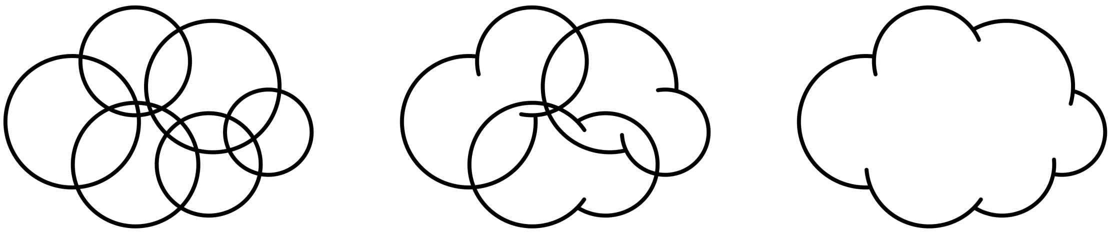
Point Clouds
Between Local and Global Approaches



Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

am Fachbereich Mathematik und Informatik
der Freien Universtität Berlin

vorgelegt von
Eric Zimmermann

Berlin 2023

Erstgutachter/in: Prof. Dr. Konrad Polthier
Zweitgutachter/in: Prof. Dr. Stefanie Hahmann

Tag der Disputation: 02.02.2024

"Before the mountains were brought forth,
or ever thou hadst formed the earth and the world,
even from everlasting to everlasting,
thou art God."

Psalm 90, 2 (KJV)

Table of Contents

Introduction

The representation of a surface, which is a 2-dimensional entity, in space has various solutions, when they are smooth as well as discrete. In all cases we ensure a 2-dimensional appearance in the vicinity of each point of the surface in question. Yet with the introduction of points as display primitives for the representation of surfaces, which we call point sets, cf. [LW85], the mere utilization of point scanning [JHPS21], and the advantages of processing point sets, see [GP07], comes with the price of missing connectivity among the points. This poses a major challenge to their treatment namely the definition of vicinity or in its generalization connectivity among points.

Point set processing led to a large variety of research always dealing with the lack of connectivity, either practical approaches for point neighborhoods are used or they receive focus and investigation in the determination supporting certain tasks done on the surface. Often such approaches position themselves at the two extremes. On the one hand there are local approaches in which every point in the point set receives its individual neighborhood whereas inter-relations of these neighborhoods are neglected (for instance in [HDD$^+$92]). On the other hand global approaches process the point set as a whole (for example [AB98]), sometimes setting a local focus again, e.g. in [ABCO$^+$03].

The goal of this thesis is to find a structure to represent point sets, i.e. using a set of subsets[1] of points, such that they relate in such a way—recall the inter-relations of point neighborhoods mentioned before—, that this structure carries the basic idea of an atlas, with its subsets and charts describing a surface or manifold more generally. The motivation to this goal is at least threefold: First, such a structure would impose some kind of atlas to point sets representing a surface, giving some notion of connectivity. Second, it positions itself between local and global approaches for point sets allowing to be either one of them and providing access to the incorporation of neighborhood inter-relations. Third, it allows a treatment of point sets representing surfaces with detailed microscopic structures like tree bark or skin in which the deviations of point positions are important and not the result of undesired influences like noise caused by scanning processes, see [GP07].

Structure and Main Contributions

We are going to investigate possible descriptions for set systems applied to points and thus called point clouds which then could be perceived similar to atlases for manifolds. These investigations are partitioned in a macroscopic and microscopic way. In the first we discuss two—under certain conditions—dual approaches and

[1]Later on we call these subsets point clouds.

thus focus on the inter-relation of the clouds, i.e. subsets of the points, and the properties we can derive when working with the points. In the latter we focus on a cloud itself researching inter-point relations and how those could influence the choice and treatment of a cloud.

We have created almost all implementations and resulting figures in JavaView [PRYZ], with implementation exceptions mentioned throughout the work.

Cloud Surfaces To find a suitable set system applied to a set of points and have these sets relate in a manifold manner, we make use of the vast theory of simplicial complexes and surfaces created of those. This enables us to define a set system on objects[2] (cloud set) s.t. the underlying structure (an inherited cloud complex) is preferably an (abstract) simplicial surface, i.e. a so called cloud surface, cf. Section 1. Thus the pure knowledge on chosen subsets of points accompanied with an inter-relation—in our case the intersection of subsets determines whether clouds are neighbored or not—decides in a combinatorial manner whether our cloud set is connected in a manifold way. In consequence each cloud represents a discrete neighborhood for all the points it contains. After the description we investigate the generation of cloud surfaces. At first, we start in a synthetic manner (Section 2.1), i.e. starting with a simplicial surface, generating points and placing them in respective clouds so that the simplicial surface we started with already accounts for the underlying structural object. Second, we make use of coverings to generate clouds from points while at the same time preserving the desired topology of a 2-manifold, see Section 2.2. The research on these coverings led to a couple of topological questions which we will further discuss. Following the generation, we set our focus on an application scenario making use of the cloud surfaces namely cloud dynamics, which shall allow point movements in space w.r.t. the underlying structure (Section 2.3). Here the simplicial surface, derived from the cloud surface, acts as a skeleton whereas positional changes of surface points have an impact on the combinatorial clouds and thus on the points they contain.

Surface-like Structures A dual perspective on such a set system is to treat a subset of points (a cloud) as a 2-dimensional entity, thus acting dual to the approach of cloud surfaces, where a cloud was represented by a single vertex in the underlying cloud complex. Here we describe structures and conjectures so that they possibly behave like 2-manifolds, thus we call them surface-like in Section 3. Following the description we discuss a possible generation of a surface-like structure using a point segmentation approach based on *k-means* clustering (Section 4), which is published in [SZP20]. As such the dependency on the number of seed points and placement of those is crucial, which we are able to overcome introducing splitting and merging of clouds (or regions). Additionally, we propose a switch operation to show guaranteed convergence of the energy accompanied with the clustering approach. As an application we generate a simplified mesh[3] out of the segmentation we obtained before. To do so we generate points while we extend variational tangent plane intersection to the point set setting and connect those points to form polygonal faces, i.e. the

[2]The derivation is done so that it applies for more general objects, yet we set our focus on points in Euclidean space.

[3]Such a mesh could be thought of a polygonal surface as a generalization of a simplicial surface.

2-dimensional entities of a polygonal surface. As the underlying surface-like structure is a complex, whereas each cloud might get bounded by multiple topologically closed curves, we are able to generate polygonal faces for not even simply connected regions and thus responding to an open question in [SZP20].

Inside a Cloud The investigation of the microscopic behavior of a cloud is split in two parts. At first we elaborate on an optimal shape of a cloud in Section 5, which is also published in [SZ21]. To find such a shape we incorporate point normal information in a sigmoid-weighted covariance matrix and investigate on a large scale the parameter space of the used weighting function and neighborhood sizes. The quality of a neighborhood is evaluated by an energy model using feature classification entities. In the second part (cf. Section 6) we propose a feature classification model, called flatness model, based on the eigenvalues of a covariance matrix. This model shall measure how planar or not planar a neighborhood is, therefore we would allow some curvature if the distribution of the points resembles a 2-dimensional, topological disk instead of the shape of a curve or volumetric extent. The motivation is to find a feature entity on points that decides whether the point has a surrounding neighborhood which we would assume to be 2-dimensional and thus playing towards a surface, or not. The energy connected to the flatness model provides a smooth (C^∞) model, which we incorporated in a point set denoising pipeline, similar to [YRS+18], yet our model is using an iterative scheme of energy determination (feature classification) and point update, without the requirement of point normals.

Publications Prior to the Thesis

The following list contains publications by this author prior to this thesis. The parts which have already been published ([SZP20][4] and [SZ21]) and discussed in this work are further mentioned above and in the respective sections they appear. Works which do not deal with point set processing are [RZP18] and [SRZ21].

[RZP18] *Two-Layer Woven Surfaces with Planar Faces*
Ulrich Reitebuch, Eric Zimmermann, and Konrad Polthier
In: Proceedings of Bridges 2018: Mathematics, Art, Music, Architecture, Education, Culture, 2018

[YRS+18] *Constraint-based point set denoising using normal voting tensor and restricted quadratic error metrics*
Sunil K. Yadav, Ulrich Reitebuch, Martin Skrodzki, Eric Zimmermann, and Konrad Polthier
In: Computers & Graphics, vol. 74, 2018

[SZP19] *Variational Shape Approximation of Point Set Surfaces*
Martin Skrodzki, Eric Zimmermann, and Konrad Polthier
In: IGS 2019 International Geometry Summit – Poster Proceedings, 2019

[4]The preceding poster [SZP19] and the work [SZP20] itself received the best poster award of the International Geometry Summit (IGS) 2019 and the third place best paper award granted for the special issue on Computational Geometric Design of Computer Aided Design 2021, respectively.

[SZP20]	*Variational shape approximation of point set surfaces* Martin Skrodzki, Eric Zimmermann, and Konrad Polthier In: Computer Aided Geometric Design, vol. 80, 2020
[SZ20]	*Large-scale Evaluation of Neighborhood Weights and Sizes* Martin Skrodzki and Eric Zimmermann In: Joint SPM/SMI 2020 Conference – Poster Proceedings, 2020
[SZ21]	*A Large-Scale Evaluation Of Shape-Aware Neighborhood Weights And Neighborhood Sizes* Martin Skrodzki and Eric Zimmermann In: Computer-Aided Design, vol. 141, 2021
[YSZP21]	*Surface Denoising Based on Normal Filtering in a Robust Statistics Framework* Sunil K. Yadav, Martin Skrodzki, Eric Zimmermann, and Konrad Polthier In: Proceedings of the Forum "Math-for-Industry" 2018, 2021
[SRZ21]	*Investigations of structures in the parameter space of three-dimensional Turing-like patterns* Martin Skrodzki, Ulrich Reitebuch, and Eric Zimmermann In: AUTOMATA2021, 2021

Acknowledgments

First and foremost, I want to thank my advisor Prof. Dr. Konrad Polthier for providing the opportunity to become a doctoral candidate and his continuous support of my research. I am also grateful to him for giving me the chance to adequately balance my work and family. In addition I am thankful that I could take part in the development of JavaView [PRYZ], gaining experience and using this geometry processing environment to develop most of the algorithms, experiments, and visualizations in this thesis.

Second, I am grateful for the funding I received by the Deutsche Forschungsgemeinschaft (DFG - German Research Foundation) - Project-ID 195170736 - TRR109, and thus being part of the project C05 in the collaborative research cluster Discretization in Geometry and Dynamics. The cluster with their offered workshops, talks, annual meetings, summer schools, and lively exchange laid a foundation for new and great research results and still continues to do so.

Third, I thank my (former) colleagues Dr. Konstantin Poelke who encouraged me to do my PhD, Dr. Martin Skrodzki for all the work and achievements we conducted together, further Dr. Sunil Yadav, Ulrich Reitebuch, Henriette Lipschütz, and Jakub Rondomanski for successful collaborations and fruitful discussions.

Finally, and most importantly, I thank my family and friends, especially my wife and children, for their enduring patience, never ending love, and valuable assistance.

Chapter A

Cloud Surfaces

Using points to represent surfaces is an ill-posed task, as surfaces inherit the property in which each point has a 2-dimensional neighborhood. Points in contrast are all independent from each other without any connectivity. However, either the utilization of points as display primitives ([LW85]) or their use to solve various problems on surfaces represented by them led to a broad field of research. To overcome the missing connectivity approaches are mostly applied locally or globally to the point set, i.e. local means that solving a specific task at a point demands a local neighborhood, thus we first find a subset of the points counting as an individual neighborhood for that point in question. A global approach in contrast would deal with the point set at once even in the case of focusing on a specific point. An example of point set treatment is surface reconstruction surveyed in [BTS$^+$17]. For instance in [HDD$^+$92] we can find tangent plane estimation carried out via subsets accounting as local neighborhoods. In [ABCO$^+$03] the authors make use of the *moving least squares* method to compute and render point set surfaces. The method itself uses weight functions without compact support but often with a controllable decay. Thus the points are considered all at once with points being farther away receiving effectively zero weight. This is something the authors mentioned and trimmed their algorithm to. On the very other end the algorithm in [AB98] generates a piecewise-linear approximation of a finite set of points supposed to be taken from a smooth surface using *Voronoi diagram* and *Delaunay triangulation* applied to the whole set of points at once. These shall count as examples in one discipline dealing with points representing surfaces and they neither mark the beginning of such approaches nor the end, as in the following years researchers needed to find solutions to the problem of missing connectivity. The terminology found in [HDD$^+$92] speaking of unorganized points describes it aptly and leads to the question whether we can impose a structure to organize the points in a rough way, so that we can inherit surface properties. Hence, we are building a kind of discrete atlas on the points, with our focus on the selection of subsets included in the charts (see Chapter A and B) and then use these organization on the points to perform specific tasks.

The creation of point neighborhoods for various methods was treated as a necessity, but some works were also dedicated to investigate the controlability and hence quality of such neighborhoods to serve certain goals in point processing, for instance in optimal size analysis for 3d lidar point sets, cf. [DMDV11], to use them for point feature classification in [WJM14]. Another example is to generate additional information like point normals in [MNG04] and at the same time elaborating on the effect of various neighborhood sizes. In contrast, whenever further information is given, they might be included in the neighborhood determination and account as weights like done in [SZ21] which is discussed in more detail in Section 5. In general finding optimal neighborhoods is related to certain tasks, thus optimality refers to a specific measure or purpose. In our scenario, in which we want to select a subset of the power set of the points we are given, such results could guide our choices. Chapter B for instance has a measure to generate subsets of the points where the points within each subset have less point normal deviation. Such a subset could be valid as a neighborhood for all the points it contains and applies to be used in a discrete atlas. In Chapter C we discuss a microscopic perspective on clouds, i.e. addressing subset selections which on one hand could account locally, i.e. an individual subset for every point, or on the other hand a broader selection to inherit a surface beyond.

The tasks to be solved in point processing include for instance surface reconstruction, meshing, or denoising. Notions of differential entities on point sets, which we know from smooth surfaces, are then either a byproduct of the proposed approaches or investigated at their own right. A task we are going to address with a cloud surface is to use the clouds as a rough representation of the points they contain and with positional changes applied to the structure, we want to inherit corresponding changes applied to the points. The advantage of a cloud surface is its underlying (abstract) simplicial surface which acts as a control structure and from which we inherit notions already known. We discuss the idea of point dynamics w.r.t. motions of the control structure. This relates to the research area of *skinning* used in character animation, cf. [Yan22], were deformations of geometries represented mostly by meshes are performed using an underlying skeleton and approaches dealing only with point sets seem rare. The authors in [FCD21] propose a method directly working on point sets encoding the points by a sphere-mesh model which is also used as the skeleton for deformation, yet they focus on the update and referring to other works for the generation and calibration of such model. Similar to this approach and in contrast to other works we also do not incorporate point weights. However, using our control structure, we can use surface entities derived from the geometric realization of the skeleton for the deformation. The intent for cloud dynamics was to find a more simple representation using a control structure which itself might represent a large number of points and apply motions to the structure and possibly on the points on demand. Such operations then are used in character animation and skinning. Hence we propose a generation of a skeleton for points in our scenario and discuss possible update schemes, both without the requirement of further information.

To conclude the contributions of which some are mentioned above we get:

- Definition of a cloud surface, i.e. a set system on abstract vertices which induces an abstract simplicial surface as underlying structural object.
- Generation and results of a cloud surface from a simplicial surface via probabilistic sphere models.
- Generation and results of a cloud surface from points in $\mathbb{R}^3$. Discussion of the influence of sampling properties towards the generation results.
- Topological discussion on coverings used for the generation of a cloud surface.
- Description and discussion of point dynamics w.r.t. the underlying structure.

1 Description of Cloud Surfaces

We are going to derive the basis for cloud surfaces starting with a general notion of clouds which are subsets of some given set. Such set appears to be abstract, an index set for instance, mapping itself to other objects, e.g. points in a Euclidean space. We first agree on the following notation.

Remark 1. *For a countable set X and an arbitrary set I we write $X_I = \{x_i\}_{i\in I}$, thus X_I becomes a family.*

Definition 1. *Let V denote a countable set called* abstract vertices *(or abstract vertex set) and an element $v \in V$ is called an* abstract vertex.

Definition 2. *Let V be an abstract vertex set and X a space. A map $\psi : V \to X$ is called a* vertex map *and its image $P := \psi(V)$* points.

Definition 3. *Let V be an abstract vertex set. A countable set $\mathcal{C}$ of subsets of V is called a* cloud set *(or clouds), abbreviated $\mathcal{C}(V)$ or $\mathcal{C}$, and an element in $\mathcal{C}$ is called a* cloud.

Example 1.

i) The edge set of a graph is a cloud set.

ii) An abstract simplicial complex (see Definition 4) is a cloud set.

iii) Let $V_{\mathbb{N}}$ be an abstract vertex set, $\psi : V_{\mathbb{N}} \to \mathbb{N}$, $v_i \mapsto i$ a vertex map and $\mathcal{C}_{\mathbb{N}} = \{C_i\}_{i\in\mathbb{N}}$ with

$$C_i = \left\{ \left(\sum_{j=1}^{i-2} j \right) + 1 + k \mid k = 0, \dots, i-1 \right\}.$$

Then $\mathcal{C}_{\mathbb{N}}$ is a cloud set in which every two consecutive clouds have non-empty intersection and every successor has one more element than its predecessor. For instance $C_1 = \{1\}$, $C_2 = \{1, 2\}$, and $C_3 = \{2, 3, 4\}$.

iv) Let (X, d) be a metric space, V an abstract vertex set, and $\psi : V \to X$ a vertex map with $P = \psi(V)$. Further let $r \in \mathbb{R}_{\geq 0}$. The geometric neighborhood of *$x \in X$ is the subset of P w.r.t. r given by*

$$N^g(P, x, r) = \{p \in P \mid d(x, p) \leq r\}.$$

A cloud set $\mathcal{C}$ on V then can be given as a subset of

$$\left\{\psi^{-1}\left(N^g(P, p, r)\right)\right\}_{p\in P} \text{ for } r \in \mathbb{R}_{\geq 0}.$$

v) Let (X, d) be a metric space, V_I an indexed abstract vertex set for an index set I, and $\psi : V_I \to X$ a vertex map which is additionally bijective with $P_I = \psi(V_I)$. Further let $k \in \mathbb{N}_0$ with $k \leq \#\{P_I\}$[1]. The combinatorial neighborhood of *$x \in X$ is the subset of P_I w.r.t. k given by*

$$N^c(P_I, x, k) = \operatorname*{arg\,min}_{\tilde{N}\in\tilde{N}(P_I,x,k)} \left\{ \sum_{p_i\in\tilde{N}} i \right\} \quad \text{with}$$

$$\tilde{N}(P_I, x, k) = \operatorname*{arg\,min}_{\tilde{P}\subseteq P_I, \#\{\tilde{P}\}=k} \left\{ \sum_{p_i\in\tilde{P}} d(p_i, x) \right\}.$$

[1]Let $\#\{A\}$ denote the cardinality of a set A.

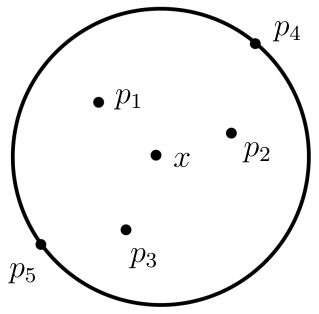


Figure 1.1: A choice for $k = 4$ yields that p_4 belongs to the combinatorial neighborhood of x but not p_5, thus it provides a unique selection.

A cloud set $\mathcal{C}$ on V_I then can be given as a subset of

$$\bigcup_{i \in I} \left\{ \psi^{-1} \left(\tilde{N}(P_I, p_i, k) \right) \right\} \text{ for } k \in \mathbb{N}_0, k \leq \# \{P_I\}.$$

Remark 2. *The last two examples in Example 1 are common approaches to obtain nearest neighbors in a point set. Nonetheless, looking for a natural amount of neighbors does not provide unique subsets in general. In the example however, $\tilde{N}(P_I, x, k)$ holds all possible candidates, yet $N^c(P_I, x, k)$ provides a unique representative, see Figure 1.1.*

The clouds are up to this point just a collection of sets without further structure. We build an underlying skeleton for the clouds achieved from an abstract simplicial complex which in addition fulfills the property of being an abstract version of a simplicial surface. We define an abstract simplicial complex similar to the one in [Lee00] as follows:

Definition 4. *A collection K of non-empty finite sets is called an* abstract simplicial complex *if for $\sigma \in K$ we have that every subset of σ is in K. If K is such a complex, a set $\sigma \in K$ is called a* simplex *(or d-simplex, also denoted σ^d, with $d = \# \{\sigma\} - 1$) and $K^{(d)} \subseteq K$ denoting the subset containing all d-simplices for $d \in \mathbb{N}_0$. A non-empty subset of a simplex σ is called a* face *of σ.*

Now we set up a bijection to identify the clouds with some abstract vertex set which gives us the opportunity to define a cloud complex.

Definition 5. *Let $\mathcal{C}$ be a cloud set and $\mathcal{V}$ a set of abstract vertices with $\# \{\mathcal{V}\} = \# \{\mathcal{C}\}$. Then a bijection $\Psi : \mathcal{C} \to \mathcal{V}$ is called a* cloud vertex map *and the set $\mathcal{V}$* cloud vertices*.*

Definition 6. *Let $\mathcal{C}_I$ be a cloud set of finite clouds, Ψ a cloud vertex map, and $\mathcal{V}_I$ the resulting cloud vertices. Then we define a complex K on $\mathcal{V}_I$ with elements*

$$\sigma^m = \{v_{i_0}, \ldots, v_{i_m}\} \in K(\mathcal{V}_I) \iff \bigcap_{k=0}^{m} \Psi^{-1}(v_{i_k}) \neq \emptyset \text{ for } m \in \mathbb{N}_0.$$

This complex is called a cloud complex*.*

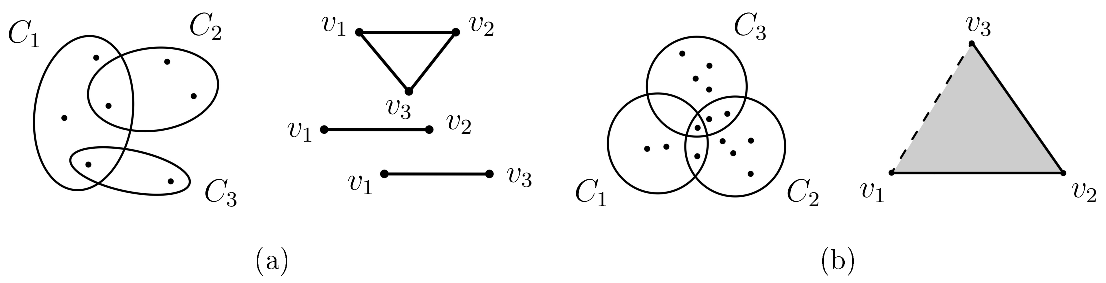


Figure 1.2: Note that all drawings shown here are realizations (cf. Definition 10) of abstract entities. (a) The three clouds C_1, C_2, C_3 on 7 abstract vertices cause three cloud vertices v_1, v_2, v_3 and with the relation mentioned in Remark 3 i) we do not achieve an abstract simplicial complex. (b) Shown are three clouds and a realization of the 2-simplex they cause. Due to the definition of a cloud complex we have $C_1 \cap C_2 = C_1 \cap C_2 \cap C_3$.

