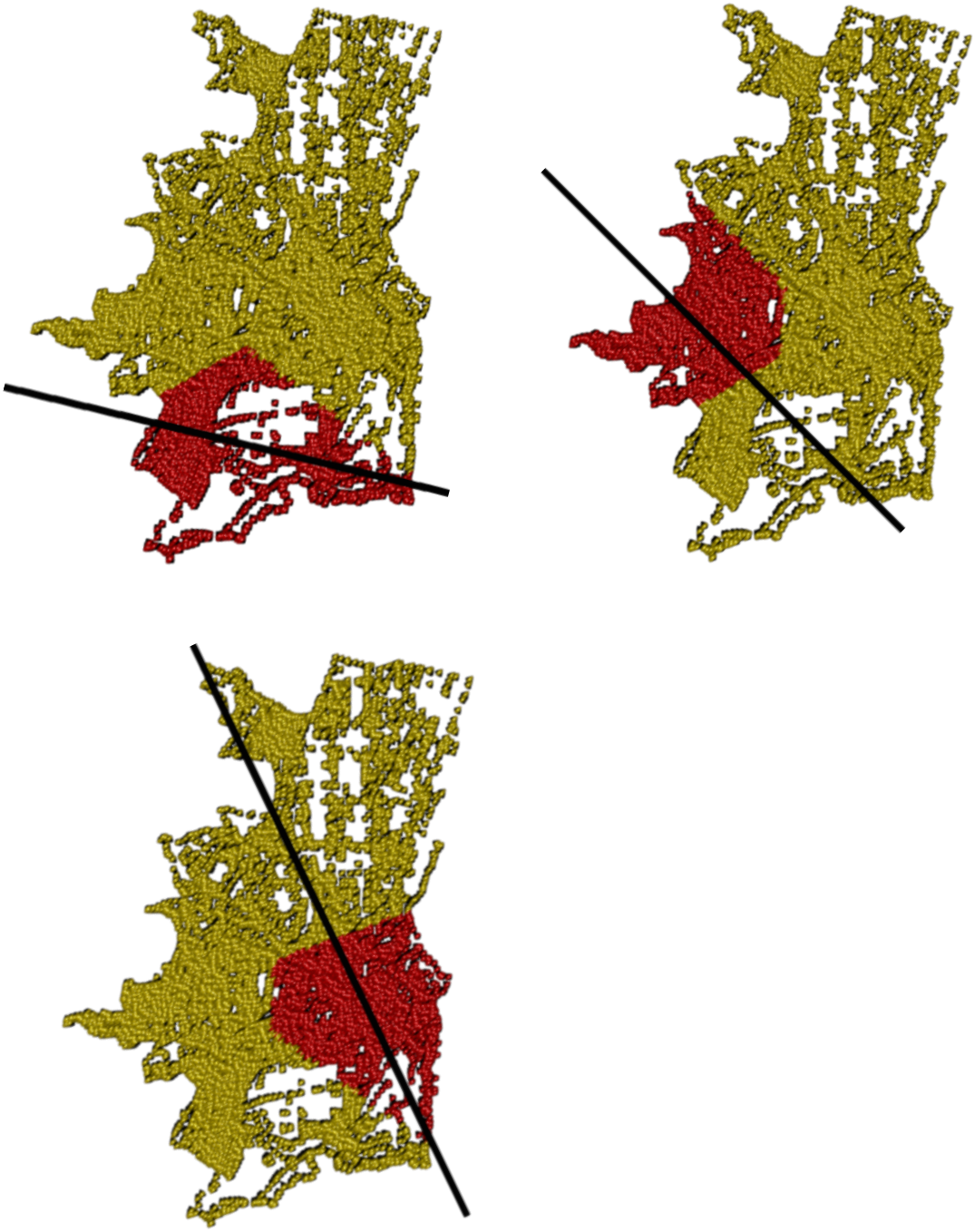
Manual weights (5, 1)