Proposition 1. *The complex defined in Definition 6 is an abstract simplicial complex of dimension*

$$\max_{\substack{\mathcal{C}' \subset \mathcal{C} \\ \bigcap_{C \in \mathcal{C}'} C \neq \emptyset}} \{\# \{\mathcal{C}'\}\}.$$

Proof. Let K be a cloud complex and $\sigma^d \in K$. Then by Definition 6 all subsets of σ^d must have non-empty intersection via Ψ^{-1} thus constituting a face of σ^d and therefore being contained in K. As all clouds are finite we have that K is an abstract simplicial complex. The dimension follows from the simplex of highest size and as the size of a simplex corresponds to the number of clouds having non-empty intersection. □

Remark 3. *i) A cloud complex is defined to use the intersection of sets to decide whether sets are related or not*[2]*. A similar object is the nerve of a cover, see [Hat10], which yields a simplicial complex with a vertex for each element in the cover and a k-simplex if the corresponding $k+1$ elements of the cover have non-empty intersection. However, this works on topological spaces and our clouds as a cover deals with a discrete set without an equipped topology. It is something we could do, but is neither necessary nor advisable in our scenario. Especially as we may want to have more flexibility in the choice of cloud relation besides the intersection, it is also not of importance in the upcoming discussion. For instance, imagine we vary Definition 6 in terms of its elements, i.e.*

$$\sigma^m = \{v_{i_0}, \dots, v_{i_m}\} \in K(\mathcal{V}_I) \iff \sum_{k=0}^{m} \# \{\Psi^{-1}(v_{i_k})\} > N \text{ for } m, N \in \mathbb{N}_0.$$

This example would not give us an abstract simplicial complex, as larger elements in K not necessarily contain its subsets, see Figure 1.2(a). Such a relation on clouds coupled with the resulting complexes could be a direction for future investigation.

ii) The definition of a cloud complex does not provide a unique identification between abstract vertices (sitting inside the clouds) and the simplices they cause in the cloud complex. In Figure 1.2(b) we have an example in which we do not find abstract

[2]Constructing a graph out of this property would yield an intersection graph, see [Wes01].

vertices lying only in the intersection of C_1 and C_2. Thus by definition, vertices in all three clouds cause the 2- *and* 1-*simplex at once omitting an identification. In our scenario this generalization is valid and gets treated properly in Section 2.3 when an unique reference for each abstract vertex is needed. If necessary, we could restrict the definition on cloud complexes and add the condition that a simplex does only exist if the respective clouds have elements in their intersection which do not appear in any other cloud.*

With the complex to serve as an underlying structure, we proceed to demand an important property, namely that it appears to be a surface. Therefore, we turn to the following definition, see [BNPS17].

Definition 7. *Let K be an abstract simplicial complex on abstract vertices V with $\#\{V\} < \infty$, i.e. K is finite. Let $K^{(1)}$ and $K^{(2)}$ denote the sets of* 1- *and* 2-*simplices, resp. We call K an* abstract simplicial surface *if the following conditions hold:*

i) $V = \bigcup_{\sigma^2 \in K^{(2)}} \sigma^2$,

ii) any 1-*simplex in $K^{(1)}$ is contained in at most two* 2-*simplices in $K^{(2)}$, and*

iii) for any 0-*simplex $v \in V$ the set of all* 2-*simplices containing v can be arranged in a sequence $(\sigma_1^2, \dots, \sigma_n^2)$ s.t. σ_i^2 and σ_{i+1}^2 have a* 1-*simplex in common, for $i = 1, \dots, n-1$.*

It is called closed *if any* 1-*simplex belongs to exactly two* 2-*simplices.*

Observe that the third property seeks to mimic the behavior of manifolds with boundaries, which are in their basic topological form (for a d-manifold) second countable Hausdorff spaces in which each point has a neighborhood homeomorphic to either an open subset of Euclidean space of dimension d, or to an open subset of the closed d-dimensional upper half space, see [Lee00]. Thus having such sequence of 2-simplices around a 1-simplex shall preserve that property of a 2-dimensional manifold. For the interior of 2-simplices it is fulfilled by definition and also for the interior of 1-simplices as we allow up to two 2-simplices to contain such 1-simplex as a face, despite the fact that we do not know by now what the interior of a 1- or 2-simplex means, but the abstract combinatorial setup lays ground for the geometric appearance later on.

Definition 8. *A finite cloud set is called a* cloud surface *if its cloud complex is an abstract simplicial surface.*

Currently we focus on abstract simplicial complexes and its surface analogues, because we want to use them as an underlying structure for point data. Thus from the three defining properties of simplicial complexes provided in [Lee00, §5] we still keep that every face of a simplex (which is a subset of that simplex) in a complex K belongs to K (by Definition 4) and that K is a locally finite collection. The latter is guaranteed, as we only deal with at most countable sets of data (see Definition 1). However, the point data we work with later is finite. The only property we miss is that the intersection of any two simplices is either empty or a face of each. But this property is not important to us at the moment. Later we will work with the interior of simplices, but to do so we will use combinations of the simplex' vertices or indeed their geometric representation on which we will focus on next.

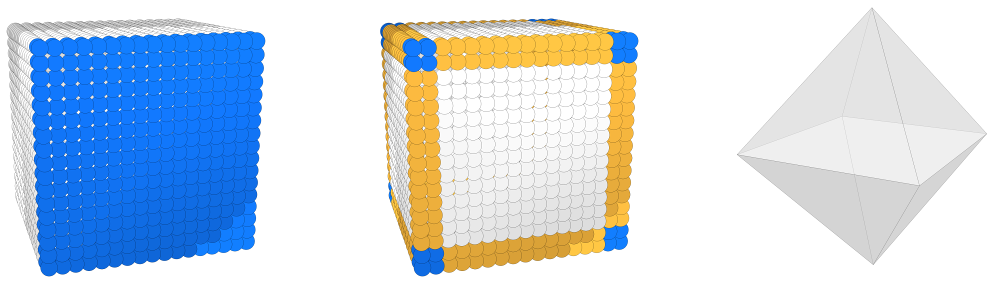


Figure 1.3: Left shows one highlighted cloud (🟦). The center image colors every point w.r.t. to the number of clouds it belongs to, i.e. either one (⬜), two (🟧), or three (🟦). (More than three are not possible as we set our focus on cloud surfaces). Right then gives a geometric realization of the respective cloud complex.

Definition 9. *Let $\mathcal{C}$ be a cloud complex, $\mathcal{V}$ its cloud vertices, X a space, and $\psi : V \to X$ a vertex map. Then we call $\mathcal{P} := \psi(V)$* cloud points.

The cloud points provide a geometric interpretation of a cloud complex and it uses the vertex map from Definition 2. However, to address the cloud complex together with its cloud points at a later stage we make use of the geometric realization of a simplicial complex described in [Lee00], here stated without the necessity that simplices intersect only in their faces.

Definition 10. *Let K be an abstract simplicial complex and $\psi : K^{(0)} \to \mathbb{R}^d$ a vertex map, $d \in \mathbb{N}_0$. The* geometric realization of a simplex *$\sigma \in K$, denoted $|\sigma|$, is the convex hull of the cloud points referring to σ, i.e.*

$$|\sigma| = conv\{\psi(v) \mid v \in \sigma\}.$$

The geometric realization of a collection of simplices *K is the union of the geometric realizations of all the simplices it contains, i.e.*

$$|K| = \bigcup_{\sigma \in K} |\sigma|.$$

If K is an abstract simplicial surface, we call $|K|$ a (simplicial) pseudo-surface, *with $|\sigma|$ called a* vertex, edge, *and* triangle *if it is of dimension 0, 1, or 2 and σ is a 0-, 1-, or 2-simplex, respectively.*

Applying the geometric realization on a collection of simplices allows us to either have a realization of an abstract simplicial complex, or to have a subset of it like $K^{(1)}$, i.e. the set of 1-simplices in K.

Example 2. *Consider points representing a cube in $\mathbb{R}^3$ with six clouds, where each cloud reflects one face of the cube with a slight overlap. Figure 1.3 represents a choice of clouds and a coloring for the points reflecting the number of clouds they belong to as well as a geometric representation of the inherited cloud complex.*

All notions covered so far can be summarized as follows (see Figure 1.4): From a set of abstract vertices V corresponding to points P via the vertex map ψ we take a set of subsets $\mathcal{C}$, i.e. instantiate a set system, which we call clouds. Note that this set coupled with V induces the same set system on P. These can be considered objects

$$\begin{tikzcd}
 & V \arrow[dl, leftrightarrow, "\psi"'] \arrow[dl, bend left, "induced"] \arrow[dr, leftrightarrow, "set system"] & & \mathcal{V} \arrow[rr, leftrightarrow, "\psi"] \arrow[dr, bend left, "induce"] & & \mathcal{P} \arrow[dr, bend left, "induce"] & \\
P & & \mathcal{C} \arrow[ll, bend right] \arrow[ur, "\Psi"] \arrow[rr, bend left] & & K \arrow[rr, bend left] & & {|K|}
\end{tikzcd}$$

Figure 1.4: Illustrates interrelations of the introduced objects abstract vertices V, points P, clouds $\mathcal{C}$, cloud vertices $\mathcal{V}$, cloud points $\mathcal{P}$, cloud complex K, and its geometric realization $|K|$. Further, we find the vertex map ψ (here two times with different domains in general) and the cloud vertex map Ψ. Note that the clouds taken on V also provide an induced set system on P via ψ.

of their own right, thus via a cloud vertex map Ψ we obtain cloud vertices $\mathcal{V}$ and geometric interpretations, i.e. cloud points $\mathcal{P}$ via a similar vertex map ψ again. The clouds together with the cloud vertices induce a complex K, called cloud complex, which is an abstract simplicial complex and possibly an abstract simplicial surface. The geometric realization $|K|$ of such (induced by $\mathcal{P}$ and K) then serves as an underlying control structure for the clouds $\mathcal{C}$ (and the points P they contain).

2 Generation and Dynamic

One natural question following the introduction of cloud surfaces, besides their applications, is how we can construct them. Due to their property carrying less information like connectivity the choice of subsets of points having that they form an underlying surface-like structure is difficult. To achieve examples to work with, we first investigate the reversed direction, i.e. we take a simplicial surface and generate points in a probabilistic manner and assign them to clouds corresponding to the structure encoded in the simplicial surface, cf. Section 2.1. The second approach takes points in $\mathbb{R}^3$ (also $\mathbb{R}^2$) and cut them out by balls attached to a scaled grid, whereas this covering provided by these balls tries to mimic the property describing the cloud surfaces themselves (cf. Definition 8), see Section 2.2. The geometric realization of a cloud surface accounts for some type of skeleton, i.e. changing cloud point positions shall then have an impact on the points the clouds contain. We investigate this in Section 2.3.

2.1 Generation from Pseudo-Surfaces

For the generation of points and clouds containing those, we at first consider pseudo-surfaces and present three probabilistic models applied to scaled spheres attached to the surfaces' vertices. These models then get used in an algorithm for which we discuss experimental results in Section 2.1.2.

2.1.1 Probabilistic Sphere Models

Suppose we are given a pseudo-surface in $\mathbb{R}^3$. As by Definition 6 and Proposition 1 a triangle (or edge) in the geometric realization of a cloud surface is caused by the intersection of three (or two) clouds, cf. Definition 6, we use these now to generate points belonging to respective clouds. Therefore, for each triangle of the pseudo-surface we attach to each of its vertices a 3-ball with a radius, s.t. each ball includes the barycenter of the triangle in question. Then we equip each ball with density functions w.r.t. the parametrization of the ball and create point samples via a probabilistic model. These samples are then distributed into clouds which we call responsible for the triangle and its faces.

Two density functions will interact with the parametrization of a ball, namely one horizontally (δ^h) and one vertically (δ^v). The only information we take from a triangle is the angle α between the two edges incident to the vertex the ball is attached to, see Figure 2.1. Therefore, we can simplify the approach. Consider a triangle $|\sigma^2|$ with its three vertices v_1, v_2, v_3 of the pseudo-surface in question. Recall that a vertex here is the geometric realization of an abstract vertex belonging to the surface's underlying complex (cf. Definition 10). Then we take the following

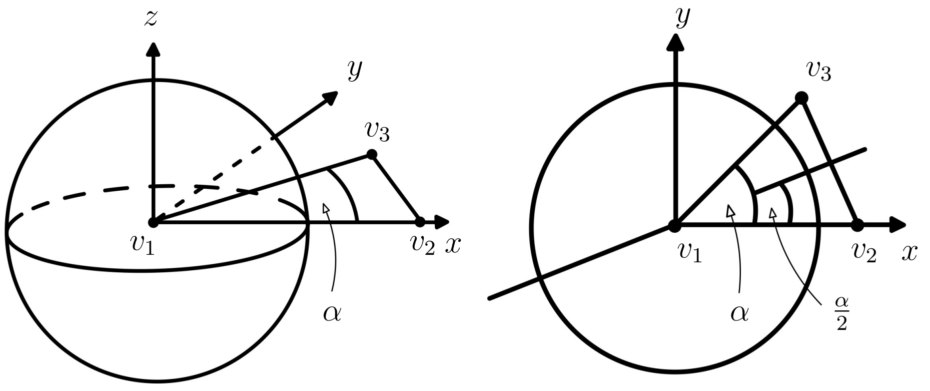


Figure 2.1: Shows a triangle lying in the xy-plane with v_1 at the origin and v_2 aligning with the x-axis. The left then shows the sphere put at vertex v_1 with spanning angle α between the two incident edges. The right illustrates the top view of how the angles distribute starting at $\alpha/2$ for the density function generation.

parametrization for a ball attached to vertex v_1, i.e.

$$\begin{aligned} f &: [0,R] \times [0,2\pi) \times [0,\pi] \to \mathbb{R}^3 \\ (r,s,t) &\mapsto r \begin{pmatrix} \cos(s)\sin(t) \\ \sin(s)\sin(t) \\ \cos(t) \end{pmatrix} \quad \text{where} \\ R &\in \left[\frac{1}{3}\sum_{i=1}^{3} v_i, \min\left\{ \|v_2 - v_1\|_2, \|v_3 - v_1\|_2 \right\} \right) \end{aligned} \tag{2.1}$$

is the range s.t. the ball includes the triangles' barycenter but not one of the other vertices.

The two density functions δ^h and δ^v shall act in the following way: The horizontal function applies to the angular parameter of the parametrization above, i.e. it gives 1 as long as it stays inside the triangle and otherwise it decreases towards a value larger or equal 0 at $\pi + \alpha/2$, see Figure 2.1. The vertical function deals with the height and offers a decrease from 0 to 1 starting at the origin going up and down. Note, that the superscript v in δ^v means vertical and does not refer to a vertex. We give the following examples:

Example 3. *i) Piecewise linear density functions for $c \geq 0$ and $\alpha \in [0,\pi)$ can be given as*

$$\delta_1^v : [0,\pi] \to [0,1], \quad t \mapsto \begin{cases} \frac{2t}{\pi} & \text{if } \; t < \frac{\pi}{2} \\ 2 - \frac{2t}{\pi} & \text{if } \; t \geq \frac{\pi}{2} \end{cases} \quad \text{and}$$

$$\delta_1^h : [0,2\pi) \to [0,1], \quad s \mapsto \begin{cases} 1 & \text{if } \; s \leq \alpha \\ \frac{1}{c+1}\left(\frac{1}{\pi - \frac{\alpha}{2}}\left(\frac{\alpha}{2} + \pi - s\right) + c\right) & \text{if } \; \alpha < s \leq \frac{\alpha}{2} + \pi \\ \frac{1}{c+1}\left(\frac{1}{\pi - \frac{\alpha}{2}}\left(s - \frac{\alpha}{2} - \pi\right) + c\right) & \text{if } \; \frac{\alpha}{2} + \pi < s \end{cases} .$$

ii) Piecewise trigonometric density functions for $c \geq 1$ and $\alpha \in [0,\pi)$ can be given

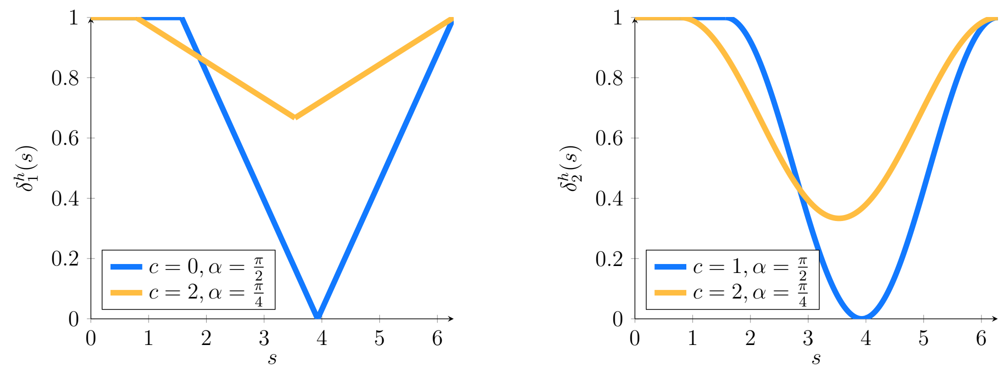


Figure 2.2: Plots of possible density functions δ_1^h and δ_2^h provided in Example 3.

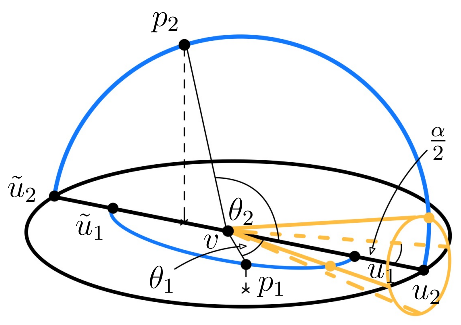


Figure 2.3: Illustrates the second model where angle α of the triangle at v_1 causes a cone in which the density receives value 1 and from the boundary from the cone towards the opposite point $\tilde{u}$ on a circular arc the density decreases linearly.

as

$$\delta_2^v : [0, \pi] \to [0, 1], \quad t \mapsto \sin(t) \textit{ and}$$

$$\delta_2^h : [0, 2\pi) \to [0, 1], \quad s \mapsto \begin{cases} 1 & \textit{if} \quad s \leq \alpha \\ \frac{1}{c+1}\left(c + \cos\left(\pi + \frac{\pi}{\pi - \frac{\alpha}{2}}\left(\frac{\alpha}{2} + \pi - s\right)\right)\right) & \textit{if} \quad \alpha < s \end{cases}.$$

Both yield combined density functions $\eta_i = \delta_i^v \cdot \delta_i^h$, $i = 1, 2$. Examples of function plots for δ_i^h are illustrated in Figure 2.2 and of η_i in Figure 2.5 in Section 2.1.2. Note that with $c \geq 0$ we get $\delta_1^h \geq 0$ for all s, whereas $\delta_2^h \geq 0$ for $c \geq 1$ for all s.

Observe that with δ^h approaching 0 yields the multiplication of both densities δ^h and δ^v leaves narrow probability for the existence of points if those are not above or below the triangle in the ball, see Figure 2.5(a) and 2.5(c) in the experiments. By an abuse of notation concerning δ^h and δ^v we provide another model.

Suppose again the ball centered at v_1, i.e. the origin, and an opening angle α, cf. Figure 2.3. Now consider the line bisecting α in the xy-plane with $\tilde{u}$ the intersection of that line with the boundary of the ball at distance R. We want the density to be 0 at that point. Thus, we let the horizontal density δ^h decrease from starting at v_1 going on the line towards $\tilde{u}$. Now let $p \neq 0$ be a point in the ball with $r = \|p\|_2$ the distance to v_1. Then we find a circle of radius r including p and intersecting the bisecting line in two points which we call u (lying in the triangle)

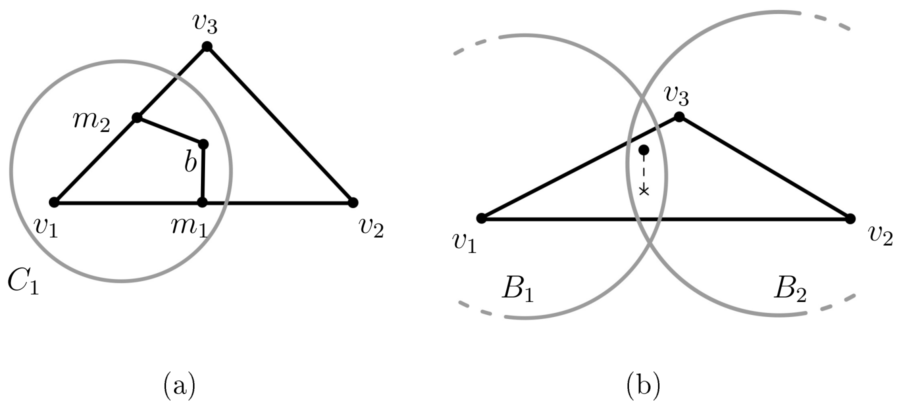


Figure 2.4: Left shows that a cloud at a vertex contains the vertex itself, the barycenter and edge midpoints of the adjacent triangle and edges respectively. Right illustrates how the new point in question, generated in the ball B_1 at v_1 also belongs to the cloud belonging to v_2.

and $\tilde{u}$. Now we want the vertical density δ^v to pass the arc of the circle starting in u going towards $\tilde{u}$. Then we set the arc length between u and p in relation to π and apply a decreasing function. Note that instead of considering the arc length we can use the angle between vector p and u, because the ball is attached to the origin and the influence of radius r on the arc length can be ignored, as it scales the lengths between u and p and u and $\tilde{u}$ similarly. Further we let α operate in such a way that it has the form of a cone (rotating the triangle around the bisection line) which is then reflected in the corresponding density function as well.