geo_data6

Local symmetry

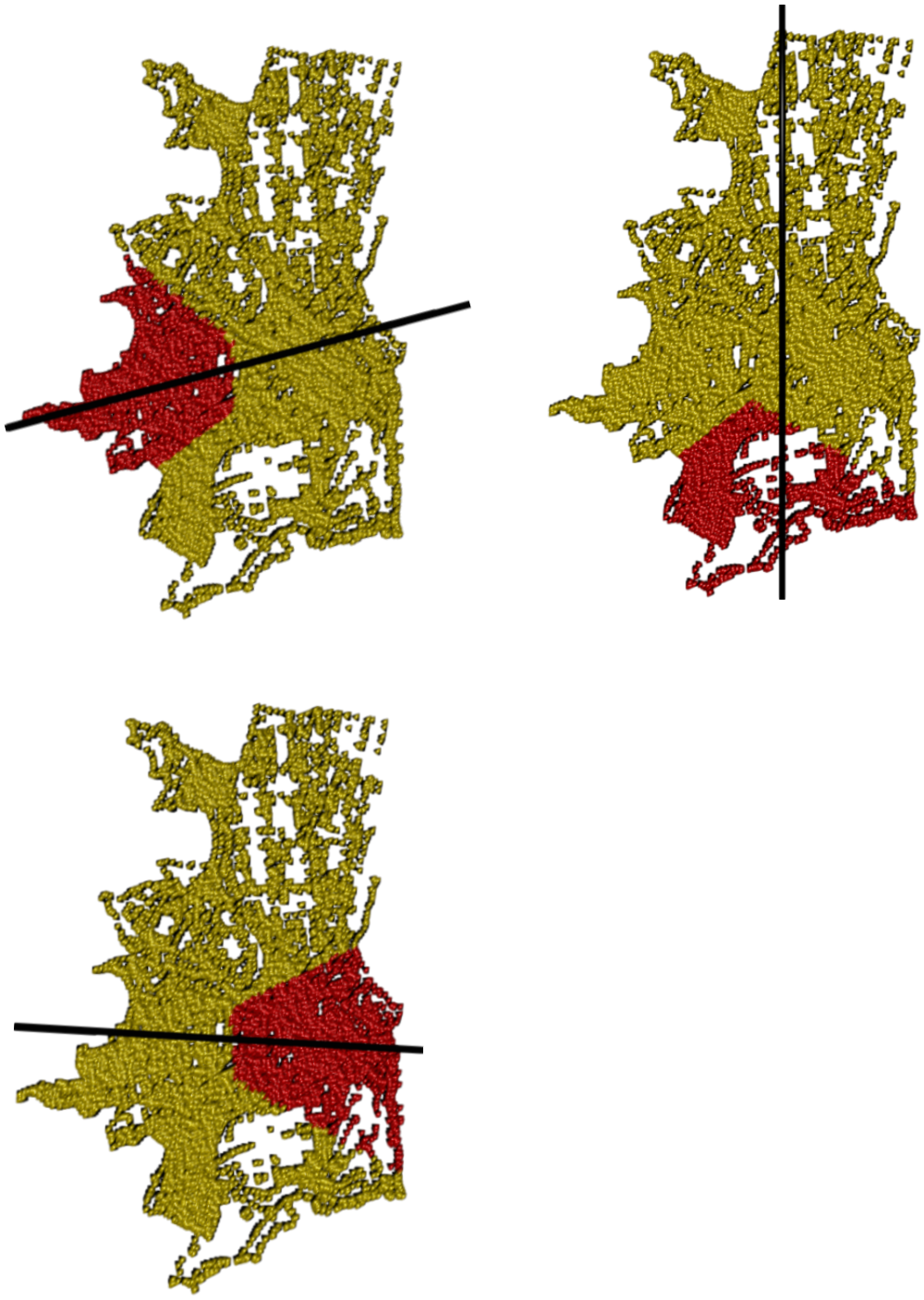
K-means (1, 0)