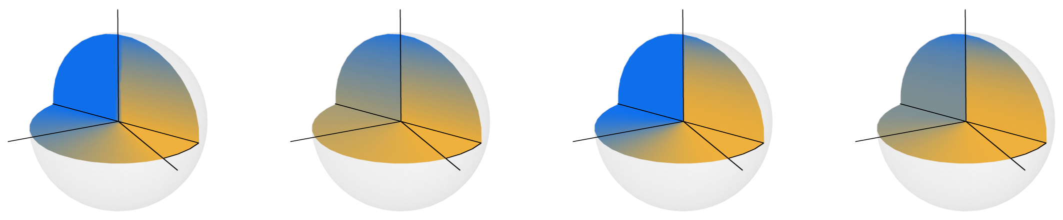
Example 4. *Let $p \in \mathbb{R}^3$, $r = \|p\|_2 \leq R$, $c_v \geq 0$, $c_h \geq 0$, and $\alpha \in [0, \pi)$ with R introduced in Equation (2.1). The two density functions then can be given as*

$$\delta_3^v : [0,\pi] \to [0,1]\,, \quad t \mapsto \begin{cases} 1 & \text{if} \quad t \leq \frac{\alpha}{2} \\ \frac{t(c_v-1)+\pi-c_v\frac{\alpha}{2}}{\pi-\frac{\alpha}{2}} & \text{if} \quad \frac{\alpha}{2} < t \end{cases},$$
$$\delta_3^h : [0,R] \to [0,1]\,, \quad s \mapsto -s\left(\frac{1-c_h}{R}\right) + 1$$

with combined density function $\eta_3 = \delta_3^v \cdot \delta_3^h$. Examples of function plots of η_3 are shown in Figure 2.6 in Section 2.1.2.

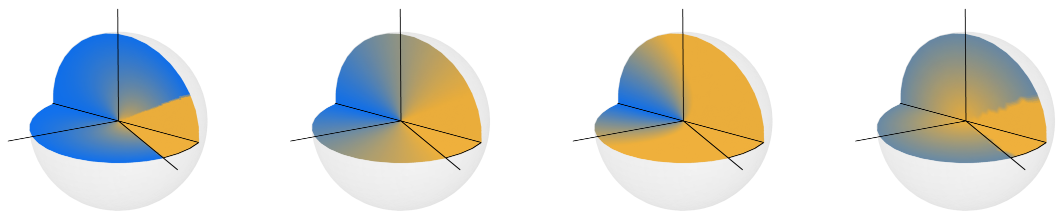
Algorithm 1. *To generate a point p, we draw random numbers in the ball (Equation (2.1)). This can be done by three random values in $[0,1]$ to get a point, normalize, and scale it with another random value in $[0,R]$. The point p then exists with probability $\eta(s,t)$ with $(r,s,t) = f^{-1}(p)$ for functions in Example 3 or $s = \|p\|_2 = r$ and $t = \cos^{-1}(pu/r^2)$ with $u = f(r, \alpha/2, \pi/2)$ for functions in Example 4. Observe that u is of length r and rotated by $\alpha/2$ around the z-axis.*

To generate clouds we set up an empty cloud for each vertex in the pseudo-surface. To achieve a cloud surface, we fill in each cloud the barycenters of the triangles incident to the corresponding vertex, as such points exist with probability 1 anyway (see Figure 2.4(a)). Afterwards, we generate random points for each vertex and triangle, fill them in the respective clouds and also in the clouds attached to the



(a) $c = 0$ (b) $c = 2$ (c) $c = 1$ (d) $c = 3$

Figure 2.5: Shows plots of η_1 (two on the left) and η_2 (two on the right) as given in Example 3 from low (🟦) to high (🟧) values. The angle $\alpha = \pi/3$ is marked with its half as a triangle.



(a) $c_h = 0, c_v = 0$ (b) $c_h = 1, c_v = 0$ (c) $c_h = 2, c_v = 0$ (d) $c_h = 0.4, c_v = 1$

Figure 2.6: Shows plots of η_3 as given in Example 4 from low (🟦) to high (🟧) values. The angle $\alpha = \pi/3$ is marked with its half as a triangle.

other two vertices if they lie in their respective balls, see Figure 2.4(b). Note that all computations can be carried out in the xy-plane as the vertex the ball is attached to is the origin, one edge u_1 aligns with the x-axis and the other edge u_2 in such a way that the product $u_1 \times u_2$ points in positive z direction. Then we are able to transform the sample points to their respective locations in $\mathbb{R}^3$ using the triangles' vertex information.

2.1.2 Experimental Results

In the following we are going to discuss three experiments on the generation of clouds from pseudo-surfaces, i.e. the three proposed combined density functions from Examples 3 and 4, their application to geometries using Algorithm 1, and an incorporation of geometric properties into the generation of such clouds.

Combined Density Functions We proposed three different models to generate points covered in clouds upon a pseudo-surface. All three provide freedom how fast the densities shall decrease or if at all. Illustrative examples are shown for all models in Figures 2.5 and 2.6. The model there is a sphere with some slice cut-off highlighting the three axes and half the angle of α obtained from placing the sphere at one of the triangles' vertices. The color coding then ranges as combined density from low probability in blue to high in orange.

The first two, namely the linear and trigonometric models given in Example 3, are fairly similar in their behavior. By definition they assign value 1 to the angle caused by the triangle and then decrease towards to the opposite, which is $\pi + \alpha/2$. Especially setting c in either case to the lower bound causes a direct jump in density when leaving the triangle behind the vertex the sphere is attached to.

A more continuous transition can be obtained by increasing c in both cases, yet still preserving the small density jump. This might be beneficial when we want to focus the generation of points above or below the triangle and possibly a bit across the edges. However, the pseudo-surface is not flat in general at such a vertex, thus, due to the curvature, there might be a conical volume from all the triangles meeting at the vertex in which we would then obtain less points inserted into clouds. To enhance this, our third example provides a more flexible and continuous transition. In Figure 2.6 especially the first and fourth show the conical extension, which is preserved with density 1, where the two in the center provide a continuous decrease. Thus, the before mentioned areas which are not met above or below the triangles in normal directions achieve a higher probability for the point generation.

Combined Density Functions on Geometries A first example is the application of the linear model from Example 3 with $c = 0$ to the Cube model with 386 vertices and a number of 10 randomly selected points. In a second one we use the trigonometric model from Example 3 with $c = 1$ on the Hand model with $1,577$ vertices and a number of 5 randomly selected points. This model contains a boundary at the wrist. Respective results are illustrated in Figures 2.7 and 2.8.

Each of the figures show in the center row the surface the model is applied to above and below a geometric realization of the cloud complex obtained from the generated clouds which in both cases is again a simplicial surface. Something which is not obvious from the alignment of the generated (colored) points reflecting the clouds displayed in the right column in each of the figures.

This provides several directions for further investigation. First, we have forced in Algorithm 1 that a triangles' barycenter belongs to each cloud reflected by the triangles' vertices. Here we do not extend it to the edges which is not necessary but might be important for other considerations. This refers to Remark 3 ii) with the missing unique identification. Second, the placement of points does not reflect the spatial behavior of the cloud (compared to for example results discussed in Section 2.2). This is mainly due to the probabilistic behavior of the models and an exploration of those could be linked to the investigation of the combined density functions themselves. Third, at the moment the generation of clouds from surfaces gives a possibility to generate such objects in a synthetic way to work with them. However, obtaining a triangular mesh from the clouds raises the question whether other polygonal meshes could be obtained and thus providing the possibility for re-meshing into quads for instance.

Incorporation of Geometric Properties The process of point generation can be steered by several geometric attributes. Here we incorporate discrete versions of Gaussian and mean curvature as implemented in [PRYZ] of the pseudo-surface, i.e. the randomly generated points from Algorithm 1 at each vertex in the pseudo-surface are normalized, i.e. re-scaled w.r.t. the minimal and maximal value of curvature found at vertices of the pseudo-surface. As this is a first attempt to incorporate information of this kind, another direction could be to define a function on the pseudo-surface reflecting the curvature, for instance Gaussian or mean, and let this further influence the combined density model. Results are shown in Figure 2.9.

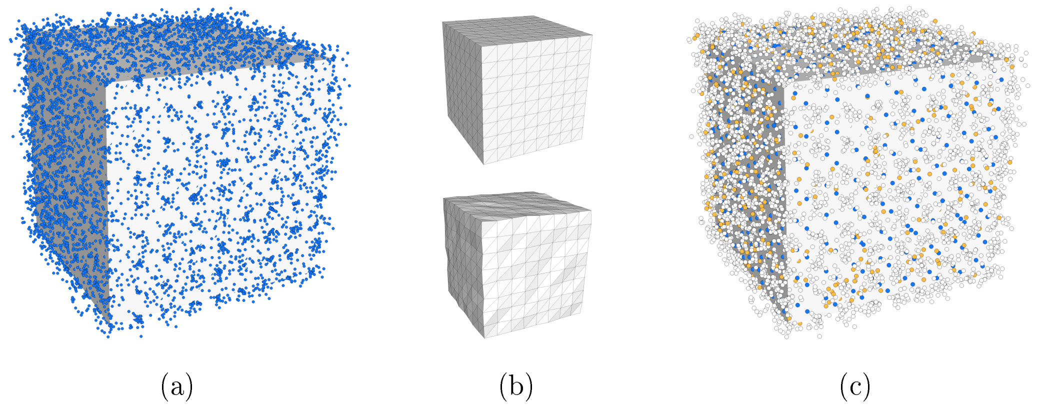

(a) (b) (c)

Figure 2.7: Shows cloud generation on a cube mesh with 386 vertices where (a) represents the generated points (🟦), (b) illustrates the initial mesh on top and a geometric realization of the cloud complex below, and (c) displays coloring of points according to Example 2.

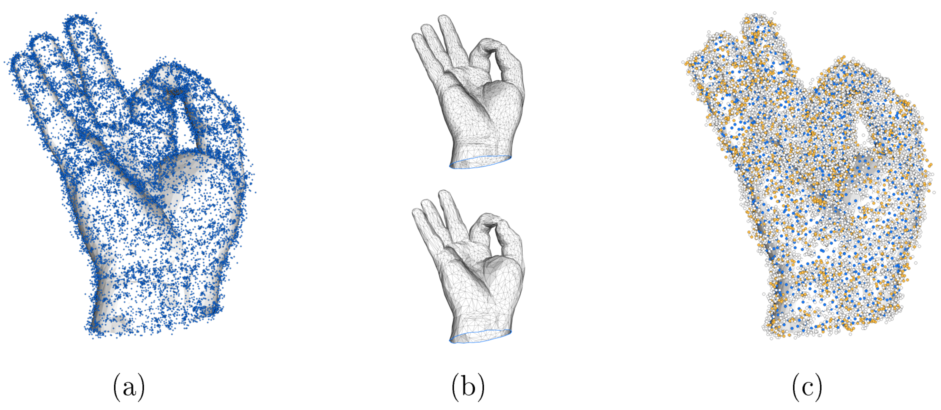

(a) (b) (c)

Figure 2.8: Shows cloud generation on a hand mesh with $1,577$ vertices where (a) represents the generated points (🟦), (b) illustrates the initial mesh on top and a geometric realization of the cloud complex below, and (c) displays coloring of points according to Example 2. Note that the mesh is not closed, i.e. both meshes in (b) have a boundary at the respective bottoms.

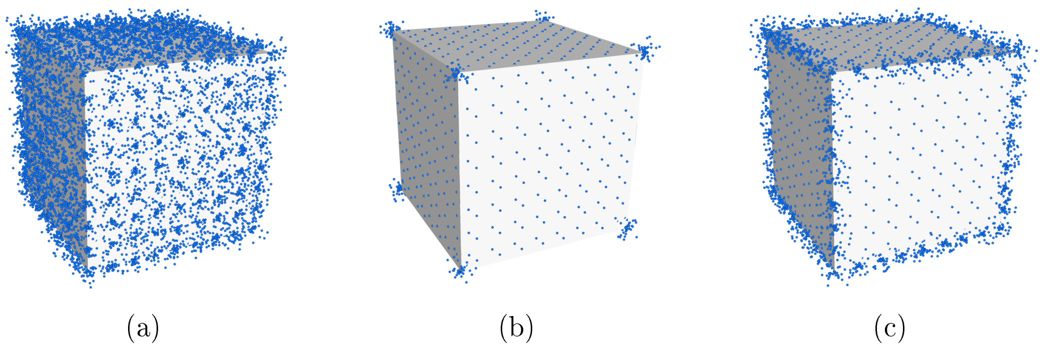

(a) (b) (c)

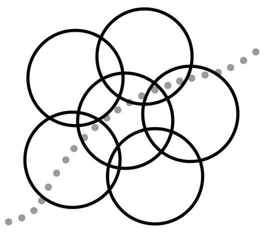
Figure 2.9: Shows cloud generation on a cube mesh with 386 vertices using η from Example 4 normally (a), respecting Gaussian (b), or mean curvature (c).

2.2 Generation from Points

A cloud surface contains a structure we can find almost equally in coverings of the Euclidean space respecting certain conditions. In the following subsection we investigate coverings of the Euclidean space and some of the necessary conditions for our setup which raise other topological questions. With direct covering constructions for the spaces $\mathbb{R}^2$ and $\mathbb{R}^3$ we obtain an algorithm and thus discuss experimental results afterwards in Section 2.2.2.

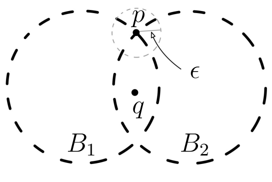
2.2.1 Cloud Surfaces and Coverings of the Euclidean Space

Now suppose the contrary direction of the generation from pseudo-surfaces discussed before. We are given a set of points in Euclidean space which we in our case assume to originate from a topological manifold of dimension 2 embedded in $\mathbb{R}^3$. (On the way we also discuss constructions which allow to treat 1-manifolds in $\mathbb{R}^2$). We want to select clouds (subsets) of these points such that the cloud set is a cloud surface reflecting a coarse approximation of the manifold in question. The idea here is to use the property defining a cloud surface or moreover the necessary cloud complex (cf. Definitions 6 and 8), i.e. the intersection of clouds cause simplicies to exist and we never allow more than three clouds to have a non-empty intersection. Thus, if we can find a covering[3] of the space with balls the points live in, such that the sets of the covering share the same property a cloud surface has, we could use these sets to act as clouds with the points naturally lying in them, see image above. We start our observations using open balls in $\mathbb{R}^2$.



Proposition 2. *For every cover of* $\mathbb{R}^2$ *consisting of open balls exists a point in* $\mathbb{R}^2$ *which is contained in at least* 3 *balls of the cover.*

Proof. Let $\mathcal{B}$ be a cover of $\mathbb{R}^2$ with $B \in \mathcal{B}$ denoting an open ball, $\bar{B}$ its closure, and $\partial\bar{B}$ its boundary. Let $B_1, B_2 \in \mathcal{B}$ be two balls with $\bar{B}_1 \cap \bar{B}_2 \neq \emptyset$. If we assume that this intersection is empty and it would hold for all pairwise elements in $\mathcal{B}$, then this cover would consist of a sequence of balls where a successor contains its predecessor and as all radii are in $\mathbb{R}$ but not infinite, we find such point lying in all balls, i.e. more than two. Now let $p \in \partial\bar{B}_1 \cap \partial\bar{B}_2$ be a point lying on both boundaries. As p is not in B_1 and B_2 there is a B_3 with $p \in B_3$. As B_3 is open there is an $\epsilon > 0$ s.t. the ϵ-ball B_ϵ around p is contained in B_3. But as p is on the boundary of the other two, B_ϵ has non-empty intersection with B_1 and B_2. Now choose $q \in B_1 \cap B_2$ and as $B_1 \cap B_2$ is convex the line L starting in q going to p (but not including p) lies in $B_1 \cap B_2$ as well. But then points on $B_3 \cap L$ lie in all three balls B_1, B_2, and B_3 which proves the statement. □



[3]Let X be a topological space. A collection $\mathcal{U}$ of subsets of X is a cover of X if each point $x \in X$ lies in at least one set $U \in \mathcal{U}$, see [Lee00].

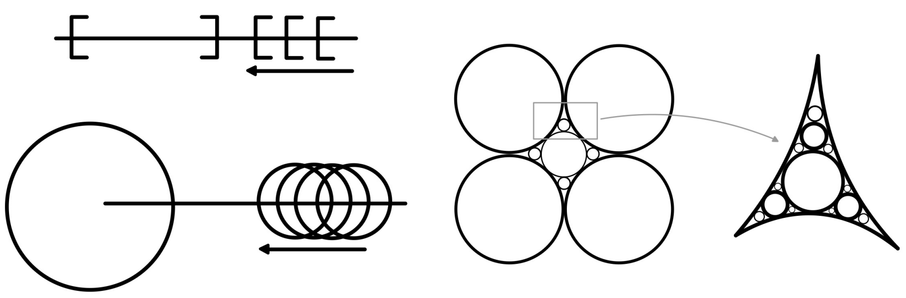


Figure 2.10: Left shows a covering of $[-1,1]$ via $[-1,0] \cup \left\{\left[\frac{1}{n},1\right] \mid n \in \mathbb{N}\right\}$ where the point 0 only lies in one set. This example could serve as an abstraction for the covering of a real line or even for balls in $\mathbb{R}^d$ as through a point on the boundary we can fit this line and find a sequence of balls from the complement of the ball to achieve the same behavior. Right illustrates a covering of $\mathbb{R}^2$ with closed balls, where at first we place balls with equal radius at all integer coordinates intersecting in a single point or not at all. This procedure continues as a sequence where we fit in every uncovered space a ball intersecting again in at most single points with all already existing balls. This then should cover $\mathbb{R}^2$, yet it seems that there are no points lying in more than two balls.

The considerations become more difficult passing to higher dimensions or in the case when we use closed balls for the covering. Figure 2.10 illustrates two examples using closed balls in $\mathbb{R}^d$ where the property that a point lies in $d+1$ balls does not hold. But as we are going to utilize balls[4] in our construction we agree on a list of properties inherent in our construction. We want our cover to be finite. Necessary in practice this also simplifies considerations on closed balls and higher dimensions and it bounds the balls' radii. Such finite subcover can be directly obtained from compact subsets which we generate to surround our points later. Another property is that each ball is necessary in the covering, thus excluding redundant balls or single points.

Definition 11. *A cover $\mathcal{U}$ of X is called* irreducible *if for every set in $\mathcal{U}$ exists a point $x \in X$ not contained in any other element in $\mathcal{U}$.*

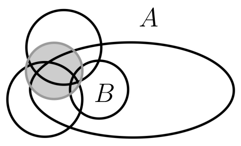
Assumption 1. *Let $A \subset \mathbb{R}^d$, $d \in \mathbb{N}$, be a compact set. Then we assume to have an irreducible open cover $\mathcal{B}$ of $\mathbb{R}^d$ consisting of balls in $\mathbb{R}^d$ with at least one ball $B \in \mathcal{B}$ s.t. $\bar{B} \cap \bar{A} = \emptyset$ and $\bar{B} \subset A$, with $\bar{B}$ denoting the closure of B.*

Thus with a cover meeting Assumption 1 we limit our derivations on a finite (i.e. a finite subcover of $\mathcal{B}$ covering A), irreducible cover of balls with fixed radii fulfilling the desired properties.

Proposition 3. *Let $\mathcal{B}$, A, and $d = 2, 3$ be according to Assumption 1. Let $\mathcal{B}^*$ be a finite irreducible subcover of $\mathcal{B}$ covering A. Then each point in A is contained in at most $d+1$ balls of $\mathcal{B}^*$. The same holds true making $\mathcal{B}$ a closed cover, i.e. $\bar{\mathcal{B}}^* = \left\{\bar{B} \subset \mathbb{R}^d \mid B \in \mathcal{B}^*\right\}$.*

[4]Such sets can be treated experimentally without further obstacles by applying the geometric or combinatorial neighborhoods explained in Example 1 iv) and v).

Proof. First observe that $\mathcal{B}^*$ can be irreducible w.r.t. $\mathbb{R}^d$ but not necessarily w.r.t. A, i.e. yielding the covering of A by intersecting the balls in $\mathcal{B}^*$ with A. As an example consider the gray ball illustrated on the right. However, it can be made irreducible w.r.t. A removing balls from the finite collection $\mathcal{B}$, s.t. it fulfills the property w.r.t. A. Let us assume $\mathcal{B}^*$ to be a covering of A irreducible w.r.t. A.



Let $\mathcal{B}^*$ be an open cover. For $d = 2$ we refer to Proposition 2. In the case $d = 3$ we make use of the follow up Lemma 1. According to this, we find a point p on the intersection of the boundaries of three balls in $\mathcal{B}^*$. With this point we apply the same argumentation used in Proposition 2 and the fact, that the intersection of convex sets is convex.

As the closed balls are generated by taking the closure of the open balls for which the statement holds, it follows for the closed versions in both dimensions as well. □

Lemma 1. *Let $\mathcal{B}$, A, and $d = 3$ be according to Assumption 1 with $\mathcal{B}^*$ a finite irreducible subcover of $\mathcal{B}$ covering A. Let B_1 denote the ball contained in A (Assumption 1). Then there exists $B_2 \in \mathcal{B}^*$ with $B_1 \cap B_2 \neq \emptyset$ s.t. $\partial \bar{B}_1 \cap \partial \bar{B}_2 \not\subset B$ for all $B \in \mathcal{B}^*$.*

Proof. A note on $\mathcal{B}^*$ being irreducible can be taken from the proof of Proposition 3. To continue here assume contrary. Then each intersecting circle, i.e. the non-empty intersection of the boundaries of two balls B_1 and B_2 in $\mathbb{R}^3$, needs to be included in some ball in $\mathcal{B}^*$. As $\mathcal{B}^*$ is finite and all intersections with B_1 need to be included in a ball it follows that B_1 is covered by these balls (which include all the intersecting circles caused on B_1). But this contradicts that $\mathcal{B}^*$ shall be irreducible, thus it is a contradiction. □

The restrictions or properties we set up are suitable for our construction, however, they lead to open questions, which we only conjecture, one concerning the restriction of Lemma 1 to the compact set A and another one allowing more general open sets in arbitrary dimension without all these properties.

Conjecture 1. *Let $\mathcal{B}$ be an open cover of $\mathbb{R}^3$. Then there exists $B_1, B_2 \in \mathcal{B}$ with $B_1 \cap B_2 \neq \emptyset$ s.t. $\partial \bar{B}_1 \cap \partial \bar{B}_2 \not\subset B$ for all $B \in \mathcal{B}$.*

Conjecture 2. *For every cover of $\mathbb{R}^d$, $d \in \mathbb{N}$, consisting of simply connected[5] open subsets of $\mathbb{R}^d$ exists a point in $\mathbb{R}^d$ which is contained in at least $d + 1$ elements of that cover.*

Ideas for a possible proof for the latter statement could be to make use of the nerve complex of the cover, cf. Remark 3, and the argument that the existence of a d-simplex proves the statement, as then $d + 1$ elements of the cover need to have a non-empty intersection. Another approach could be to use the argument of a surrounding compact set once more to make use of a finite subcover, yet meaning to bound the sets. A third way could be to consider the cover as an atlas (yet lacking

[5]A path connected space with trivial first fundamental group is called simply connected, see [Lee00].

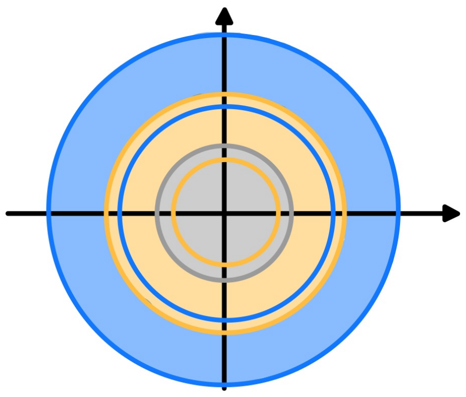


Figure 2.11: A 2-ball in the center and a sequence of $(d-1)$-spheres of some thickness provide a covering of $\mathbb{R}^2$, s.t. each point is in at most 2 sets.

the maps in the charts) describing $\mathbb{R}^d$ as a topological d-manifold and drawing a conclusion on the intersection of charts.

Note that besides the natural treatment of balls (cf. Example 1 iv) and v)) dropping convexity would mean a sequence of blown up $(d-1)$-spheres with increased radii together with a d-ball sitting at the origin yields that a point sits in at most 2 sets for $d > 1$, see an example in Figure 2.11. Leaving out boundedness would even yield that just one set covers the entire space and allowing that the sets consist of single points would provide the possibility to choose each point in space as a ball of dimension 0, closed in the natural topology of $\mathbb{R}^d$. But then each point sits in exactly one ball and we do not obtain any connection via intersection whatsoever.

Now we want to construct a covering such that we only face non-empty intersections of at most $d+1$ clouds, for $d = 2, 3$, and these intersections consist of single points so that we minimize the occasions in which a point sits in $d+1$ clouds. Therefore, we propose the following:

Proposition 4. *The Euclidean space $\mathbb{R}^d$, $d = 2, 3$ can be covered by convex, compact sets, s.t. every $x \in \mathbb{R}^d$ lies in at least one and at most $d+1$ of those sets.*

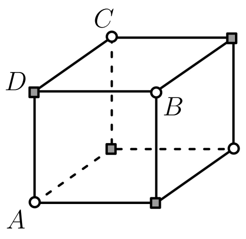
Proof. Let $d = 3$ and $B(c, r) = \{x \in \mathbb{R}^3 \mid \|c - x\|_2 \leq r\} \subset \mathbb{R}^3$ be a closed ball with center c and radius r. Now we claim that the collection $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$ with

$$\mathcal{B}_1 = \left\{ B\left(c, \frac{\sqrt{3}}{2}\right) \mid c \in \mathbb{Z}^3 \wedge \sum_{i=1}^{3} c_i = 0 \mod 2 \right\} \quad \text{and}$$
$$\mathcal{B}_2 = \left\{ B\left(c, \frac{\sqrt{3}}{6}\right) \mid c \in \mathbb{Z}^3 \wedge \sum_{i=1}^{3} c_i = 1 \mod 2 \right\}$$

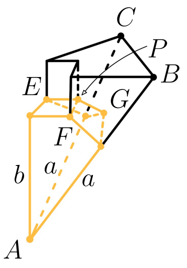
with c_i the i-th coordinate of c yields a covering with the desired property.

Observe that all sets are bounded and convex. W.l.o.g. we may restrict our consideration to the unit cube $[0, 1]^3$ for which we have four balls in each $\mathcal{B}_1$ and $\mathcal{B}_2$ (displayed with gray squares and white circles see inset image). We denote these subsets $\tilde{\mathcal{B}}_1$ and $\tilde{\mathcal{B}}_2$, resp. This is valid as both radii $\frac{\sqrt{3}}{6}$ and $\frac{\sqrt{3}}{2}$ are smaller 1 and no other ball, not in $\tilde{\mathcal{B}}_1$ or $\tilde{\mathcal{B}}_2$, would intersect the cube.

We need to show that each $x \in [0,1]^3$ lies in $a)$ at least one ball and $b)$ at most four balls. $a)$ For the convex hull on centers of $\tilde{\mathcal{B}}_1$ we have $\operatorname{conv}\left\{\{c_i\}_{i=1}^4\right\} \subset \bigcup_{i=1}^4 B\left(c_i, \frac{\sqrt{3}}{2}\right)$. Then, w.l.o.g. we consider the pyramid $ABCD$ with $A = (0,0,0)$, $B = (1,0,1)$, $C = (0,1,1)$, $D = (0,0,1)$, $G = \frac{1}{3}(A+B+C)$, and point $P = \frac{1}{6}(1,1,5)$. The small cube bounded by D and P is included in the ball in $\tilde{\mathcal{B}}_2$ attached to D as P fulfills the equation $x^2 + y^2 + (z-1)^2 \leq \frac{3}{36} = \left(\frac{\sqrt{3}}{6}\right)^2$. It remains to show that the pyramid without the small cube is contained in the 3 balls in $\tilde{\mathcal{B}}_1$ attached to A, B, and C. Or that one of the three similar pieces (one in orange see the inset image) is contained in one ball, here in the ball attached to A. Note that $\overline{AP} = \frac{\sqrt{3}}{2}$ which is the radius of the ball at A. Distances $a = \frac{\sqrt{2}}{2}$, $b = \frac{5}{6}$, $c = \overline{AE} = \overline{AF} = \frac{\sqrt{26}}{6}$, and $d = \overline{AG} = \frac{\sqrt{6}}{3}$ are all smaller then $\overline{AP}$ and the orange piece is convex, thus the piece is contained in the ball attached at A.



Therefore, each of the three pieces is contained in one of the balls at A, B, and C and the small cube in the ball at D. This holds for all pyramids and thus every point in the cube is contained in one ball in $\tilde{\mathcal{B}}_1 \cup \tilde{\mathcal{B}}_2$.



$b)$ All balls in $\tilde{\mathcal{B}}_1$ intersect in only one point, i.e. the center of $[0,1]^3$ for which the claim is fulfilled, as no ball in $\tilde{\mathcal{B}}_2$ reaches it. A point—not the center of the cube—then lies in at most three balls of $\tilde{\mathcal{B}}_1$. As all balls in $\tilde{\mathcal{B}}_2$ have a pairwise empty intersection, such point may lie in at most four balls (an additional ball from $\tilde{\mathcal{B}}_2$) which fulfills our claim.

The case $d = 2$ follows by the same construction by fixation of one coordinate, for instance consider the xy-plane. □

Note that the covering in Proposition 4 is constructed in such a way, that if four balls in $\mathbb{R}^3$ have non-empty intersection, then they intersect in a single point. Figure 2.12 provides an illustration for a unit cell and the attached balls. Thus using it as a scheme to generate clouds from sample points minimizes the possibility of a sample point lying in four balls. Further, we can conclude that in $\mathbb{R}^d$ with $d > 3$ the same construction is not applicable anymore, as we used the half of the diameter of $[0,1]^d$ for the larger balls. But then $\frac{\sqrt{d}}{2} \geq 1$ and thus centers of balls would be included in other balls. In addition, a variable choice for the radii, especially for those of the larger balls is not possible, as otherwise the four large balls would not only intersect in the barycenter of the grid cell but a larger non-empty set. The case $\mathbb{R}^2$ however allows a more general construction with a variable choice for the radii of the balls.

Proposition 5. *The following two constructions allow a covering of $\mathbb{R}^2$ with balls s.t. each point lies in at most three balls:*

i) Consider the tiling[6] $\{3,6\}$ of $\mathbb{R}^2$ with vertices V and edges E. Let radius r be at least the distance of a vertex to the barycenter of an incident triangle and at most $e \cdot \cos(\pi/6)$ with edge length e. Then $\mathcal{B} = \{B(p,r) \mid v \in V\}$ is a covering of $\mathbb{R}^2$.

[6]Regular tiling of $\mathbb{R}^2$ given here with its Schläfli symbol.

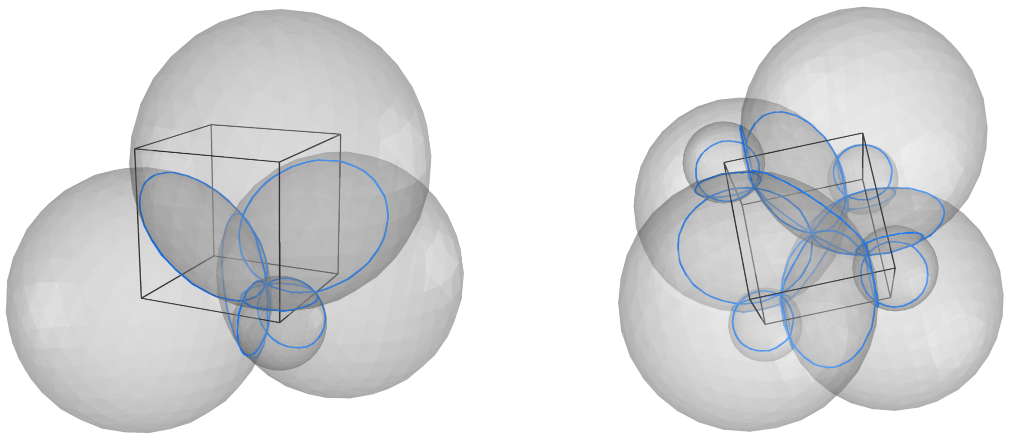


Figure 2.12: Shows a unit grid cell in $\mathbb{R}^3$ and four (left) or eight (right) attached balls to the grid points. The left version with fewer balls shall support visibility. The ball-to-ball intersections are highlighted as circles (🟦) which causes convex subsets whenever two or three balls have non-empty pairwise intersection. The case in which more than three intersect non-empty leads to a single point and this occurs five times, namely one in the center of the grid cell for the four larger balls and the other four occurrences are when three larger and one smaller ball intersect.

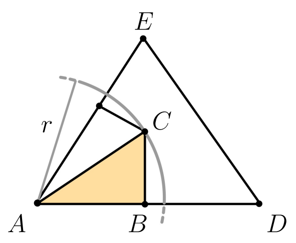
ii) $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$ *with* $r \in \left(\frac{1}{\sqrt{2}}, 1\right), \tilde{r} = \frac{1}{\sqrt{2}} - \sqrt{r^2 - \frac{1}{2}}$ *with*

$$\mathcal{B}_1 = \left\{ B(c, r) \mid c \in \mathbb{Z}^2 \wedge \sum_{i=1}^{2} c_i = 0 \mod 2 \right\} \quad \textit{and}$$

$$\mathcal{B}_2 = \left\{ B(c, \tilde{r}) \mid c \in \mathbb{Z}^2 \wedge \sum_{i=1}^{2} c_i = 1 \mod 2 \right\},$$

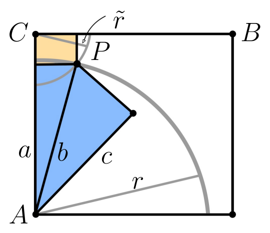
where $B(c, r)$ *denotes the closed ball with center* c *and radius* r *and* c_i *the* i*-th coordinate of* c*.*

Proof.
i) Observe that the upper limit for r gives the distance of a vertex to the opposite edge in the triangle and thus each triangle intersects only with the balls attached to its vertices for all r in the prescribed range. Thus each point in $\mathbb{R}^2$ lies in at most three balls. Further, the triangle ABC (see inset image) lies in $B(A, r)$ and thus by symmetry of triangle ABC in triangle ADE follows that every point in $\mathbb{R}^2$ is covered by $\mathcal{B}$.



ii) We restrict our considerations to the unit square. Observe that only the balls attached to the four corner points intersect $[0, 1]^2$. Show that a) $\mathcal{B}$ is a covering and b) each point in $\mathbb{R}^2$ lies in at most three balls.

a) By symmetry we restrict to the triangle ABC. Note that we can obtain P by $B(A, r) \cap B(C, \tilde{r})$, i.e. $P = \left(\sqrt{d/2}, d/2\right)$ with $d = r^2 - \tilde{r}^2$. Then the square bounded by P and C is contained in $B(C, \tilde{r})$. W.l.o.g.



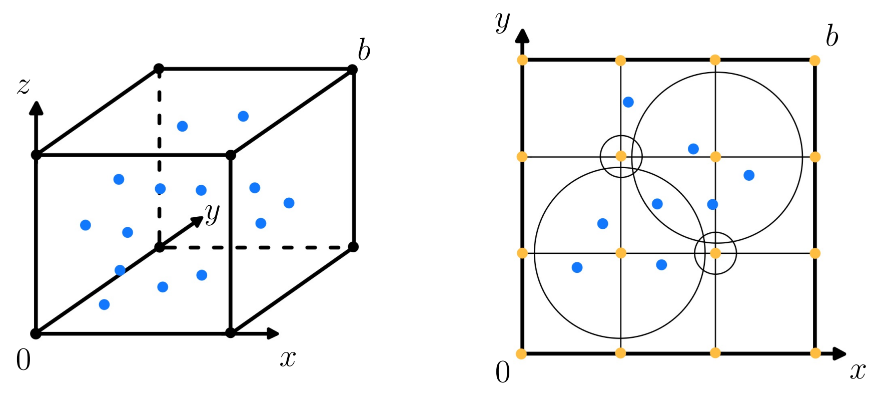


Figure 2.13: Left shows the bounding box covering points in the first quadrant. Right gives an illustration in $2d$ of the scaled grid (grid points in orange) and four exemplary placed balls in the center. The blue points in both images mark the given points which we want to cover with clouds.

we can conclude that the 4-gon in blue, see inset, has (interior) lengths $a = d/2$, $b = r$, and $c = 1/\sqrt{2}$ all smaller or equal to $r \in [1/\sqrt{2}, 1)$ and thus all inside the 4-gon is covered by $B(A, r)$. Hence, $\mathcal{B}$ is a covering of $\mathbb{R}^2$.

b) As $\tilde{r} = 1/\sqrt{2} - \sqrt{r^2 - 1/2} < 1/\sqrt{2}$ we have that $B\left((0,1), \tilde{r}\right) \cap B\left((1,0), \tilde{r}\right) = \emptyset$ and thus each point is in at most three balls. $\square$

Remark 4. *The coverings in the previous proposition are constructed in the way, that the non-empty intersection of three balls is a single point in i) if the radius r is the distance of a vertex to the barycenter and in ii) in general.*

Remark 5. *A covering using a tiling as done in Proposition 5 cannot be translated right away to one in $\mathbb{R}^3$, as we would need a regular tiling of $\mathbb{R}^3$ where there is only one with cubes having a Schläfli symbol $\{4, 3, 4\}$. But in such we would require the radius to be at least $\sqrt{3}/2$ to contain the cubes' center where all eight balls intersect in a singleton. But four balls located at the vertices of each quadrilateral face would have an intersection consisting of more than a single point.*

The generation of clouds from points using the covering in Proposition 4 can be carried out by the following algorithm.

Algorithm 2. *Given points in $\mathbb{R}^3$. At first we translate them s.t. they all sit in the first quadrant of a Cartesian coordinate system. The bounding box of the newly located points then has the origin and a point $b \in \mathbb{R}^3$ as representing points, see Figure 2.13. For a value $s \in \mathbb{R}_{>0}$ we scale the integer grid used in Proposition 4, as well as the radii used for the balls attached to the grid points. We solve for every of the three coordinates individually. If i denotes the i-th coordinate, then we take all natural numbers $j = 0, \ldots, \lceil \frac{b^i}{s} \rceil$, b_i is the i-th coordinate of b, to set coordinate i of a center of a ball to be $s \cdot j$. The radii from Proposition 4 are scaled by s accordingly. Finally, we use the geometric neighborhood (see Example 1 iv)) of these centers and radii to search the given points. Each of such neighborhood, if non-empty, gives rise to a cloud.*

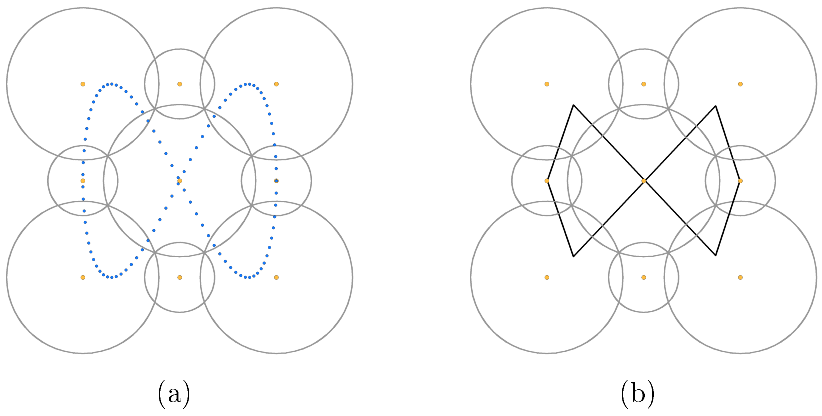


Figure 2.14: Left shows points in blue taken from a curve and placed in a grid with balls of two sizes. Right illustrates the geometric realization of the 1-dimensional cloud complex.

2.2.2 Experimental Results

Up to this point we built a relation between the subsets of points (punched out via a finite covering) encapsulated in a box in $\mathbb{R}^2$ or $\mathbb{R}^3$, depending on whether it represents a 1- or 2-dimensional object in $\mathbb{R}^2$ or $\mathbb{R}^3$, and the geometric realization of the complex created on these subsets. This relation shall serve as a way to encode and control the points, which approximate (or interpolate) an object in question, by a simpler object. However, this has some restrictions to it, for example the necessary combinatorics entailed in a cloud surface, i.e. the clouds themselves need to be designed s.t. no more than three have a non-empty intersection. Something which does not arise yet but already earlier when we introduced them. At this moment it is more so reflected in the way how the balls are arranged (see Propositions 4 and 5) to cut them out. Another restriction or rather consideration we need to face is the existence of points representing curves or surfaces which can be correctly processed into subsets by these balls delivering correct topology in the geometric realizations. Here we need to take into account point distributions respecting densities or geometric objects like the medial axis as further tools.

However, we want to investigate some first examples addressing the generation of clouds from points. We start with the generation from 2-dimensional data, cf. Proposition 5, reflecting curves to emphasize the idea and discuss the difficulty of sampling criteria afterwards. Then we build examples on points representing pseudo-surfaces in $\mathbb{R}^3$ with and without noise. In all experiments we recall that s denotes the grid scale parameter mentioned in Algorithm 2.

Curve Reconstruction Let P be a set of points in $\mathbb{R}^2$ representing a curve in the same space, see Figure 2.14. For this example we do not define a 1-dimensional analogue of the simplicial surface in $\mathbb{R}^3$—our curve parametrized by $t \in [0, 2\pi)$ via $t \mapsto (\cos(t), \sin(2t))$ is not differentiable at $(0, 0)$—, as we do not need it later. Observe further that we restrict ourselves to $\mathbb{R}^2$ and did not consider curves in $\mathbb{R}^3$ using the same construction as in Proposition 4, because the non-empty intersection of three balls shall cause a triangle which is not suitable for a curve. Thus representing curves in $\mathbb{R}^3$ need another treatment.

The 2-dimensional example allows us to represent the balls, the grid points, and

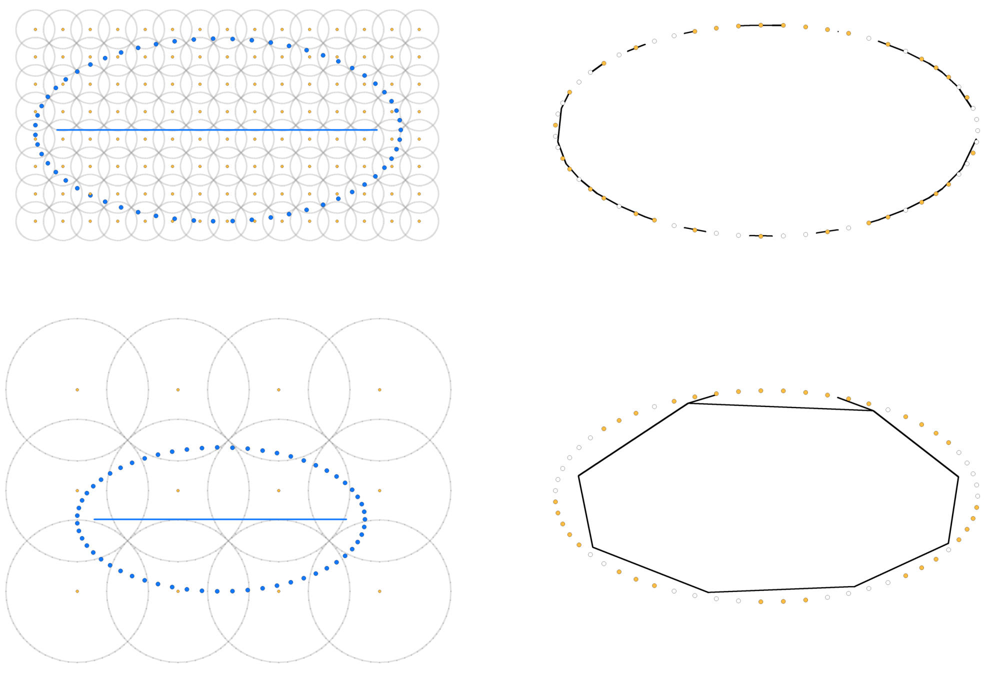


Figure 2.15: Left column shows points in blue taken from an ellipse and placed in a grid with balls with two grid sizes $s = 0.3$ and $s = 1.4$, from top to bottom. The blue line represents an approximation of the medial axis of the original ellipse. Right column illustrates the respective geometric realizations of the complexes, whereas in the top we face missing information and in the bottom false information. Note that all grid balls have one radius, as we set $r = 1/\sqrt{2}$ in Proposition 5 and thus both radii are equal.

how the curve points are covered by those. The right image in Figure 2.14 gives a geometric realization of the 1-dimensional cloud complex caused by the balls. Here the grid scale s accurately reflects the curves' appearance acting as a control structure below the clouds. However, the example also illustrates the sensitivity of point data to grid scale, a circumstance we could theoretically capture with sampling criteria.