geo_data6

Local symmetry

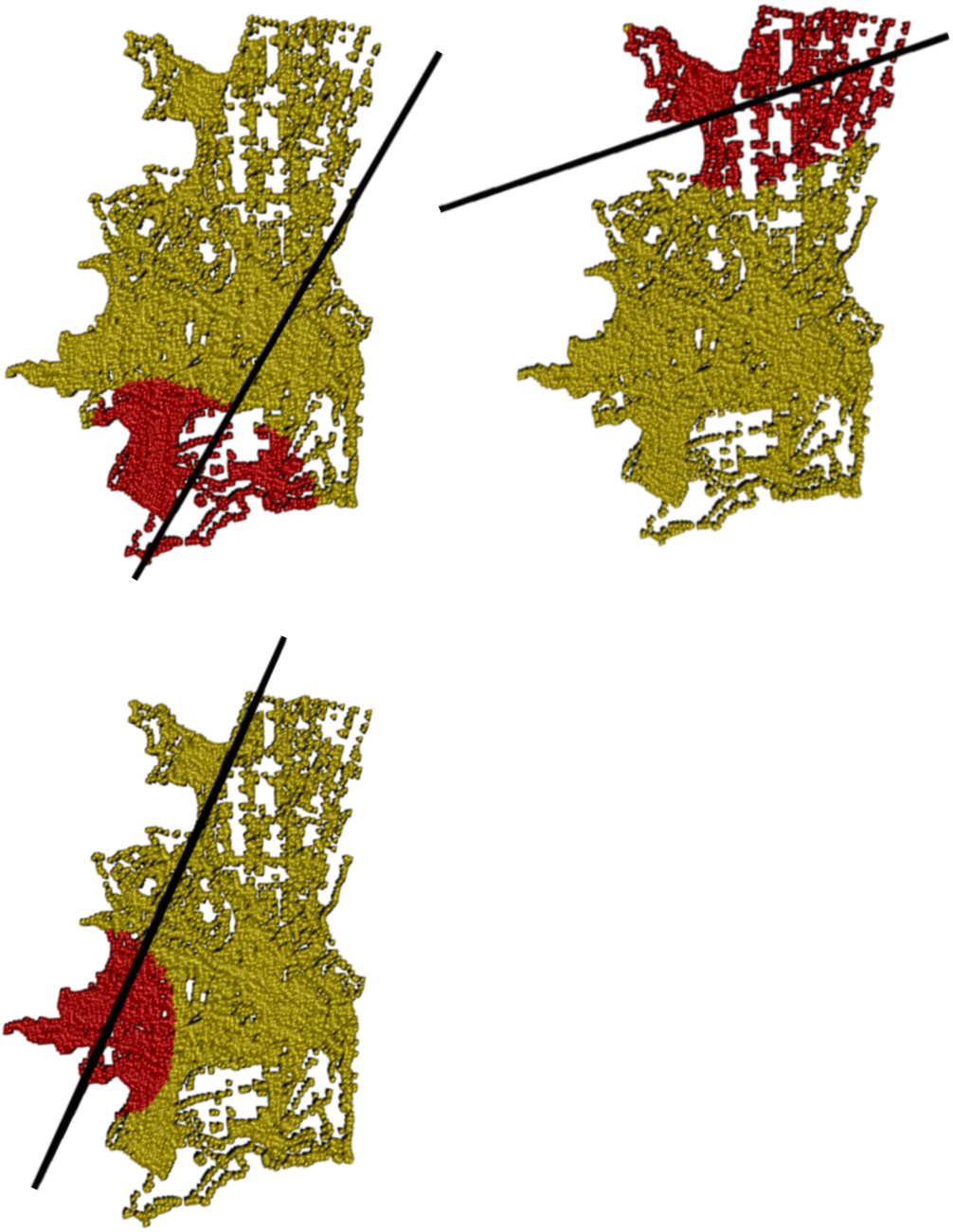
K-means (5, 1)



geo_data6

Local symmetry

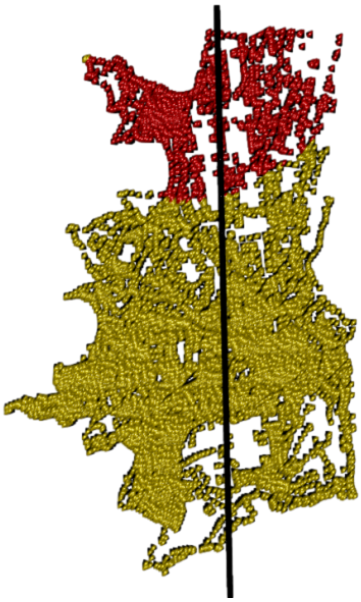
Bounding box (1, 0)

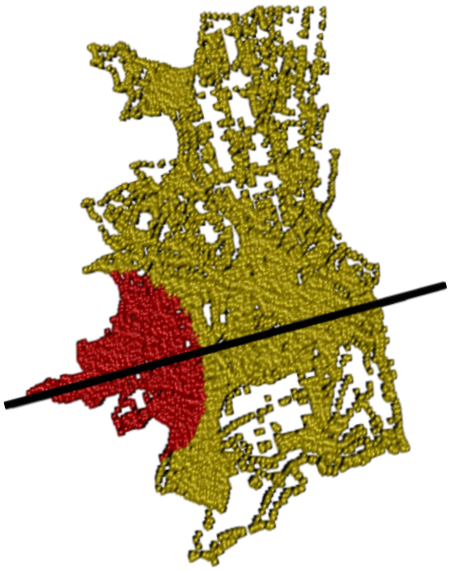


geo_data6

Local symmetry

Bounding box (5, 1)



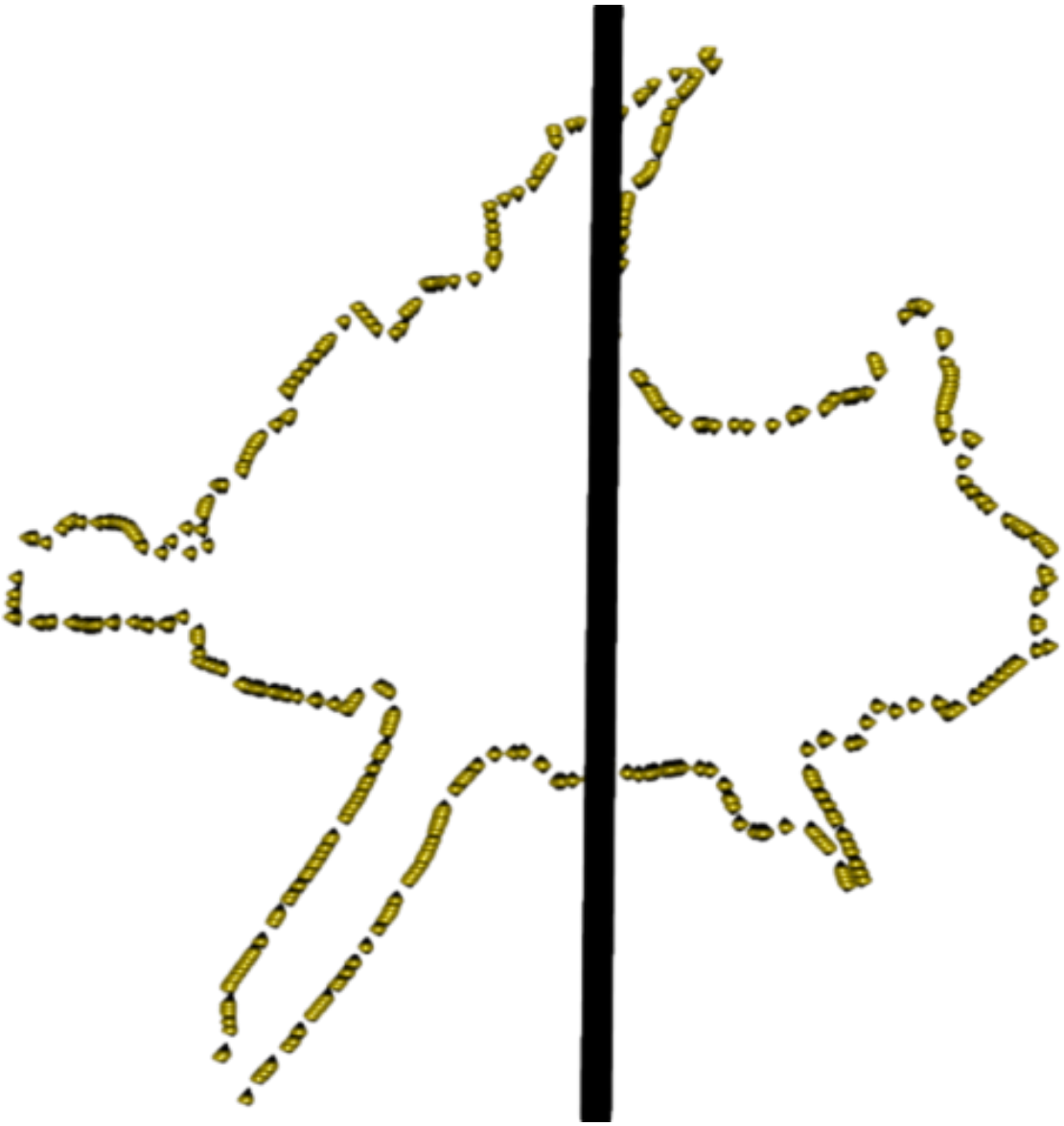


geo_data8

Global symmetry

geo_data8

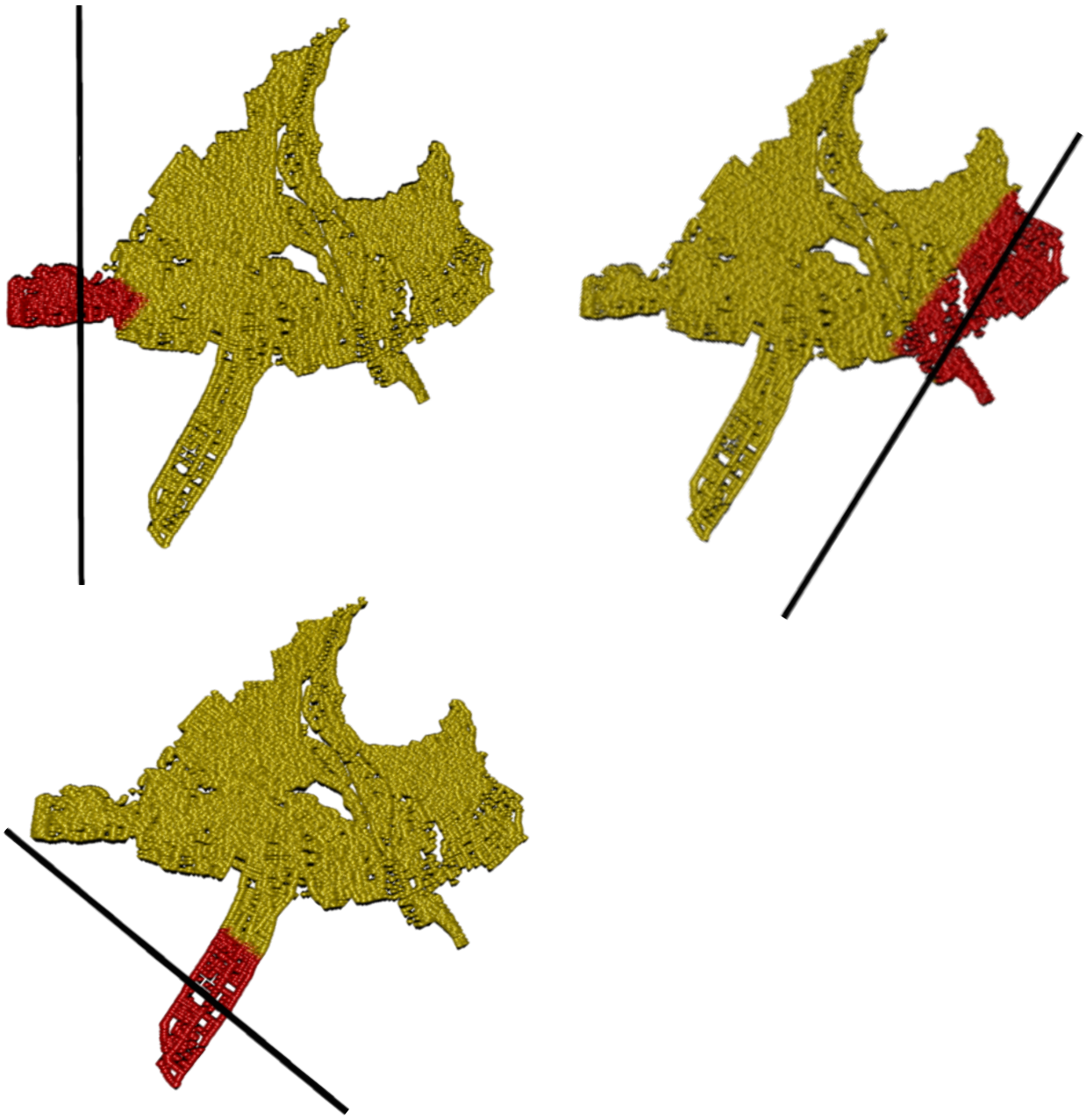