Sampling Criteria In our scenario we have balls with up to two different radii in $\mathbb{R}^2$ or $\mathbb{R}^3$ and a grid scale adjusting them. This raises the question whether we can find certain criteria for our points taken from either curves or surfaces, i.e. topological 1- or 2-manifolds (cf. [Lee00]) in such a way that the geometric realization of the resulting cloud complex is homeomorphic to the respective manifold the points were taken from. In the following we consider two sampling criteria. Besides the desired topology a geometric realization of a cloud complex might have, i.e. it is homeomorphic to the manifold the points were taken from, we could either miss information or construct false information, i.e. connections which should or should not be there, respectively (cf. Figure 2.15). Both have an impact on the combinatorics and thus topology. The following criteria are discussed in terms of the possible construction of false information.

A first notion could be a global criterion as mentioned in [BSW09] where we restrict ourselves to d-dimensional manifolds in $\mathbb{R}^{d+1}$ for $d = 1, 2$, as we are considering curves and surfaces at the moment. Suppose our point set P is a so called ε-sampling for an $\varepsilon \in \mathbb{R}_{>0}$, of a manifold M, which means that any point $p \in P$ shall lie in M and for any $x \in M$ we find a $q \in P$ such that $\|x - q\|_2 \leq \varepsilon$. We could think of relating the two ball radii to ε, especially the one we choose, because the smaller one depends on the larger radius. But as we want each of these balls to cut out a component from the manifold which is homeomorphic to a disk of dimension 1 (curves) or 2 (surfaces) we need an additional condition. As otherwise the balls might couple points into a cloud which come from two disconnected components. Thus, in addition to the ε-sampling we demand that $B(x, \varepsilon) \cap M$ is homeomorphic to D^d, for all $x \in M \subset \mathbb{R}^{d+1}$ with $B(x, \varepsilon) \in \mathbb{R}^{d+1}$ closed balls of radius ε around x, and D^d the closed unit disk in dimension $\mathbb{R}^d$. With this given, we choose a radius r to be used in Propositions 4 and 5 fulfilling $r < \varepsilon/2$, so that we can ensure the disk-like subsets to be cut out by the balls. Thus, a grid ball either cuts out nothing from the manifold or one component which means in consequence that points distributed into clouds by such a ball belong to a component, if they are not empty. What is left as a future direction is an investigation to look further into crucial connections and the preservation of topology.

A second criterion could be an ε-sampling[7] as introduced in [AB99]. This is a local criterion as it takes the medial axis of the manifold into account and thus states that P is an ε-sampling of M if for each point $p \in P$ there is a point $x \in M$ s.t. $\|p - x\|_2 \leq \varepsilon \cdot \mathrm{lfs}(p)$. The function lfs describes the local feature size at a point, i.e. the smallest distance from that point to the medial axis. As we need to set a global value for the grid ball radius r we have to find a global value from all the feature sizes. One could be the infimum ρ of all local features sizes. This directly implies $\rho > 0$ and therefore we have to work with smooth manifolds sampled by our points which is another restriction. But taking a value $r < \rho$ (equality could lead to the problem that a closed ball intersects M in disconnected components again), also ensures not to construct false information, yet missing connections might occur more frequently. For instance when we consider the curve in Figure 2.15 for the example with $s = 0.3$ then this satisfies $r < \rho$. What the image already illustrates is that parts in the complex are missing. Even more so as we only related it to ρ, giving a lower limit to all local feature sizes, hence parts with larger distance towards the medial axis are even less sampled and thus will not get caught by the grid balls.

In consequence, both approaches could provide a construction without causing false connections, while missing information remains an open question, even more so for the local criterion.

Cloud Surfaces from Points For an example in $\mathbb{R}^3$ we consider a Sphere model with 10,000 vertices. Two results are illustrated in Figure 2.16 for the two different scales $s = 0.8$ and $s = 1$. The left example contains three boundary components highlighted in the geometric realization of the cloud surface. Here the grid scale $s = 0.8$ and placement of the points in the bounding box do not ensure that the respective three clouds have non-empty intersection (cf. Figure 2.16(a) for an example as there is a portion of blue points missing in the framed rectangle to cause the desired

[7] It is actually introduced as r-sampling which is changed here so that we do not confuse it with the grid ball radius.

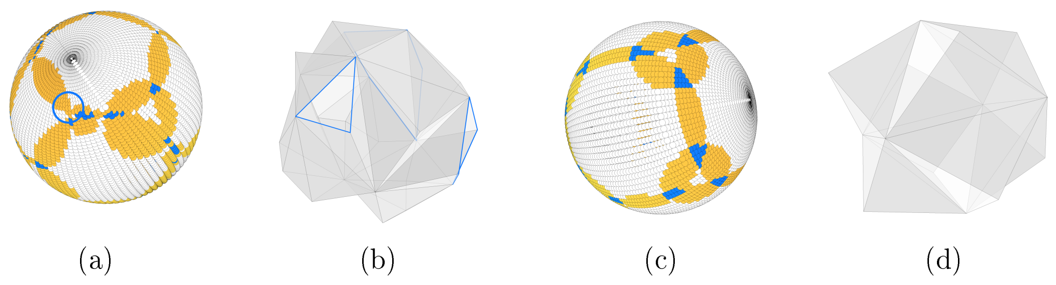


Figure 2.16: Shows cloud generation on $10,000$ points representing a sphere with $s = 0.8$ (a) and $s = 1$ (c) where one example of the missing non-empty intersection of three clouds is framed (🟦) in (a). For the coloring we refer to Example 2. In (b) and (d) are the respective geometric realizations of the cloud complexes. Note that in (b) we have three boundary cycles (🟦), while in (d) we obtain a closed cloud surface.

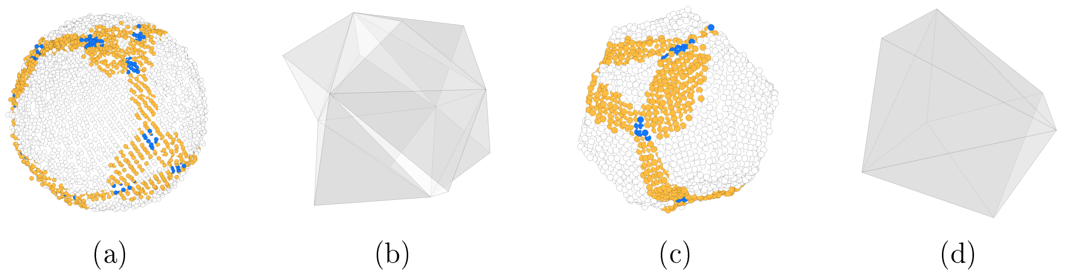


Figure 2.17: (a) and (c) show cloud generations on noisy Sphere and Dodecahedron models, respectively. (b) and (d) illustrate the respective geometric realizations of the resulting cloud complexes.

triangle). The example with $s = 1$ provides a geometric realization of a closed cloud surface.

The approach in $\mathbb{R}^3$ according to Proposition 4 operates with fixed radii instead of the flexibility in $\mathbb{R}^2$, but nonetheless it carries the same obstacles of balancing point set density against grid scale and point set placement inside the bounding box. The dependency on density, grid scale, and placement is a future direction to investigate.

Cloud Surfaces from Noise In the last example we consider the Sphere and Dodecahedron models represented by $10,000$ and $3,842$ points, respectively. We equipped both with Gaussian noise in normal direction with an amplitude of 50% (Sphere) and 25% (Dodecahedron) of the average neighbor distance taken as averaged sum over all points and their 12 nearest neighbors, see visual results in Figure 2.17. The grid scales were set to $s = 1$ for the Sphere and $s = 1.9$ for the Dodecahedron. Note that the second example relates to the paragraph discussing the noisy Dodecahedron in Section 4.4.2 in the second chapter. However, here we had to increase the density thus having more points available to construct the cloud complex.

Both examples show that deviations in the points caused by noise do not necessarily affect the outcome and therefore yield the desired topology in the geometric realization of the cloud complex. The grid balls themselves offer some spatial freedom in which the points possibly lie. As mentioned in the previous paragraph, the contribution and tuning of the grid scale and model placement in space are future directions. For this experiment we received values from a run through, were we gave a range for s with some stepsize and checked whether the geometric realization of the cloud complex is homeomorphic to a topological 2-manifold (cf. [Lee00]). Therefore, we obtained $s = 1.9$ in the second example as a valid grid scale. Furthermore, rotating the model might benefit the outcome. Note that translations and scalings will not have an impact as we proceed with all considerations with the models' bounding box, yet rotations change spatial positions inside this box. The Sphere in our example was rotated beforehand providing us a valid grid scale $s = 1$.

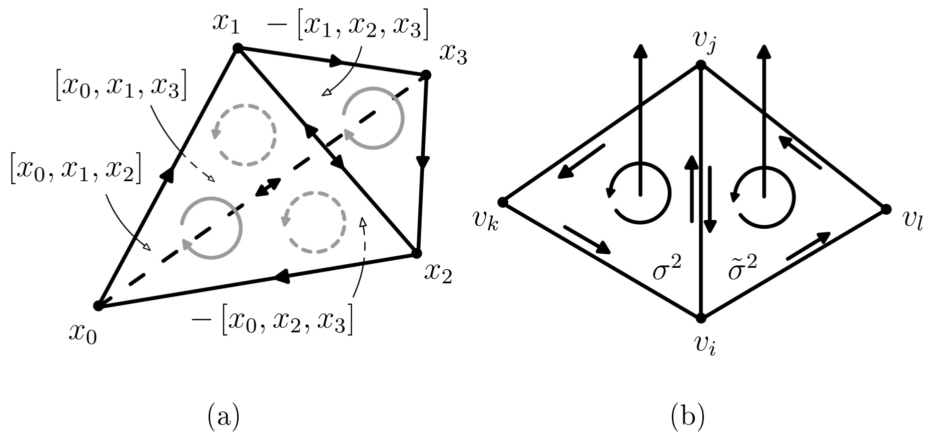


Figure 2.18: Left shows an example of simplices respecting their orientation and the induced orientations of their faces by the boundary operator. Right illustrates two adjacent triangles with their orientations and the normals obtained by Equation (2.2).

2.3 Cloud Dynamic

Now we want to use the geometric realization of a cloud surface, perceivable as a skeleton, to introduce some sort of point dynamic. Here we make use of information contained in an abstract simplicial complex, more so we ask for orientability. From the combinatorics of clouds we further derive influence areas, i.e. a distinction of individual point updates respecting the clouds they belong to. This leads to an algorithm allowing reversible point updates and a discussion of experiments in Section 2.3.2.

2.3.1 Surface Orientation and Update Schemes

Following the discussion about the generation of cloud surfaces, we now want to put them to application. The cloud surfaces' underlying pseudo-surface acts as a control structure where positional variations of the vertex positions shall cause the points in the respective clouds to change their positions too. Each triangle can be treated as a local coordinate system with the third direction caused by a natural choice of a normal for that triangle.

Definition 12. *Given a pseudo-surface $|K|$ in $\mathbb{R}^3$. Then $n : |K| \setminus |K^{(1)}| \mapsto \mathbb{S}^2$ with $p \mapsto n(p)$ assigns to each point p a so called* normal *(up to sign), s.t. $n(p)$ is orthogonal to the triangle $|\sigma^2|$ it belongs to.*

Observe that we only assign normals to points sitting in the interior of triangles, thus we do not have normals along edges or at vertices at all, as there is no natural unique (up to sign) choice compared to the interior of triangles.

Changing a vertex position let the normals of incident triangles undergo some rotations. Hence we agree on an orientation of our pseudo-surface. The following definition is taken from [Toe17] working with simplicial manifolds, yet it applies to our abstract simplicial surfaces as well. For an illustration see Figure 2.18(a) also showing an example for the orientation of simplices as a recreation of an example in [Toe17].

Definition 13. *The sequence*[8] *of abstract vertices in a d-simplex* $\sigma^d = \{v_0, \ldots, v_d\}$ *defines its* orientation. *All even permutations*[9] τ *on the index set define the same orientation* $\left\{v_{\tau(0)}, \ldots, v_{\tau(d)}\right\}$ *of* σ^d. *Changing two abstract vertices gives the opposite orientation of* σ^d, *denoted* $-\{v_0, \ldots, v_d\}$. *Each orientation* $\{v_0, \ldots, v_d\}$ *of* σ^d *induces via a boundary operator*

$$\partial \{v_0, \ldots, v_d\} = \sum_{i=0}^{d} (-1)^i \{v_0, \ldots, \hat{v}_i, \ldots, v_d\},$$

where $\hat{v}_i$ *leaves out entry* v_i, *an orientation on the faces of* σ^d. *Two d-simplices are* neighbors *if they share a* $(d-1)$*-face. Two d-simplices have* coherent orientation *if they have induced opposite orientations on their shared face via the boundary operator.*

As the orientation of simplices are defined on the abstract vertices without further requirements, we deviate from the definition applied to combinatorial (or simplicial manifolds) as provided in [Toe17] and define it directly on abstract simplicial complexes from which the abstract simplicial surfaces inherit that property.

Definition 14. *An abstract simplicial complex is* orientable *if all* 2*-simplices can receive an orientation s.t. all neighbored* 2*-simplices have coherent orientation.*

Proposition 6. *Given a pseudo-surface* $|K|$ *in* $\mathbb{R}^3$ *and* K *an oriented abstract simplicial complex. Then for the oriented* 2*-simplex* $\sigma^2 = \{v_i, v_j, v_k\} \in K$ *the normal* $n(p)$ *for all* $p \in |\mathring{\sigma}^2|$ *can be given by*

$$n(p) := \frac{(|v_j| - |v_i|) \times (|v_k| - |v_i|)}{\|(|v_j| - |v_i|) \times (|v_k| - |v_i|)\|_2}, \tag{2.2}$$

with $\times$ *denoting the cross product (cf. Figure 2.18(b)) and* $|v.|$ *the geometric realization of* $v.$.

Proof. As we only ask for a normal attached to the interior points of a triangle which is perpendicular to that triangle it suffices to take the normalized cross product of its edges, as they span the 2-dimensional affine subspace representing the points in the triangle. Thus we get a unique normal up to sign. □

Now we introduce a time parameter $t \in [0, 1]$ and denote P_t the points (Definition 2) and $|K_t|$ the pseudo-surface (Definition 10) w.r.t. time t. Note that an abstract vertex set, the abstract simplicial complex or a cloud set do not need an additional subscript, as they only provide references and do not change. As a reminder, we are interested in how the points P change w.r.t. the positional variation of $|K|$. Such motion can be carried out via a map

$$\varphi : |K^{(0)}| \to \mathbb{R}^3, v \mapsto \varphi(v), \tag{2.3}$$

acting on the vertices of $|K|$.

[8]Rather then using common sequence notation we stick to the curly brackets and emphasize whenever the orientation is of importance.

[9]See [Toe17] for a brief introduction.

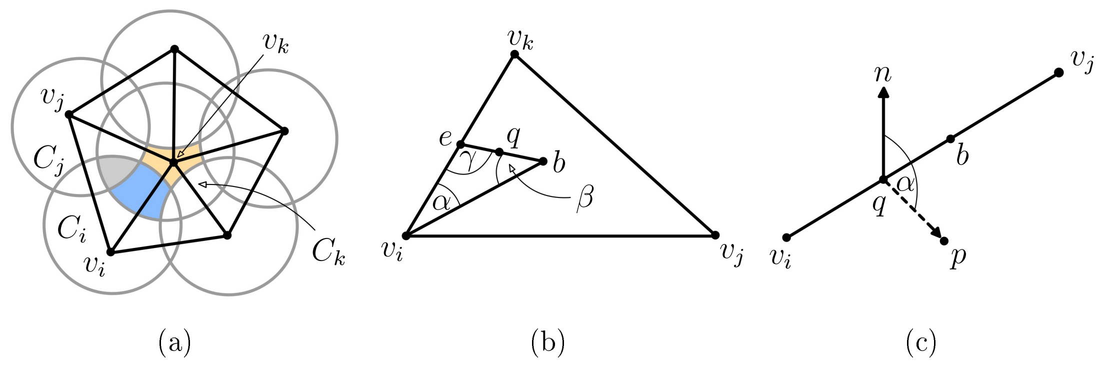


Figure 2.19: (a) Shows the influence determination. A point in the intersection of three clouds C_i, C_j, C_k in the gray region is represented by the one triangle. A point only in the intersection of two, say C_i, C_k in the blue region is represented by the two triangles incident to that edge. The remainder in the orange region respects all incident triangles. (b) Illustrates the calculation of ratios of the projected point q w.r.t. the triangle $|\{v_i, v_j, v_k\}|$. Here we are interested in the ratios $\|v_i - e\|_2 \,/\, \|v_i - v_k\|_2$ and $\|b - q\|_2 \,/\, \|b - e\|_2$ with triangle barycenter b and intersection e of the edge and ray starting in b towards q. (c) Calculation of ratio $\|b - q\|_2 \,/\, \|b - v_i\|_2$ to find the updated projection $\tilde{q}$. Note that we use angle α between the edge normal and vector $q - p$ to get the updated point $\tilde{p}$.

Assumption 2. *Let $|K_{t=0}|$ be a pseudo-surface with $K_{t=0}$ oriented and $|K_{t=1}|$ obtained by $\varphi(|K_{t=0}|)$, from Equation (2.3). Note that $K_{t=1}$ has the same orientation as $K_{t=0}$, because it was defined on the underling abstract simplicial complex and the realization does not change. We assume that φ preserves d-dimensional simplices for $d = 0, 1, 2$ and the sign of normals (Equation (2.2)).*

To investigate how the points change under φ we first find out how points are influenced. The idea is that when a cloud changes its position controlled by the respective vertex in the pseudo-surface the point inside the cloud should move as well. Such change could mean a translation and rotation of the local coordinate system the point lives in as well as new coordinates in that system. As we have a natural choice for normals inside a triangle but not on edges or vertices of a pseudo-surface, we are free to propose certain choices for these (done in the upcoming algorithm). The points themselves are updated w.r.t. the clouds they belong to, see Figure 2.19(a). That means a point in the intersection of three clouds—reflecting a triangle—is represented and influenced by that triangle only. Next, a point in the intersection of two clouds excluding the triangle points—reflecting an edge—is influenced by the triangles incident to that edge which are at most two. The remaining points solely sitting inside a cloud—reflecting a vertex—are influenced by all triangles attached to that vertex. We propose the following algorithm as an example of an update scheme for a single point.

Algorithm 3. *At first we set the normals at vertices and edges of the pseudo-surface to be the weighted mean normals of incident triangles to the vertex or the edge in question, resp. As weights we choose the areas of the triangles. All of the following instances are considered at time $t = 0$, whereas an object o at time $t = 1$ is denoted $\tilde{o}$.*

For the movement of a point w.r.t. to a triangle (sitting inside three clouds) we do the following (cf. Figure 2.19(b)). Let $p \in \mathbb{R}^3$ *and* $|\sigma^2| = |\{v_i, v_j, v_k\}|$ *be a triangle with* n *the constant normal for all points in* $|\overset{\circ}{\sigma^2}|$ *according to Equation (2.2). We are interested in* $\tilde{p}$*. Therefore, we project* p *orthogonally onto* $|\sigma^2|$ *getting* q*. With this we get the edge with endpoints, say* v_i *and* v_k*, which intersect the ray starting in* b *in direction* q *in* e*. Then we take the ratios*

$$\frac{\|e - v_i\|_2}{\|v_k - v_i\|_2} \quad \textit{and} \quad \frac{\|q - b\|_2}{\|e - b\|_2} \tag{2.4}$$

with b *the triangle's barycenter. Note that we can obtain these ratios by the law of sines as*

$$\|e - v_i\|_2 = \|v_i - b\|_2 \sin(\gamma)^{-1} \sin(\beta) \quad \textit{and} \quad \|e - b\|_2 = \|v_i - b\|_2 \sin(\gamma)^{-1} \sin(\alpha)$$

with $\alpha, \beta,$ *and* γ *the angles at the vertices of the triangle spanned by* $v_i, b,$ *and* e*, resp. Now we determine* $\tilde{q}$ *preservering the ratios in the triangle* $\widetilde{|\sigma^2|} = \varphi(|\sigma^2|)$*. Afterwards we obtain*

$$\tilde{p} = \tilde{q} + \omega \|p - q\|_2 \, \tilde{n}, \tag{2.5}$$

where $\omega \in \mathbb{R}$ *is some weight influencing the height above the triangle. A natural choice could be to set* $\omega = \mathit{area}(|\sigma^2|) / \mathit{area}\left(\widetilde{|\sigma^2|}\right)$*, i.e. the ratio of areas of the triangle before and after the movement.*

Similarly to the previous case, we move a point w.r.t. to an edge (sitting inside two clouds), see Figure 2.19(c). At first we obtain the projection q *of point* $p \in \mathbb{R}^3$ *onto the line described by the barycenter* b *and unit direction* u *obtained by the edge with endpoints* v_i*,* v_j*. Then we select the edge endpoint lying on the same side compared to* b *as* q*, say* v_i*, and obtain the ratio* $\|q - b\|_2 / \|v_i - b\|_2$*. The projected point* $\tilde{q}$ *then preserves the ratio on the edge with endpoints* $\tilde{v_i}$*,* $\tilde{v_j}$*. First we express* $\tilde{p}$ *according to Equation (2.5), with a possible weight* ω *to be the length of* $\widetilde{|\sigma^1|}$ *divided by the length of* $|\sigma^1|$*. But as it does not necessarily lie in direction of the edge normal* n*, we further rotate it by* $-\alpha$ *around the direction* u*, where* $\alpha = \angle_u(n, q - p)$*.*

To change the point position of $p \in \mathbb{R}^3$ *w.r.t. a vertex* v *(lying in a single cloud), we apply a translation from* v *to* $\tilde{v}$ *and rotation sending the vertex normal* n *to* $\tilde{n}$*.*

Observe that with the scheme in Algorithm 3, which relies on intrinsic ratios and a simplex and orientation preserving map φ (Equation (2.3)), the point position updates are reversible.