Global symmetry - outline



geo_data8

Local symmetry

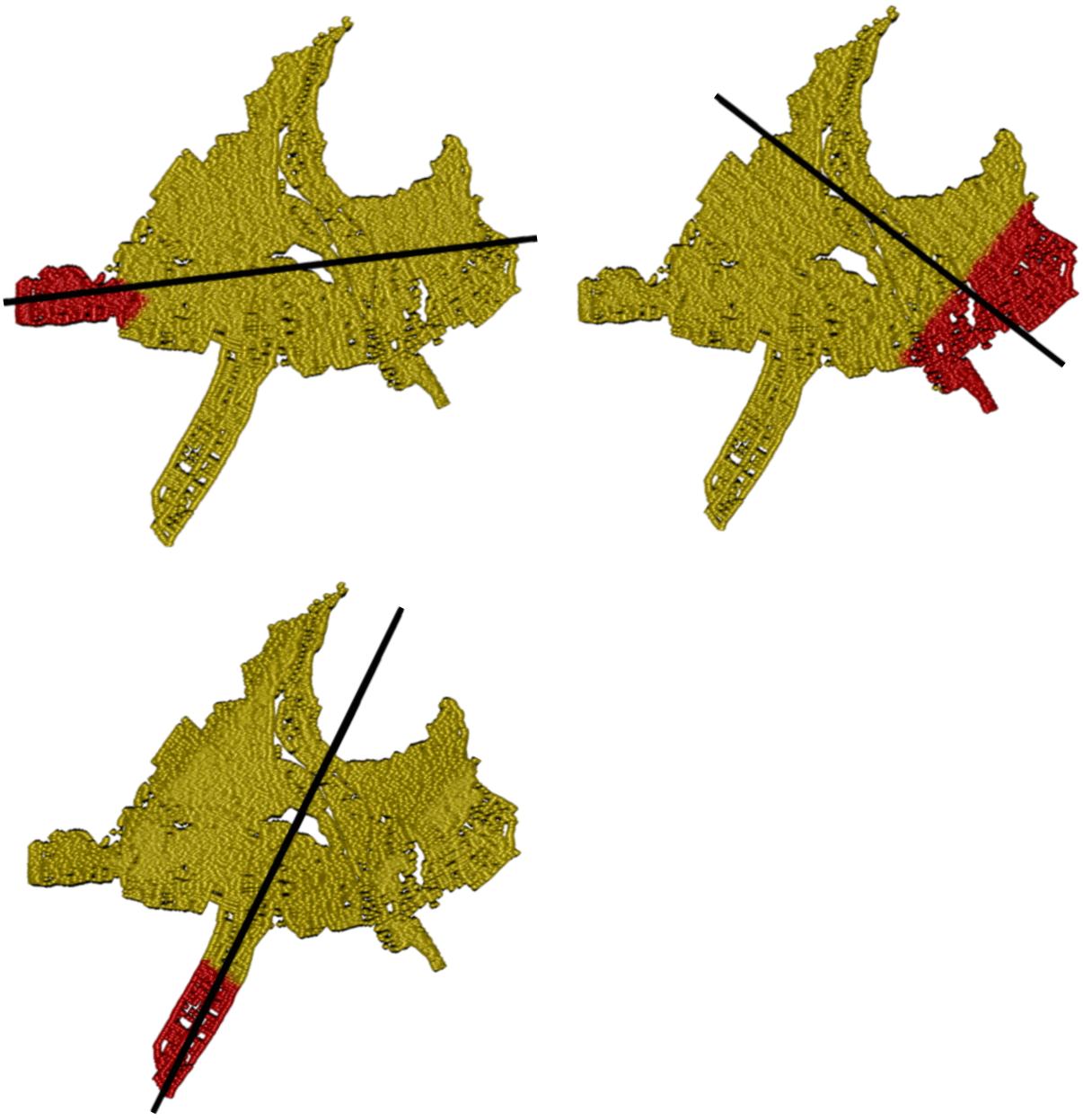
Manual weights (1, 0)



geo_data8

Local symmetry

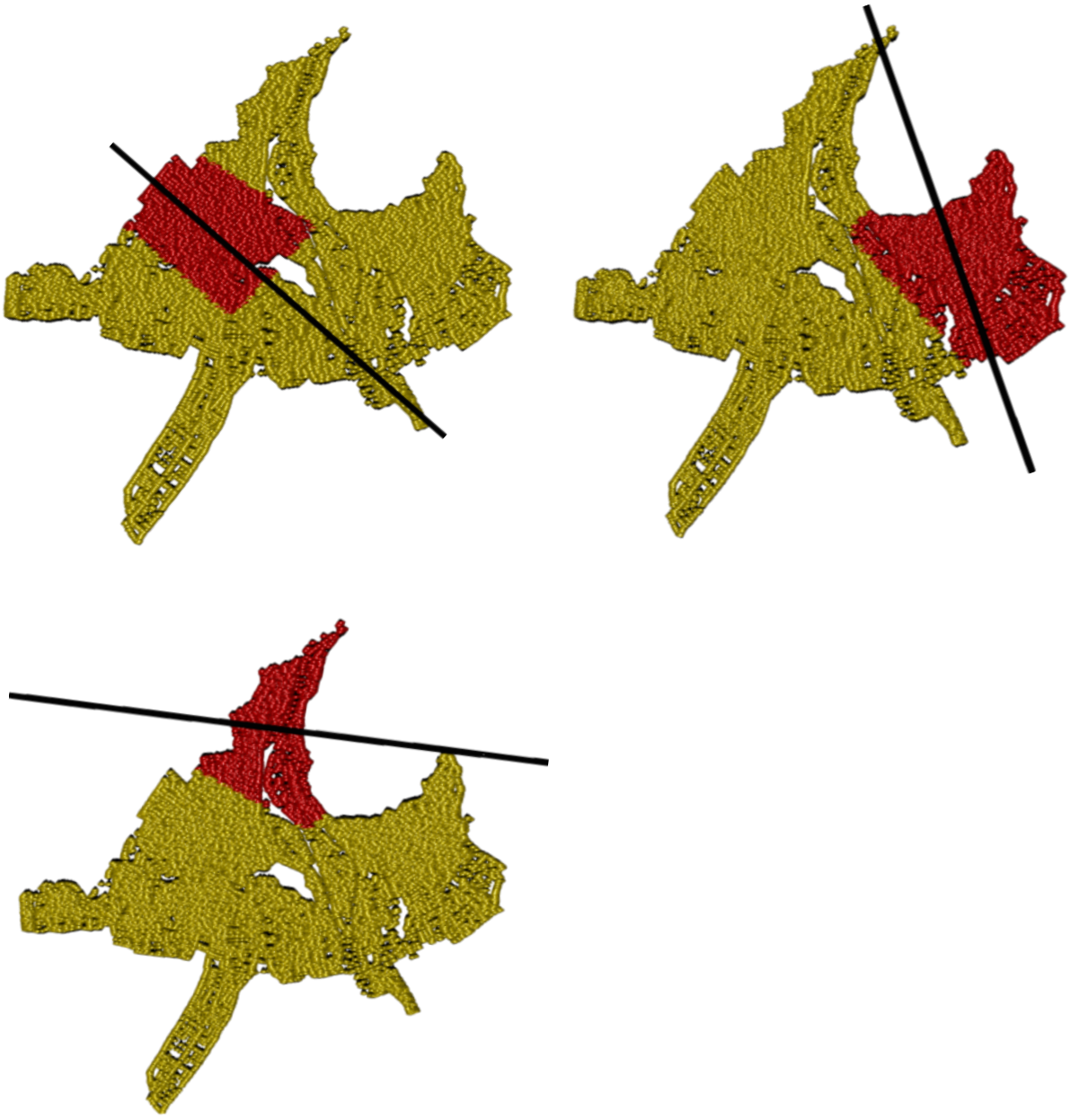
Manual weights (5, 1)