2.3.2 Experimental Results

To exemplify Algorithm 3 we chose two sets of points, i.e. a planar quadrangular region consisting of 4,225 points and a Cube model with 1,538 points. Clouds are generated via Proposition 4 with grid scales $s = 0.25$ (Quadrilateral) and $s = 1$ (Cube). Illustrations can be seen in Figures 2.20 and 2.21. Edge and vertex normals are estimated via weighted means of adjacent triangle normals. For the Quadrilateral we manually changed all cloud points which do not appear at the boundary and lifted them up. The points in the respective clouds changed accordingly. For the Cube model we normalized all cloud points in the geometric realization of the cloud complex so that they live in the unit 2-sphere and consequently the corresponding clouds disassemble.

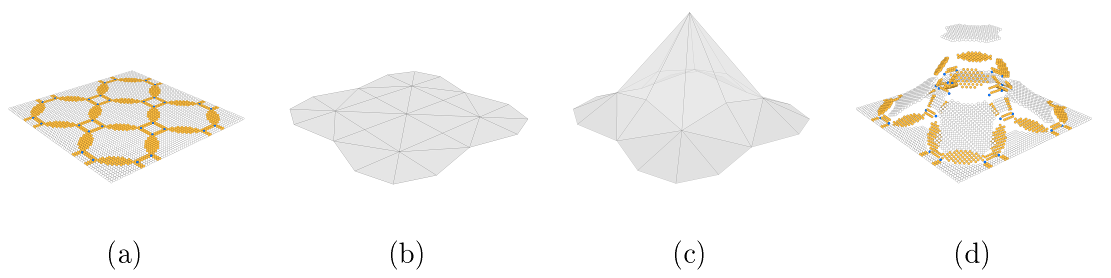


Figure 2.20: Shows clouds on planar points (a) with the corresponding complex illustrated in (b). In (c) a change of cloud point coordinates is applied and thus the resulting movement of points in the clouds is shown in (d). Coloring of cloud points is according to Example 2.

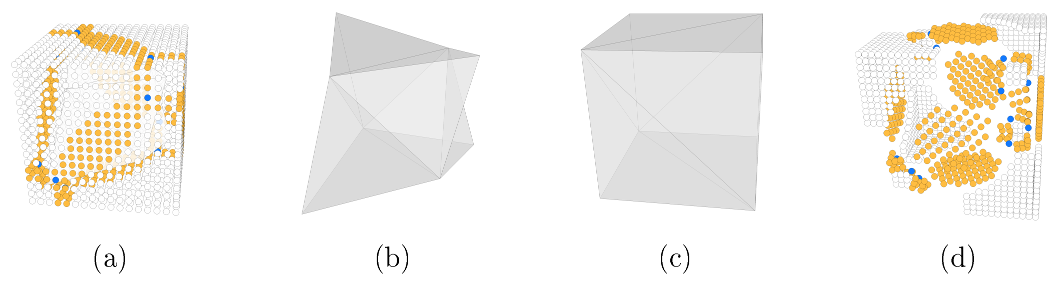


Figure 2.21: Shows clouds on points taken from a Cube model (a) with the corresponding complex illustrated in (b). In (c) a change of cloud point coordinates is applied, i.e. all got normalized to sit in the unit 2-sphere, and thus the resulting movement of points in the clouds is shown in (d). Coloring of cloud points is according to Example 2.

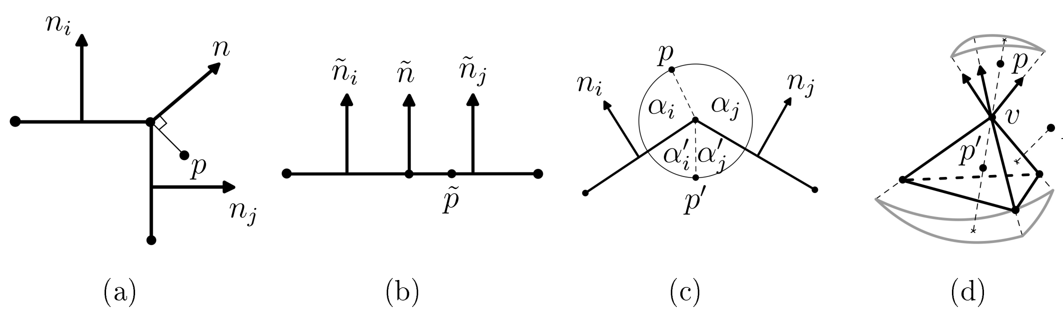


Figure 2.22: (a) Illustrates a side view of two connected triangles with their normals n_i and n_j, the mean normal n used at the edge and p which is going to be updated w.r.t. the edge. (b) After we have changed the triangle vertices all new normals point up and $\tilde{p}$ lies inside the triangle. (c) Shown is the side view of two connected triangles with two reference angles, i.e. dihedral angles between triangles and the plane spanned by the edge and the connecting vector of p and the edge. Here the references are chosen depending whether the point lies outside (p) or inside (p') assuming our surface is closed. (d) Shown are three triangles attached to vertex v with the arrows indicating the triangle normals attached to v. Thus p and p' lie in the cones spanned by the normals or triangle edges, respectively. These cones intersect with the unit 2-sphere as well as the rays from v through p and p' where the latter could be expressed with barycentric coordinates w.r.t. spherical polygons caused by the cones. The treatment of p'' not sitting in such cone remains an open question.

The algorithm gives a first way to incorporate reversible cloud dynamics. However, especially in Figure 2.20 it can be seen that a more continuous transition would be desirable and natural. Note that points only influenced by one triangle as respective tangent space receive a somehow understandable update. The problem occurs whenever multiple triangles influence updates whereas points sitting only in one cloud need to be represented by all triangles attached to the corresponding cloud point. At the moment we only treat those by translation and rotation so that both approximating spaces represented by cloud points and estimated normals are transferred to each other and thus the points in the clouds accordingly. As these points make up a majority compared to edge or triangle instances in our experiments, the results appear rather disassembled. Which is not only visually unsatisfying but it complicates the notion of vicinity even more.

In addition to this, we might want to control the updates more naturally w.r.t. where the points lie. For instance in Figures 2.22(a) and 2.22(b) the illustrated point gets moved to lie inside the triangle. From a current standpoint this is not favorable as we would assume that ratios of angles a point has towards the two respective triangles (if we consider an edge point) be preserved accounting for a more natural dynamic. Further, as a consequence points lying inside the space reflected by a triangle would stay in that space, which is not the case at the moment.

An open direction and idea is to incorporate so called reference angles to account for the angle ratios. For a point which needs an update according to an edge we consider the configuration displayed in Figure 2.22(c). Here a point causes two dihedral angles and their sum. Assuming that our surface is closed and oriented, we could decide whether a point lies inside or outside w.r.t. the triangle normals and further we know which angles we need to use. For instance, a point p outside the surface has angles α_i and α_j and thus one ratio $R_i = \alpha_i/(\alpha_i + \alpha_j)$ it has towards triangle $|\sigma_i^2|$, and the other ratio analogously. After changing cloud point positions of the geometric realization the triangle normals update accordingly. Then $\tilde{\alpha}_i = R_i \cdot \tilde{\alpha}$, with $\tilde{\alpha}$ dihedral (outer) angle spanned by $|\tilde{\sigma}_i^2|$ and $|\tilde{\sigma}_j^2|$, is the new angular position for the updated point ($\tilde{\alpha}_j$ analogously). Angular means that we further could incorporate a change in radial direction w.r.t. some ratio respecting edge lengths or triangle areas as weights.

A similar route could be considered for a point which sits in only one cloud and gets thus influenced by multiple triangles. In Figure 2.22(d) we have an example with three different locations for such a point p. Suppose such a point lies in the cone spanned by either the triangle normals or the triangle edges (depending on whether it lies inside or outside the closed oriented surface), and assume for a moment we focus on the first case (as the second is analogue). Then we attach a unit 2-sphere at cloud point v so that the normals spanning the cone give us points on that sphere, and connecting them in the order of the adjacent triangles provides a spherical polygon. Now the line through v and p gives us another point on the sphere for which we seek (spherical) barycentric coordinates inside the spherical polygon. Then after an update these barycentric coordinates should determine at least the angular placement on the 2-sphere in the cone spanned by the updated triangle normals. Besides the details for both update cases[10] the treatment of a point not sitting in any cone is open, and thus both are future directions.

[10] These are point updates corresponding to an edge (influenced by 2 triangles) and the other to a cloud point (influenced by more than 2 triangles).

2.4 Conclusion and Further Research

In this section we investigated possible ways of generating cloud surfaces either synthetically from surfaces or from points in space. With these at hand we considered such cloud surface as a control structure to introduce point updates respecting changes in the control structure. Cloud surfaces are one way to think of an atlas-like structure set up on points, not necessarily putting emphasis on local neighborhoods around a point or respecting all at once in a global manner. Their advantage lies in the strengths of thoroughly investigated (abstract) simplicial surfaces, yet causing some restrictions. Some of those provide future directions for research comprised in the following list.

- Currently the generation of cloud surfaces from pseudo-surfaces can be used as a starting point to have cloud surface examples to work with. However, as such points and clouds entail the desired structure we could think of using the procedure in terms of remeshing into polygonal meshes (triangular, quadrilateral, etc.) of certain quality.
- For both, the sphere models introduced in Section 2.1.1 and their contained parameters, it could be investigated how they influence generation results.
- In Section 2.2.1 we introduced coverings suitable for the determination of cloud surfaces from points. As mentioned above, the underlying structure is a simplicial surface which forces restrictions on the combinatorics of cloud complexes. The generation from points now suits this direction, however here an in-depth investigation of what we can expect and achieve from such ball configurations in $\mathbb{R}^2$ and $\mathbb{R}^3$ remains open, yet first steps were taken to consider certain sampling criteria.
- Again about the determination of cloud surfaces from points (cf. Section 2.2.1) an open direction is to extend those in terms of adaptivity to work in multiscale scenarios. Especially in situations with non-uniform samplings individual parts could be processed with other grid-scales. However, a proper gluing along the boundaries of multiple scales preserving the topology is a challenging question.
- In Section 2.3.1 we introduced update schemes leading to a reversible dynamic. In the experiments we started a discussion about a more continuous point update to reflect dynamics more fluently. At the moment we have direct updates at time steps $t = 0$ and $t = 1$ where we could also think of continuous or even differentiable functions for the dynamics (preferably reversible).
- The cloud dynamic assumes a closed cloud surface to work with. Here the treatment of boundaries, especially necessary for points relating to edges, could be interesting, as it opens a possibility to cut and flatten surfaces and with them the respective points.

Another way to think of a set system applied to points, yet entailing some structure, is in some sort the dual—here to cloud surfaces—perspective discussed in the next chapter.

Chapter B

Surface-like Structures

Having the structure of simplicial complexes at hand and using them as an underlying basis provides the possibility to take over a well studied machinery onto subsets of points, i.e. clouds. However, the clouds themselves may carry data from 2-dimensional surface patches and thus letting each of them get represented by a 2-dimensional entity in a complex seems natural. A motivation to further investigate such surface-like structures came from segmentation results of point sets, see Figure 2.1.

Under certain conditions such a structure (or complex) which is at most of dimension 2 and shall reflect manifold behavior suggests to be dual[1] to cloud surfaces. A necessary requirement for duality then is the restriction that no more than three clouds cause a vertex in the structure. This is because in the dual this vertex need to represent a triangle as it then belongs to a cloud complex. Here we already deduce that we do not want to rely on duality, rather we want to achieve it under certain circumstances.

Being dual or not, another viewpoint to encode such structures could be a CW complex. But in them the basic object is a k-cell, $k \in \mathbb{N}_0$, which needs to be homeomorphic to the open k-dimensional unit ball, see [Toe17], thus, the boundary of its closure is homeomorphic to the unit sphere of dimension k-1. This might be plausible whenever a segment seems to have one bounding cycle, but we would like to allow multiple (disjoint) cycles to bound a segment (see Figure 2.1(b)) yet violating the definition of a CW complex.

Therefore, we are going to discuss a way of thinking about surface-like structures acting dual under certain conditions to cloud surfaces and present a generation of such structure utilizing point set segmentation and how they act beneficial in a subsequent application, i.e. simplification.

Parts of the work that follow are published in [SZP20], whereas Section 3 and the paragraph about general faces in Section 4.4.2 are extensions. Especially the latter shall provide a solution to the third comment risen in the further research in [SZP20] addressing the treatment of planar patches which are not star convex. Further, some results of the paper have been presented as a poster at the International Geometry Summit 2019 in Vancouver, Canada and have been published in the corresponding poster proceedings, see [SZP19]. Changes were made to Figures 4.6 (except the center one), 4.9 (only our examples), 4.10, 4.11, and 4.14 in visual representation to match with the coloring used throughout this thesis, as well as minor textual

[1]In the context when identifying 0- and 2-dimensional entities, and vice versa.

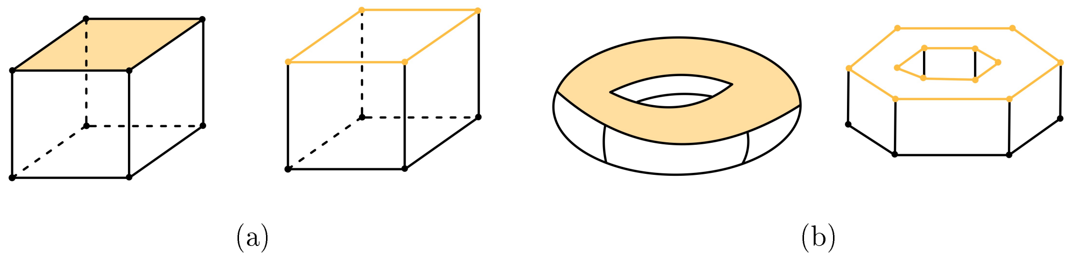


Figure 2.1: Each of both examples shows a segmentation of a point set into colored (disjoint) segments on the respective left and a complex obtained from the segmentation on the right. The complex has a vertex where more than two segments are close by and an edge where exactly two are close by. The colored cycles on the respective right illustrations represent the boundaries of the segments on top.

changes. The experimental results and discussions were split and placed in the corresponding sections dealing with segmentation and simplification, respectively. A further discussion about the interpretation of parameter η was placed in the experimental results section instead of an appendix.

Note in Section 3 we elaborate on another definition of a complex where we use the terminology of clouds to build such complex. However, throughout Section 4 we use the notion of *region* instead of *cloud* to address a subset of the given points to represent a segment as a 2-dimensional entity, because we want to stick to the terminology used in [SZP20]. Another reason is, that we deal with a disjoint segmentation in Section 4 for which a cloud complex would only include 0-simplices but none of higher dimension with its current definition, see Definition 6 and Remark 3 i). Further we use the strength of a cloud complex on the neighborhood of regions to obtain the desired new complex.

3 Description of Surface-like Structures

For a given set of clouds $\mathcal{C}_I$ on an index set I, we build

$$I^* = \left\{ \tilde{I} \subseteq I \mid \# \left\{ \tilde{I} \right\} \geq 3 \wedge \cap_{i \in \tilde{I}} C_i \neq \emptyset \right\}$$

which is a set of all subsets of I for which the intersection of the respective clouds is not empty. From this we keep

$$\mathcal{I} = \left\{ \tilde{I} \in I^* \mid \tilde{I} \nsubseteq I' \quad \forall I' \in I^* \setminus \left\{ \tilde{I} \right\} \right\}, \tag{3.1}$$

i.e. the inclusion maximal subsets. From now on we use $\mathcal{I}$ and $\mathcal{I}_J$ for an index set J interchangeably with emphasis on the latter if the indexing is important. Observe that $\mathcal{I}$ is a cloud set built on I and together with a cloud vertex map $\Psi : \mathcal{I}_J \to \mathcal{V}_J$ we obtain the cloud complex K w.r.t. $\mathcal{I}$. Here we denote $\mathcal{E} = K^{(1)}$ to be the set of *abstract edges*.

By an abuse of notation (orientation of simplices) we are going to reflect the boundaries of a 2-dimensional object as oriented d-simplices. Every cloud $C_i \in \mathcal{C}_I$ causes such a 2-dimensional object (which we call an *abstract face*) and one boundary component can be represented by a sequence of abstract vertices in $\mathcal{V}_J$, i.e. $\{v_{j_0}, \ldots, v_{j_n}\}$ for $n = 2, 3, \ldots$ s.t. $I_{jl} \in \mathcal{I}_J$ and $\left\{ v_{jl}, v_{j(l+1) \mod n} \right\} \in K^{(1)}$ for $l = 0, \ldots, n$. Further, we require the induced subgraph[2] on these vertices from the graph caused by $\mathcal{V}_J$ and $\mathcal{E}$ to be a cycle[3], so that we can ensure that each boundary component has the correct topology, i.e. it is homeomorphic to the 1-sphere. The set of cycles bounding the abstract faces is denoted $\mathcal{F}$[4]. Then we set up a complex K' as

$$K' = \mathcal{V}_J \cup \mathcal{E} \cup \mathcal{F}. \tag{3.2}$$

See Figure 3.1 for an example. Observe that we can read off information of the contributing clouds in all instances sitting in the three subsets $\mathcal{V}_J, \mathcal{E}$, and $\mathcal{F}$. For instance, intersecting clouds (in cloud set $\mathcal{I}$) of the two abstract vertices sitting in an abstract edge give the indices which refer to the original clouds (or one cloud) having this edge as a boundary component. We could do the same for objects in $\mathcal{F}$. To look for those cycles corresponding to cloud C_i we need to find all cycles in $\mathcal{F}$ where i appears in every cloud (in cloud set $\mathcal{I}$) corresponding to the respective vertex in the sequence.

The complex K' (Equation (3.2)) is again an abstract combinatorial object. Now we could think of relating it to a topological 2-manifold in the same way we did to get a cloud surface in Definition 8. We recall that neighborhoods around points in a 2-manifold are Euclidean of dimension 2. If we have interior points of an abstract face—and it is again not defined what interior means, but it should be considered on an abstract level—, we assume them to have the desired property. For abstract edges we require each edge to be the boundary component of at most

[2]An *induced subgraph* is a subgraph obtained by deleting a set of vertices, see [Wes01].

[3]A *cycle* is a graph with an equal number of vertices and edges whose vertices can be placed around a circle so that two vertices are adjacent if and only if they appear consecutively along the circle, see [Wes01].

[4]One entry does not reflect one abstract face in general. The set contains boundaries so that multiple elements in here might bound an abstract face with possible holes.

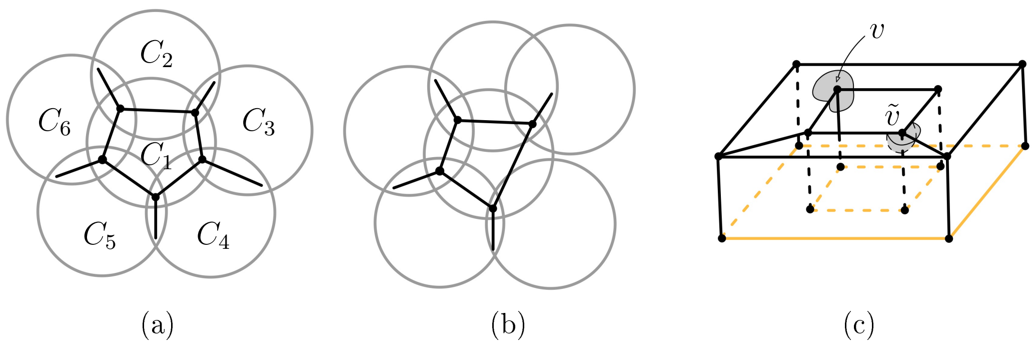


Figure 3.1: *a*) Shown are six clouds and drawings of abstract vertices and edges. Two of such vertices are for instance $\{1, 2, 3\} \subset I$ and $\{1, 3, 4\} \subset I$ as their respective clouds have non-empty intersection. Both cause an edge running between C_1 and C_3. To get the boundary cycle of the abstract face we could (for example) start at the vertex belonging to $\{1, 2, 3\}$ and continue to the vertex representing $\{1, 3, 4\}$ as they are connected. Continuation of this process provides a sequence as bounding cycle of the abstract face for cloud C_1. *b*) Sketches an example in which an edge for the abstract face for cloud C_1 bounds only one face. *c*) Illustrated is a piecewise planar approximation of the torus as a possible geometric realization of a surface-like structure. The top consists of two faces bounded by two cycles, one of each with highlighted vertices v and $\tilde{v}$ having sequences of cycles with the desired property (see Conjecture 3). At the bottom there is an example of a face bounded by more than one cycle displayed in orange.

two abstract faces, i.e. they are present in two cycles at the most. What remains is to treat neighborhoods around abstract vertices. We conjecture a definition similar to Definition 7, with an example in Figure 3.1(c).

Conjecture 3. *A complex K' according to Equation 3.2 is called a* surface-like structure *if*

i) each abstract vertex is in at least one cycle bounding an abstract face,

ii) each abstract edge is contained in at most two cycles bounding abstract faces, and

iii) for each abstract vertex all cycles containing it can be arranged in a sequence, s.t. consecutive entries in the sequence have an abstract edge in common.

Conjecture 4. *A closed cloud surface K on $\mathcal{C}_I$ is dual*[5] *to a surface-like structure K' on $\mathcal{C}_I$ if every abstract edge in $\mathcal{E}$ is contained in exactly two cycles in $\mathcal{F}$, each abstract face is bounded by one cycle, and $\#\left\{\tilde{I}\right\} = 3$ for all $\tilde{I} \in \mathcal{I}$, i.e. each inclusion maximal subset of I is of size* 3.