geo_data8

Local symmetry

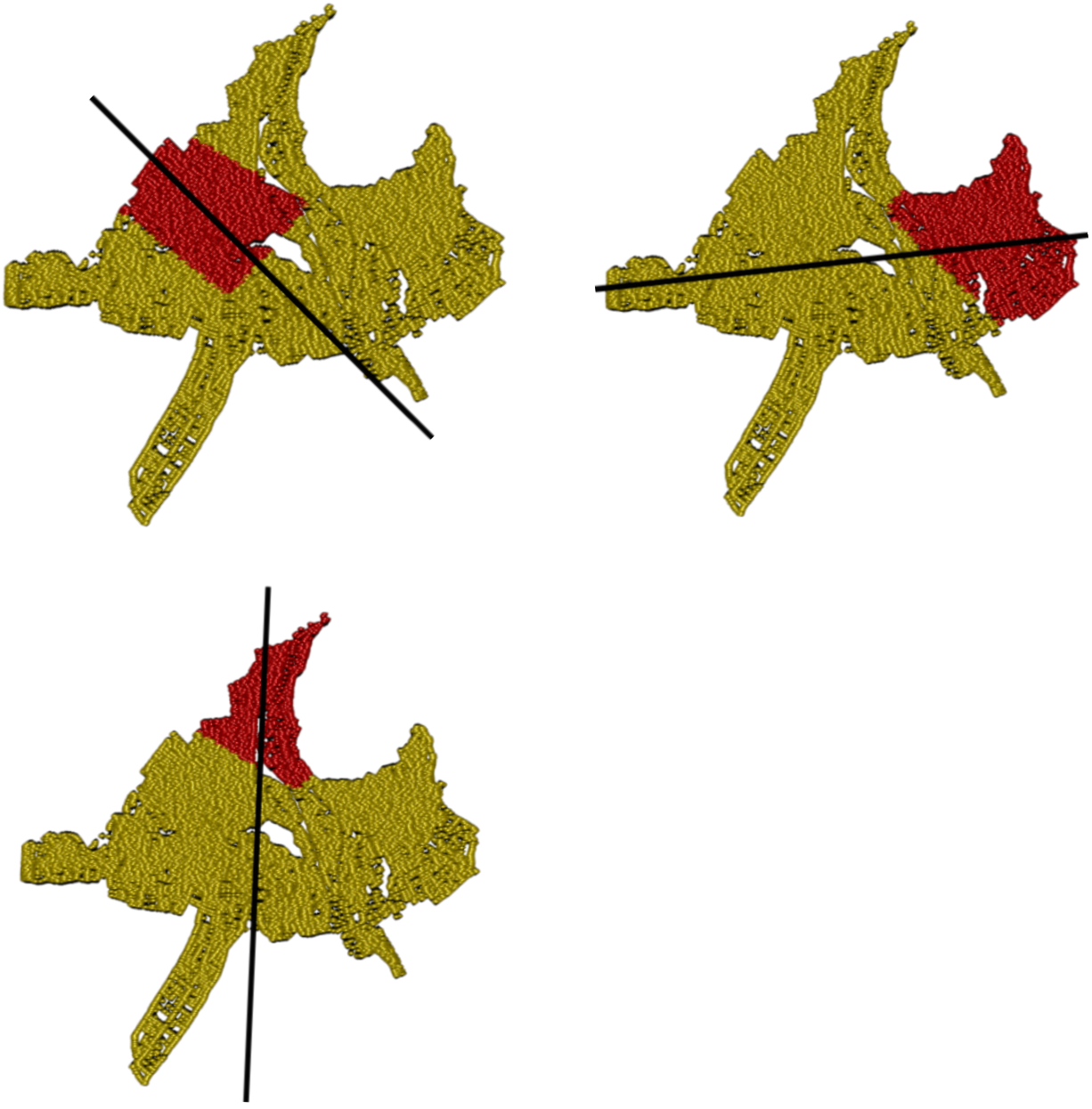
K-means (1, 0)