The terminology in Conjecture 3 will not get investigated further, as we set our focus on the generation and application of the complex derived in Equation (3.2), yet it marks a possible direction for future work, whether it is well defined, carries desired 2-manifold properties, and could be of potential interest.

[5]Dual means that we identify 0- and 2-dimensional objects and vice versa. The 1-dimensional entities update accordingly.

4 Segmentation and Simplification

In the following section we discuss a method to generate a complex as given in Equation (3.2) and its application. Such generation can be achieved via point set segmentation.

In many applications, large parts of the point set carry redundant information. For example, a flat area of a surface can be sampled sparsely without—compared to an area of high curvature—loosing information. In several applications, it is not even necessary to consider all details carried by the point set. For instance, in architecture—for a first draft—the rough outline of a building suffices and there is no need to send more detailed geometries. In general, when transmitting geometries e.g. to give an overview of a certain portfolio, the general outlines of the geometries suffices and sending only these saves on bandwidths during transmission. In this sense, algorithms are necessary that reduce a complex geometry to several basic shapes that still retain the most important features of the input. Towards this end, [CSAD04] proposed their *Variational Shape Approximation* (VSA) for meshes.

The VSA procedure segments a mesh into a given number of flat proxy regions, see Section 4.2. Finally, a simplified surface is obtained with only one element for each region, see Section 4.4. A translation of the VSA method to the setting of point sets was done by [LB16] with the explicit goal of feature curve extraction. While VSA is able to provide an easy to implement simplification of any geometry, it also has several downsides. First, it is dependent on the number of proxies which has to be chosen a priori. Second, in the previous publications, the quality of the result depends heavily on the manual placement of the starting seeds for the proxies ([CSAD04, LB16, YLW06]). Towards this end, two manual operations were proposed, which allow for splitting proxy regions of high error and merging neighboring ones with a combined low error [CSAD04, Sec. 3.5]. In the context of meshes, these operations have been automatized [YLW06, Sec. 3.1]. Third, the previous publications were not able to construct a VSA algorithm with guaranteed convergence. This work closes these gaps. Our main contributions are (with the last one extending [SZP20]):

- Providing an example of a growing error during the run of the VSA algorithm which applies to meshes and point sets alike.

- Presentation of a modified VSA procedure including the *switch* operation and proof of its guaranteed convergence.

- Inclusion of the two operations, *split* and *merge*, as automatic parts in the point set processing pipeline, making the initial choice of a fixed proxy number and the manual selection of seeds unnecessary.

- Extension of variational tangent plane intersection to the setting of point sets and inclusion of the procedure in the VSA pipeline for simplification.

- Formulation of the segmentation as a cloud complex together with cycle information as face boundaries and their usage in non-star convex face generation.

4.1 Related Work

The VSA procedure was introduced by [CSAD04] as a method for concise, faithful approximation of complex three-dimensional meshes. It does so by fitting a set of planar proxies to the input mesh. We will provide a detailed discussion of the procedure in Section 4.2. As the resulting elements are oriented corresponding to all associated faces of the original mesh, the effects of simplification are less drastic as in the classical approach of [GH97]. A next step towards even better approximations consisted of the inclusion of more than just planar shapes. In the work of [WK05], also e.g. cylinders and spheres are used as proxies to even better approximate the input shape. This was generalized even further by [YLW06] who utilized general quadrics as proxies to be fitted to the input. However, all these methods are implemented in the setting of surface meshes.

A translation of the VSA procedure to the setting of point sets was performed in an article by [LB16]. The authors studied the problem of computing smooth feature curves from CAD type point cloud models. Their reconstructed curves arise from the intersections of developable strip pairs which approximate the regions along both sides of the features. The generation of the developable surfaces is in turn based on VSA. While the presented results are convincing, it remains unclear whether the approach of fitting developable surfaces works outside of the CAD realm. Furthermore, the work does not provide details on how to obtain the used linear planar approximations or how to construct a watertight mesh from them. These aspects motivate the present research.

Our proposed approach incorporates two different areas of point set processing. On the one hand, we aim at segmenting the input into several flat—i.e. planar—parts for which we will discuss related segmentation approaches. On the other hand, we want to construct a simplified mesh on the basis of the found flat surfaces patches. Therefore, we will present corresponding work on mesh generation and simplification.

4.1.1 Segmentation

Segmentation of point clouds is the process of classifying the input into multiple homogeneous regions, where points in the same region will have the same properties. In real-world applications—like intelligent vehicles, autonomous mapping, and navigation—the problem is challenging because of high redundancy, uneven sampling density, and lack of explicit structure in the input data. Methods for point set segmentation can roughly be classified as follows: edge-based, region-based (seeded/bottom-up or unseeded/top-down), attribute-based, model-based, graph-based, or machine-learning-based, see [NL13], [GMR17]. Following this terminology, the VSA procedure is a seeded, region-based method, which is characterized by starting the segmentation process from seed points and letting regions grow by adding neighbors if they satisfy certain conditions—like normal similarity. We refer to the survey of [NL13] for a discussion of several corresponding methods. In this work, the authors draw the following conclusion on seeded region-based methods:

> *[They] are highly dependent on selected seed points. Inaccurate choosing seed points will affect the segmentation process and can cause under or over segmentation.*

The survey paper of [GMR17] draws a similar conclusion for the corresponding set of discussed methods. Hence, in contrast to the procedures covered in the mentioned surveys, we put an emphasis on the independence of both the number and placement of seed points. See Section 4.3.3 for a corresponding discussion.

Point cloud segmentation can be considered either from a semantic or from a geometrical perspective. The former aims at separating a model into its parts: A chair should for instance be segmented into four legs, a seating surface, and a backrest. The geometric approach is to segment the model into different primitives as-well-as-possible. A recent survey paper of [XTZ20] provides a comprehensive list of methods following both approaches. In the terminology used in this paper, the VSA approach creates a “plane point cloud segmentation”. While we cannot discuss all works mentioned, we will consider two popular approaches in the following and refer to the survey of [XTZ20] for a thorough discussion of other related work.

Note that fitting different planar segments to a model can be considered as a natural approach in order to render the model with a reduced set of planar patches. Naturally, those models are captured well that are comprised of mostly planar surface parts. The presence of spherical or cylindrical shapes will cause for larger distortions when approximating only with planar parts. Thus, a next step—after planar fitting—is the usage of other geometric primitives, like spheres, cylinders, cones, or tori. Each of these primitives then require their own fitting. As the VSA approach only fits planes, we briefly discuss different fitting concepts for this primitive. Note at this point that the method of [WK05] utilizes the exact same procedure for fitting of planes as the original VSA paper by [CSAD04].

To fit a planar patch, the approach of [SWK07] considers points $p_i, p_j, p_\ell \in P$ from an input point set P and computes a normal of the plane spanned by these points. This normal is then compared to the respective normals at p_i, p_j, and p_ℓ. A fitting plane is introduced if all three normal variations stay below a user-given angle. Clearly, the results of this approach heavily depends on the choice of the three points.

In contrast, the approach of [AP10] places a plane at the weighted barycenter b of a considered subset $\{p_i\} \subset P$ of the input point set P. A weighted covariance matrix is used to determine a normal n and the fitting error is computed as a weighted least-squares formulation. However, this computation neglects the normal information at the points p_i.

Another choice for the segmentation of point clouds is the algorithm presented in [RVDHV06]. It is popular because of its easily accessible implementation in the widely used Point Cloud Library (PCL) by [RC11]. This method can be seen as a reduced version of the VSA approach. Regions are also grown from seeds according to normal information. However, the growing process is only executed once and not repeated from a different set of seeds, like in VSA (see Section 4.2). Thus, the result is even more dependent on the initial seeding than in other, comparable techniques.

For the give reasons, the discussed methods have their respective downsides. Contrasting the presented algorithms, in our translation of the VSA approach, we include the entire normal information of the input point set. Furthermore, we have a setup of the pipeline that ensures independence of the initial seed points, which eliminates the major disadvantage of seed-based region growing methods.

4.1.2 Meshing and Simplification

As stated in the beginning of Section 4, the ultimate goal of our VSA procedure for point sets is to create a simplified mesh from a set of planar proxy regions. That is, a set of mesh vertices has to be created from the intersection of the proxy planes. Then, these vertices have to be connected to represent face elements for the proxies respectively. While three pairwise non-parallel planes intersect in a unique point, this is not necessarily the case for more than three planes in $\mathbb{R}^3$. In the context of planar panelization of freeform surfaces, [ZCHK12] confirm this statement, asserting that tangent plane intersection is numerically not stable enough to obtain reliable results. The authors proceed to present a variational approach and a corresponding minimization problem in order to obtain a planar representation of a given mesh structure. This method improves the approach of [CW07] for constrained planar remeshing of architectural geometries, which is itself based on VSA. Therefore, we turn to the work of [ZCHK12] to make the calculation of a simplified mesh from the input point set as-robust-as-possible. See Section 4.4 for a discussion of the technical details of the optimization and also for our translation to the setting of point sets.

Aside from VSA, there are other approaches to obtain a simplified mesh from an input point set. For instance, a possibility is to first mesh the input point set and then simplify the created mesh. An overview of methods for meshing of point sets is presented in the survey of [BTS$^+$17]. Several methods are available for the subsequent simplification of the mesh. These mostly collapse edges in the mesh to reduce its complexity. By using quadric error metrics, it can be assured that the collapses remove elements that carry the least amount of feature information, see [GH97]. This simple approach can be improved by adjusting the position of the vertex resulting from an edge collapse according to the local curvature information, see [HHL15, YHXL15]. However, as these methods perform their operations in a greedy manner, they do not provide reliable results when performing a drastic number of simplifications. Also, these approaches require a costly meshing operation on the unfiltered point set, which can introduce topological failures, like a surface of wrong genus or flipped triangles.

Another possibility to obtain a simplified mesh from an input point set is to first simplify the point set and to then create a mesh from this. A brief introduction and (error) analysis of different point set simplification algorithms can be found in the work of [PGK02]. Important attributes in real-world applications are the performance and quality of the rendering process. This requires a specific focus on features represented by the point sets. By utilizing a bilateral filtering, both Euclidean distances and normal information can be taken into account throughout a simplification process on a point set to best preserve both the large-scale geometry and small-scale features, in [SFC10]. While these methods are feature-preserving, they are not robust in the presence of outliers or noise. Also, the construction of the mesh cannot use the full information of the input point set anymore, as the majority of points will have been removed during the simplification step.

Because of the downsides of both the approaches, we aim at taking all points of the input into account when creating planar proxies. From these, we then create a mesh by completely creating new vertices and connections on them without going through a costly meshing operation on the original input, see Section 4.4.

4.2 Segmentation

In this section, we will present the Variational Shape Approximation (VSA) as introduced by [CSAD04] and as used by [YLW06] for surfaces and surface meshes. Also, we present a translation of the procedure to the setting of point sets, similar to the work of [LB16].

4.2.1 The VSA Procedure for Surfaces and Surface Meshes

The VSA procedure of [CSAD04] acts on a surface $S \subseteq \mathbb{R}^3$. The goal is to partition S into m disjoint regions $R_i \subseteq S$, $\dot{\bigcup} R_i = S$, where each region is associated a linear proxy (C_i, N_i) with a center $C_i \in \mathbb{R}^3$ and a unit-length normal $N_i \in \mathbb{R}^3$, $i \in \{1, \dots, m\}$. The authors propose two different metrics to find the optimal shape proxies, with the first metric based on the $\mathcal{L}^2$ measure

$$\mathcal{L}^2(R_i, C_i, N_i) = \int_{x \in R_i} \|x - \pi_i(x)\|_2^2 \, dx, \tag{4.1}$$

where $\pi_i(\cdot)$ denotes the orthogonal projection of the argument on the plane with normal N_i centered at C_i. Thus, the integral (4.1) measures the squared error between points in the region R_i and its linear approximation given by (C_i, N_i).

A second metric, denoted by $\mathcal{L}^{2,1}$ is based on the $\mathcal{L}^2$ measure when evaluated on the normal field. It is given by

$$\mathcal{L}^{2,1}(R_i, N_i) = \int_{x \in R_i} \|n(x) - N_i\|_2^2 \, dx, \tag{4.2}$$

where $n(x)$ denotes the normal of the surface at point $x \in S$. As [CSAD04] conclude that the $\mathcal{L}^{2,1}$ metric is more effective, we will reduce the following discussion to this formulation.

In the discrete setting, the surface S is given by a finite set of $T \in \mathbb{N}$ (triangular) elements t_j, $j \in [T]$ and the centers C_i are found by randomly choosing a triangle t_j as center C_i. Therefore, the second smooth formulation (4.2) can be discretized to

$$\mathcal{L}^{2,1}(R_i, N_i) = \sum_{t_j \in R_i} \|n(t_j) - N_i\|_2^2 \cdot |t_j|, \tag{4.3}$$

with $n(t_j)$ the normal and $|t_j|$ the area of the element t_j respectively.

The actual minimization of expression (4.3) with respect to the segmentation of S into regions R_i and with respect to the proxies (C_i, N_i) is then performed iteratively. For this, a variation of Lloyd's fixed point iteration given by [Llo82] is used. The first step is to pick a user-given number m of center elements $C_1, \dots, C_m$ randomly from the set of triangles $\{t_j \mid j \in [T]\}$. The normals N_i are set to the normals of corresponding center triangles C_i and the regions are initialized as $R_i = \{t_i\}$. The neighbors of the chosen center triangles are collected in a priority queue $\mathcal{Q}$ sorted increasingly with growing $\mathcal{L}^{2,1}$-distance between neighboring triangle and center triangle: $\|n(t_j) - N_i\|_2^2$. Then, the following three steps are performed iteratively until convergence:

1. *Flood*: As long as the queue $\mathcal{Q}$ is not empty, pop the first element t_j from $\mathcal{Q}$. Ignore it, if it has already been assigned to a region. If it is not assigned yet,

assign it to the region R_i that pushed it into the queue and push all neighboring elements of t_j into $\mathcal{Q}$, noting that they have been pushed by R_i. Without loss of generality, we assume S to be connected. If that is not the case, the algorithm can simply be run on each connected component of S. For a connected surface S, after the queue $\mathcal{Q}$ has been emptied, all elements $\{t_j \mid j \in [T]\}$ have been assigned to some region respectively.

2. *Proxy Update*: The proxy normals N_i are updated according to

$$N_i = \frac{\sum_{t_j \in R_i} |t_j| n(t_j)}{\left\| \sum_{t_j \in R_i} |t_j| n(t_j) \right\|_2},$$

where it is ensured that the updated N_i are unit-length normals. Note that as the surface will be segmented into a large enough number of locally flat patches, the denominator of this expression will never be zero in practice.

3. *Seed*: For each region R_i, find some element $t' \in R_i$ such that

$$\|n(t') - N_i\|_2^2 \leq \|n(t_j) - N_i\|_2^2$$

for all $t_j \in R_i$. This ensures that the flooding in the next iteration is started from regions that best reflect the current proxy normals.

Finally, the iteration is stopped, when no region changes from one step to the next. From the converged regions R_i and assigned proxies (C_i, N_i), a simplified mesh with corresponding m surface elements is constructed. Respective results are shown in [CSAD04].

4.2.2 The VSA Procedure on Point Sets

We will now proceed to present a translation of the VSA procedure to the setting of point sets. A corresponding reformulation can be found in [LB16], while we include weights to obtain a more general setup. Compared to the VSA on meshes, several details have to be adjusted for the method to work on point sets. At first, consider the partition problem as stated in Section 4.2.1. In the context of point sets, not elements, but the points themselves have to be assigned to the proxies. That is, the given point set $P = \{p_1, \dots, p_\aleph\}$ will be partitioned into disjoint subsets $\dot{\bigcup}_{i=1}^m P_i = P$, $m \in \mathbb{N}$. Therefore, in the following expressions, the centers C_i denote points from the point set P, while the normals N_i at the respective center point are those obtained from a normal field imposed on the point set. The normals of the points $p_j \in P$ will be denoted by n_j respectively.

Consider the energy as defined in Equation (4.3). For proxies obtained from point sets, the area term $|t_j|$ cannot be used. Thus, we replace it by a weighting term $\omega_j \in \mathbb{R}_{\geq 0}$ which is to approximate the area represented by the point $p_j \in P$. We obtain the following energy of a single proxy and the resulting energy formulation on the set of all proxies

$$\mathcal{L}^{2,1}(P_i, N_i) = \sum_{p_j \in P_i} \omega_j \|n_j - N_i\|_2^2, \tag{4.4}$$

$$E(\{(P_i, N_i) \mid i = 1, \dots, m\}) = \sum_{i=1}^{m} \mathcal{L}^{2,1}(P_i, N_i). \tag{4.5}$$

An approximation of the area term can be obtained by

$$\omega_j = \sum_{\ell \in \mathcal{N}(j)} \|p_\ell - p_j\|_2^2 , \tag{4.6}$$

where $\mathcal{N}(j) \subset P$ denotes the neighborhood of p_j in P, see Example 1 v). Including weights reflecting the area are of interest because of varying densities. Therefore, another possible weighting scheme could be the incorporation of directional density measures proposed in [SJP18]. This could be coupled in a bilateral manner together with the Euclidean distances mentioned before. In contrast, weight determination via normal deviations should not be used, as the energy is defined upon these, i.e. weighting these terms in the same fashion seems counterproductive.

The initial seeding as outlined above can still be done in the point cloud setting, but instead of triangles, now, $m \in \mathbb{N}$ points $p_j \in P$ are chosen for the initial position of the center points C_i. Also, those points from P are pushed to the priority queue $\mathcal{Q}$ that are neighbors, but not identical, to the chosen center points C_i. For this neighborhood relation, any neighborhood concept such as combinatorial k-nearest neighborhoods or geometric neighborhoods of radius r can be used (cf. Example 1 iv) and v)). Denote the neighborhood of p_i by $\mathcal{N}(i) \subset P$. Again, the points in $\mathcal{Q}$ are sorted increasingly with $\mathcal{L}^2$-distance between their own normal and the normal of the proxy that pushed them into the queue: $\|n_j - N_i\|_2^2$. The following three iteratively applied steps remain almost unchanged:

1. *Flood*: As long as the queue $\mathcal{Q}$ is not empty, pop the first element p from $\mathcal{Q}$. Ignore it, if it has already been assigned to a subset P_i. If it is not assigned yet, assign it to the subset P_i that pushed it into the list and push all neighboring points $p_j \in \mathcal{N}(i)$ into $\mathcal{Q}$, noting that they have been pushed by P_i. As we assume S to be connected via the imposed neighborhood relation (see above), after the queue $\mathcal{Q}$ has been emptied, all elements of P have been assigned to some subset P_i.

2. *Proxy Update*: The proxy normals N_i are updated according to

$$N_i = \frac{\sum_{p_j \in P_i} \omega_j n_j}{\left\| \sum_{p_j \in P_i} \omega_j n_j \right\|_2},$$

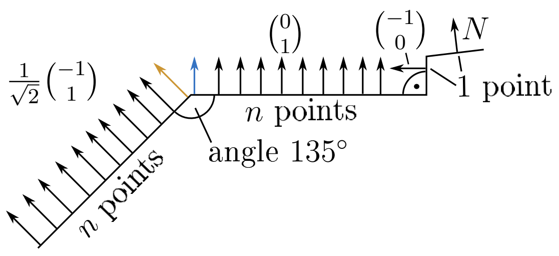
   where we once again obtain unit-length normals and will not encounter a denominator equal to zero (see above).

3. *Seed*: For all subsets P_i, find some $p_\ell \in P_i$, $\ell \in [\aleph]$, such that

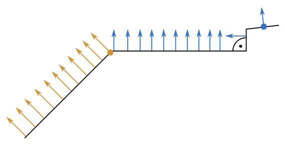
$$\|n_\ell - N_i\|_2^2 \le \|n_j - N_i\|_2^2$$

   for all $p_j \in P_i$. Again, this ensures that the next flooding step starts from regions that best reflect the current proxy normals.

Finally, once the subsets P_i do not change anymore over two iterations, the process is stopped. From the converged subsets P_i and assigned proxies (C_i, N_i), a simplified mesh with corresponding m surface elements is constructed. Respective results are shown in [LB16], while our corresponding approach will be discussed in Section 4.4.



(a) Setup for growing error functional.


(b) Segmentation after first flood.

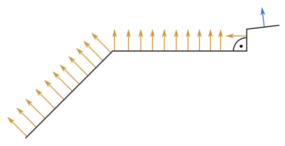

(c) Segmentation after second flood.

Figure 4.1: Example for a growth in the error measure after a flood and proxy update.

4.3 Improved Segmentation Pipeline

Having described the VSA procedure for both meshes and point sets, we now turn to our contributions for this pipeline. First, we will establish by an example that convergence of neither the meshed nor the point set version is guaranteed. Following up on this, we propose an alternative formulation of VSA with guaranteed convergence. Furthermore, we turn to a different issue of the VSA procedure. Namely, it is highly depended on both the number of initial seeds and their placement at the beginning of the procedure. We circumvent this dependency by including two more operations in the point set pipeline that have already been used manually [CSAD04] and automatically [YLW06] in the context of meshes.

4.3.1 Example of Failed Convergence of the VSA Procedure

Concerning the convergence of their algorithm, [CSAD04] state:

> *[...] Lloyd's algorithm always converges in a finite number of steps, since each step reduces the energy E: the partitioning stage minimizes E for a fixed set of centers c_i, while the fitting stage minimizes E for a fixed partition.*

While this statement holds for the original algorithm of Lloyd as presented in [Llo82], it does not hold for neither the VSA procedure on meshes as presented in [CSAD04, YLW06] nor for the translation to point sets as given by [LB16]. This is already recognized in the paragraph *Convergence* of Section 3.5 in [CSAD04]. We will demonstrate this with the following concrete example, which is to the best of our knowledge the first explicit example presented.