geo_data8

Local symmetry

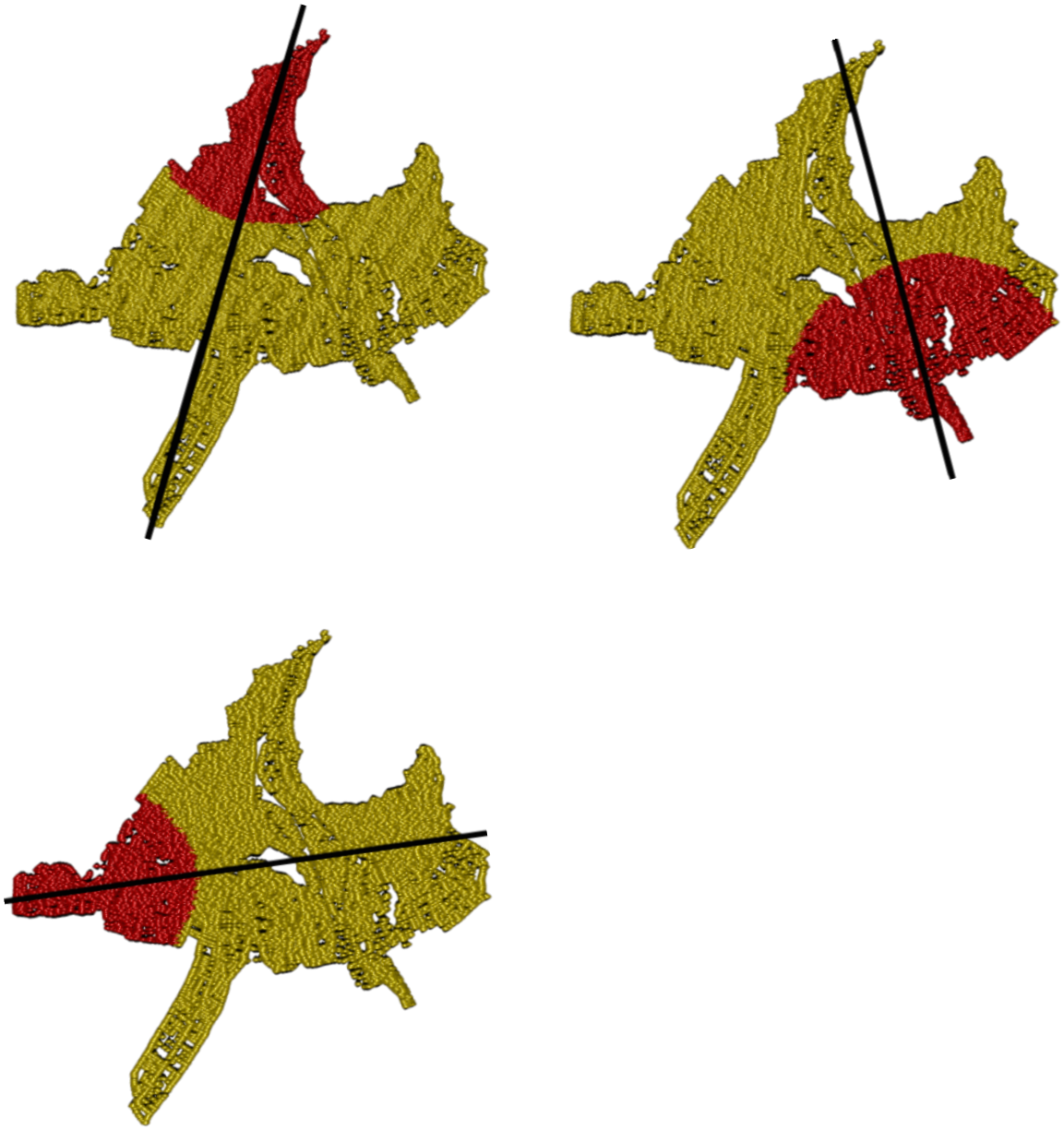
K-means (5, 1)



geo_data8

Local symmetry

Bounding box (1, 0)



geo_data8

Local symmetry

Bounding box (5, 1)