Consider the two-dimensional setup shown in Figure 4.1(a). It is given by n points connected on a line with normal $\frac{1}{\sqrt{2}}\binom{-1}{1}$ next to a line of n points with normal $\binom{0}{1}$. At the right end of the second line, there is a single point with normal $\binom{-1}{0}$ and another single point with normal N given by the equation

$$N = \frac{1}{\left\| n \cdot \binom{0}{1} + \binom{-1}{0} + N \right\|_2} \cdot \left(n \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} -1 \\ 0 \end{pmatrix} + N \right),$$

which solves to $N = \frac{1}{\sqrt{n^2+1}}\binom{-1}{n}$. Now, two proxies will act on this example, with their initial seeds shown in yellow and blue in Figure 4.1(a). They each start on one of the two lines of n points respectively. The result after a flood is shown in Figure 4.1(b), where each line is completely covered by the proxy starting on it and the two single points are associated to the proxy with normal $\binom{0}{1}$. After updating the proxy normals, the yellow proxy has normal $\frac{1}{\sqrt{2}}\binom{-1}{1}$ while the blue proxy has normal N given by the equation above. Thus, the yellow proxy starts from an arbitrary point on its line while the blue proxy starts from the rightmost point. The error after this first flood and proxy update is given by

$$E_1 = n \cdot \left\| \begin{pmatrix} 0 \\ 1 \end{pmatrix} - N \right\|_2^2 + \left\| \begin{pmatrix} -1 \\ 0 \end{pmatrix} - N \right\|_2^2 = -2(\sqrt{n^2+1} - n - 1),$$

where only the blue proxy contributes to the error, because the normals corresponding to the yellow proxy coincide with their proxy normal and cancel out in energy E_1. Starting from the new seed points, a second flood results in the situation shown in Figure 4.1(c). Here, almost all points except for the rightmost one are associated to the yellow proxy with former normal $\frac{1}{\sqrt{2}}\binom{-1}{1}$. Its new normal after a proxy update is

$$N' = \frac{1}{\left\| \frac{n}{\sqrt{2}} \cdot \binom{-1}{1} + n \cdot \binom{0}{1} + \binom{-1}{0} \right\|} \cdot \left(\frac{n}{\sqrt{2}} \cdot \begin{pmatrix} -1 \\ 1 \end{pmatrix} + n \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right),$$

which amounts to an error after the second flood and proxy update given by

$$E_2 = n \cdot \left\| \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix} - N' \right\|_2^2 + n \cdot \left\| \begin{pmatrix} 0 \\ 1 \end{pmatrix} - N' \right\|_2^2 + \left\| \begin{pmatrix} -1 \\ 0 \end{pmatrix} - N' \right\|_2^2 .$$

Note that the error term for the blue proxy cancels out, as the one representative corresponds to the normal of the proxy it belongs to. Choosing $n = 100$ points on each of the two lines, we obtain $E_1 \approx 1.9900$, but $E_2 \approx 31.6782$. Furthermore, the corresponding error value after the flood is also growing. Thus, convergence cannot be proven by an always shrinking error functional.

Note that this example is described as a curve in 2D, where neighborhood selection is generally more involved than for surfaces in 3D. However, the example can easily be extended to a surface in 3D space, see Figure 4.2. There, we also close the loop and thereby cause the original VSA algorithm to run infinitely long. For the given example, the crucial step as depicted in Figure 4.1(c) can be resolved via a manual ([CSAD04]) or automatic ([YLW06]) split of the large proxy. Thus, this example only applies to the VSA procedure as described above.

4.3.2 VSA with Guaranteed Convergence

The example presented above highlights the main deficiency of the VSA procedure as used in [CSAD04, YLW06, LB16]. Namely, if an outlier causes a proxy normal to be distorted, the new proxy seed can end up to be a border point that does not actually reflect the normal behavior of the majority of points in the proxy. In other words, the change of seeds before flooding is a problematic step. Thus, in the following, we aim at altering the VSA procedure in a way such that no new seeds need to be found, but proxies can still move and alter. In particular, proxies should be able to take over the original seed points of other proxies if necessary. These changes should finally lead to an alternative VSA procedure with guaranteed convergence. In order to achieve this goal, we propose to alter the steps of the algorithm as follows.

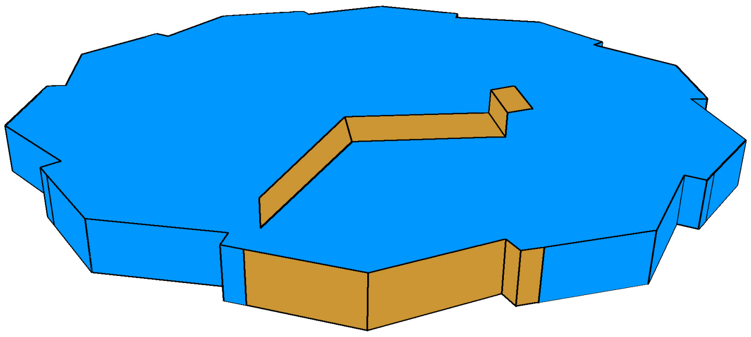


Figure 4.2: A regular 10-gon, built from the shape shown floating on top, which is a three-dimensional extension of the setup shown in Figure 4.1(a).

First, we perform an initial seeding and one flood step and proxy update as explained in Sections 4.2.1 and 4.2.2 above. Instead of the seeding step in the following iterations, we perform a different procedure:

4. *Switch*: Consider the neighborhoods $\mathcal{N}(i) \subset P$ for all points $p_i \in P$. Assume that p_i is assigned to subset P_ℓ. If any point $p_j \in \mathcal{N}(i)$ is assigned to another subset P_h, compute the change of the error measure (4.5) resulting from reassigning p_i from P_ℓ to P_h. Compare it to the current best known reassignment. After iterating through all points $p \in P$, reassign the point such that the error measure is reduced maximally.

This new switch step replaces the *seed* step and the *flood* step described in Sections 4.2.1 and 4.2.2 above. That is, it is only iterated together with the *proxy update*. The iteration is continued until no further switch operations can be performed. For this alternate procedure, we can prove the following statement.

Theorem 1 (Error reduction by switch and proxy update)**.** *Given a point set $P = \{p_1, \ldots, p_\aleph\}$ with a neighborhood structure, such that the neighborhood graph on P is connected and normals $n_1, \ldots, n_\aleph$ on P. Then, each proxy update step and each switch step as defined above leads to proxies (P_i, C_i, N_i) with a smaller error measure in Equation (4.5).*

Proof. Concerning the proxy update step, consider

$$\begin{aligned}
\nabla E(\{(P_i, N_i) \mid i \in [m]\}) &= \nabla \sum_{i=1}^{m} \mathcal{L}^{2,1}(P_i, N_i) \\
&= \sum_{i=1}^{m} \sum_{p_j \in P_i} \nabla \omega_j \, \|n_j - N_i\|_2^2 \\
&= \sum_{i=1}^{m} \sum_{p_j \in P_i} 2\omega_j (n_j - N_i).
\end{aligned}$$

Setting $N_i = \frac{\sum_{p_\ell \in P_i} \omega_\ell n_\ell}{\sum_{p_\ell \in P_i} \omega_\ell}$, we obtain

$$\begin{aligned}
\sum_{p_j \in P_i} 2\omega_j (n_j - N_i) &= \sum_{p_j \in P_i} 2\omega_j n_j - \sum_{p_j \in P_i} 2\omega_j \left(\frac{\sum_{p_\ell \in P_i} \omega_\ell n_\ell}{\sum_{p_\ell \in P_i} \omega_\ell} \right) \\
&= \sum_{p_j \in P_i} 2\omega_j n_j - \left(\frac{\sum_{p_\ell \in P_i} 2\omega_\ell n_\ell}{\sum_{p_\ell \in P_i} \omega_\ell} \right) \cdot \sum_{p_j \in P_i} \omega_j \\
&= \sum_{p_j \in P_i} 2\omega_j n_j - \sum_{p_\ell \in P_i} 2\omega_\ell n_\ell = 0.
\end{aligned}$$

Thus, at the chosen updated proxy normal, the energy reaches a (local) minimum. As the energy is convex as sum of norms, which are convex, the found minimum is indeed its global minimum for the current choice of segmentation.

Concerning the switch step, only those points are reassigned which reduce the value of error measure (4.5). Thus, trivially, after a switch operation the error is smaller. □

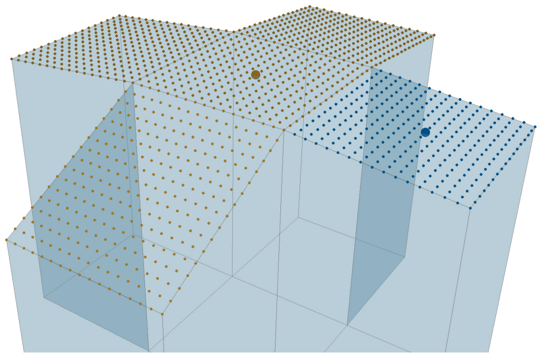
Finally, we note that there are only finitely many ways to partition the $\aleph$ points of the point set P into m subsets. This fact, together with Theorem 1 proves the convergence of our modified VSA procedure.

Consider the application of this alternative VSA version to the setup in Figure 4.1(a). After a first flood, which would still result in the proxies shown in Figure 4.1(b), the only possible switch could be performed at the border between the blue and the yellow region. However, a switch would already lead to an increase of the energy functional. Thus, the proxies remain as they are after the first flood and the example converges immediately.

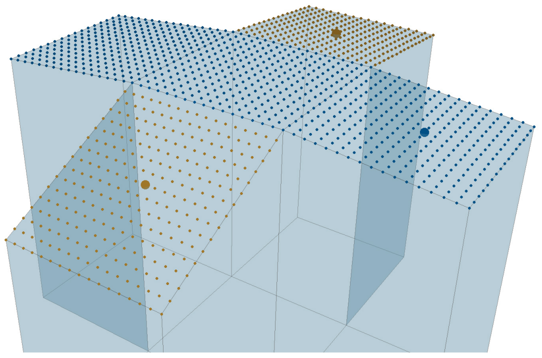
By replacing *seed* and *flood* with the *switch* operation, we can ensure convergence of the algorithm. While this result is theoretically pleasing, it is not necessarily of practical value. Finding an ideal pair of points for a switch operation requires to iterate at least over all points on the border of proxy regions. Depending on the number of proxies and on the shape of the geometry, one such switch can reach the same time complexity as a flood operation while only altering a single point's proxy assignment. Thus, in practice, utilizing the *switch* operation causes a significantly longer runtime as trade-off to the guaranteed convergence.

Furthermore, iterated application of the switch operation can tear a proxy apart, see Figure 4.3. A converged state of the algorithm might therefore include proxy regions that are not connected. In order to have a sensible result, a final step has to be included that re-interprets connected regions as proxies and that might increase the number of proxies doing so. However, a disconnectedness only arises if another proxy better reflects the local shape. Thus, a corresponding higher number of connected proxy regions is desirable in order to faithfully approximate the input geometry.

The presented *switch* operation provides one possible way to obtain guaranteed convergence. It remains as open question whether another operation or alteration of the VSA procedure provides the same result while coming with a lower runtime.



(a) A geometry after initial selection of seeds as indicated and flooding.



(b) The same geometry after several switch operations. The blue proxy split the other proxy in two components.

Figure 4.3: A proxy being torn apart by another proxy under the repeated application of the switch operation.

4.3.3 User Controlled Level of Detail

The requirements of proper seed placement and prescribed seed number naturally demand for a variable proxy-treatment in terms of *splits* and *merges*. Both concepts were introduced in [CSAD04] as means for manual adjustments by the user. For meshes, these two operations are incorporated into the pipeline described in [YLW06]. In the following, we propose a translation to point sets. With both operations, we aim at adaptability of the constructed flat pieces towards user input. That is, the user should be able to control the level of detail obtained from the flat regions. However, in contrast to [CSAD04], this control should be realized via a single input parameter instead of a time-consuming manual interaction with the modeling process. For this, we use a user-given parameter $\eta \in \mathbb{R}_{\geq 0}$ which controls the maximum deviation of a subset P_i from its flat approximation. It can be thought of as controlling the maximum bending of a segment. This parameter is used in the following two steps:

(a) *Split*: Given a subset $P_i \subset P$ such that $\mathcal{L}^{2,1}(P_i, N_i) > \eta$. We use weighted principal component analysis by [HBC11] to compute the most spread direction of P_i. The set P_i is then split at the center of this direction into two new sets $P_i = P_i^1 \dot\cup P_i^2$. The new normals are chosen as $N_i^1 = \sum_{p_j \in P_i^1} \frac{\omega_j n_j}{\sum_{p_j \in P_i^1} \omega_j}$ and N_i^2 respectively. The new centers C_i^1 and C_i^2 are then placed at those points of P_i^1 and P_i^2 that have least varying normals from N_i^1 and N_i^2 respectively.
Note that the reasoning of Theorem 1 holds for this case, too. Thus, the procedure outlined above, with an additional split step does continue to converge.

(b) *Merge*: Consider a pair P_i, P_j of neighboring subset with their respective normals N_i, N_j. If the subset $P' = P_i \cup P_j$ with normal

$$N' = \left\| \frac{|P_i| \cdot N_i + |P_j| N_j}{|P_i| + |P_j|} \right\|^{-1} \frac{|P_i| \cdot N_i + |P_j| N_j}{|P_i| + |P_j|}$$

achieves an Energy (4.5) strictly less than η, the two subsets are replaced by their union P', with normal N' and a chosen center $C' \in P'$ with its normal least deviating from N'.
Note that we could allow only those pairs of neighboring regions to merge such that

$$\mathcal{L}^{2,1}(P_i, N_i) + \mathcal{L}^{2,1}(P_j, N_j) \geq \mathcal{L}^{2,1}(P', N').$$

Then, the energy would not increase and termination of the algorithm would be guaranteed by Theorem 1. However, this would result in neighboring regions not observing the user-given η threshold. Therefore, we accept an increase of the global energy in favor of a better region layout[6].

Both operations alter the number m of proxies. Thereby, a significant disadvantage of the algorithm of [Llo82] is eliminated as the user does not have to choose m a priori. It is replaced by the user's choice of η, providing a semantic guarantee on the regions being built by the algorithm. The user can prescribe a value of η based on the curvature and number of points within a proxy. In the experiments we discuss a more detailed interpretation of η.

The possible presence of noise in the point set P gives yet another reason to refute Energy (4.1). For points distributed around the xy-plane, with normals $(0, 0, 1)^T$ and just slight deviation from the plane, this energy would create larger values for a growing number of points, while the Energy (4.5) does not suffer from this. Hence, with the chosen energy, noise on the point positions is handled more robustly.

In the merge process outlined above, we asked for two neighboring regions. Moreover this is important for our complex to be generated, cf. Equation (3.2). However, we have not defined any relation on the regions yet. In the meshed case discussed in Section 4.2.1, two regions are neighbors if and only if they share an edge in the mesh. In the context of point sets, we cannot rely on this, thus we have used the following approach.

Remark 6. *For every proxy and each of its points, we query k of the point's nearest neighbors and use the distance to the farthest of them for a geometric neighborhood determination. From all the neighbors gathered that way, we ask for their proxy assignment. If the current center point is assigned to a different proxy than its neighbors, we consider the two proxies to be neighbors.*

This finishes the whole pipeline, including the additional two steps *merge* and *split*. In addition, it provides the possibility to obtain a complex K (Equation (3.2)) from the segmentation of the point set using the neighborhood of segments outlined in Remark 6. See Figure 4.4 for an illustration of the complete pipeline.

4.3.4 Experimental Results

In the following experimental section we evaluate parameter choices regarding the segmentation and provide a quantitative comparison of the segmentation results. In addition, we discuss the interpretation of parameter η.

[6]Note that the equation for N' in this description of the *merge* procedure deviates from the equation given in the published version of the article in *Computer Aided Geometric Design 2020, Vol. 80.* The formulation given here is more general and works in particular if P_i and P_j are of different sizes. Furthermore, in the published version, the inequality on $\mathcal{L}^{2,1}$ was given in the wrong direction, which is corrected here.

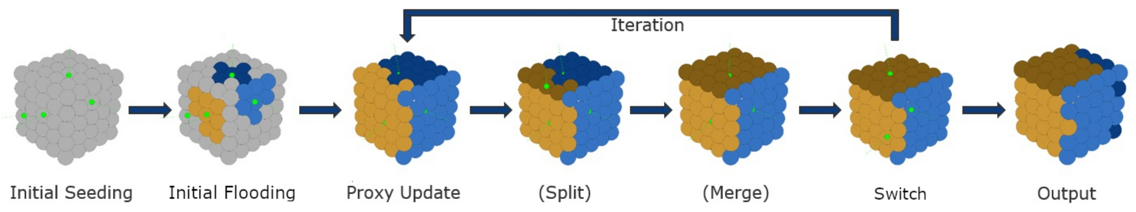


Figure 4.4: The whole pipeline contains an initial random seed selection and an initial flooding. From there, a proxy update, one or more optional splits and/or merges, and a switch are iterated until no further switches can be applied. Afterwards, we deduce a simplified model according to the proxies, which can also be considered as a simplified surface reconstruction from the initial point set.

Quantitative Comparison In all our experiments, we processed models with quite uniform samplings. Hence, for simplicity, we utilized equal weights $\omega_j = 1$ in Equation (4.4). We proceed similarly with the weight assignment in the optimization problem formulated in Equation (4.11) and set $\tilde{w} = 1$. We use neighborhoods to both propagate a proxy during the *flood* step and establish neighborhood relations between the different proxies. For the first purpose, we use a combinatorial k-nearest neighbors approach. When determining the proxy neighborhoods, we turn to a combination of the combinatorial and geometric approach based on the same k (see end of Section 4.3.3). In all our experiments, we use $k = 8$. Deviations from the default parameters are indicated.

For a large scale experiment, we chose 600 models from the repository used in the work [HZG$^+$18]. For all these models, we used the mesh information to generate an oriented vertex normal field. Furthermore, we translated the models and scaled them uniformly to fit into the unit cube. Finally, we performed our experiments on the point cloud given by the mesh vertices, disregarding the connectivity information and the triangular faces.

We compared four different approaches. The first one was the segmentation algorithm of [RVDHV06] as implemented in the *Point Cloud Library* of [RC11]. As parameters, we turned to the ones described in the original paper, see [RVDHV06]. We will refer to this experiment by *PCL*. Second, with these results at hand, we took the final numbers of proxies given by *PCL* for each geometry. This number served as number of proxies to be sought by the variational shape approximation algorithm of [LB16]. Here, no splits or merges are applied, thus we refer to this experiment by $\neg s/m$. Third, we took the total $\mathcal{L}^{2,1}$-error (Equation (4.5)) of each geometry, as produced by the *PCL* experiment, and divided it by its final number of proxies. This division provides an initial guess for a local, geometry-dependent value η. In this third experiment, we allowed splits as well as merges. Also, we started with an initial seed number of $m = \aleph$, i.e. each point was a seed at the start. Because of the η-threshold and merge-processes, the number of proxies reduced drastically over the run of the experiment. We will refer to this as *local* η s/m. Fourth and finally, without any priors, we set $\eta = 25$ and allowed splits and merges. Furthermore, we once more started with every point as a seed. The choice of η is motivated from previous experiments. We will refer to this fourth experiment as *global* η s/m. The terminology *local* or *global* indicates whether η is chosen with respect to the geometry or globally for all geometries. Observe that the experiments $\neg s/m$ and *local* η s/m are dependent on the results of *PCL*, while only *global* η s/m is independent.

	PCL	$\neg s/m$	*local* η s/m	*global* η s/m
min MSE	2.75E-09	7.31E-07	**6.95E-11**	2.11E-06
max MSE	1.12E-01	2.98E-02	4.15E-02	1.04E-02
avg MSE	1.59E-02	**2.88E-04**	**6.40E-04**	**4.50E-04**
sd MSE	2.12E-02	1.90E-03	2.47E-03	8.15E-04
min m	1.00	1.00	2.00	1.00
max m	1,103.00	1,103.00	393.00	135.00
avg m	174.17	174.17	**55.05**	**33.20**
sd m	163.10	163.10	58.41	20.69

Table 4.1: Statistical evaluation of error MSE and proxy number m taken over all 499 geometries. We give the minimum, maximum, mean, and standard deviation.

We are interested in gaining insight into the relationship between the obtained proxy-number m and the quality of the induced flat proxy-regions. Besides the $\mathcal{L}^{2,1}$-measure of Equation 4.5, we focus on the mean squared error (MSE) caused by point-to-proxy-plane distances to evaluate the region quality. The MSE is given as

$$MSE(\{(P_i, N_i) \mid i = 1, \ldots, m\}) = \frac{1}{\aleph} \sum_{p_j \in P_i} \|p_j - \pi(p_j)\|_2^2, \tag{4.7}$$

where $\pi(p_j)$ denotes the orthogonal projection of p_j onto its related proxy plane, given by normal N_i and base point C_i.

From the 600 chosen models, we obtained 499 that offered segmentation results in all four experiments. For 27 models, *PCL* was unable to provide a valid segmentation, because it assigned a zero proxy-normal to at least one region (for instance a region holding only two antipodal normals). These models were excluded for the subsequent experiments. The variational shape segmentation of [LB16] did not report a complete segmentation on additional 72 models. Here, some points are not assigned to any proxies, because they cannot be reached from the proxy centers during a *flood* when traversing the nearest neighbor graph. Increasing the parameter k alleviates this problem. Similar failures occurred on one additional model in experiments *local* η s/m and *global* η s/m respectively. Even though these experiments started with seed numbers equal to the geometries' points, they reduced the number of regions via *merge* operations. Due to proxy updates and new seed selection, it is possible that seeds travel away from sparsely sampled areas, where they do not reach all formerly assigned points in the neighborhood graph during the next *flood*. This reduced the number of models by a total of 101 failures to 499 feasible models. All reported experimental values are taken over this set of 499 models.

For the following analysis, we turn to Table 4.1. There, we give statistics on both the MSE as obtained from experiments on our model set. Regarding the average MSE over all experiments, we see that all three experiments—$\neg s/m$, *local* η s/m, and *global* η s/m—outperform *PCL* by two orders of magnitude. A direct comparison between the MSE obtained for the models reveals that *local* η s/m and *global* η s/m outperform $\neg s/m$ in roughly 8.5% of all experiments. Note that the minimal MSE error obtained over all geometries is up to five orders of magnitude smaller for *local* η s/m when compared with the other experimental setups.

Aside from the MSE results, Table 4.1 also reports statistics on the number of proxies obtained by the different experiments over all geometries. Recall that we are

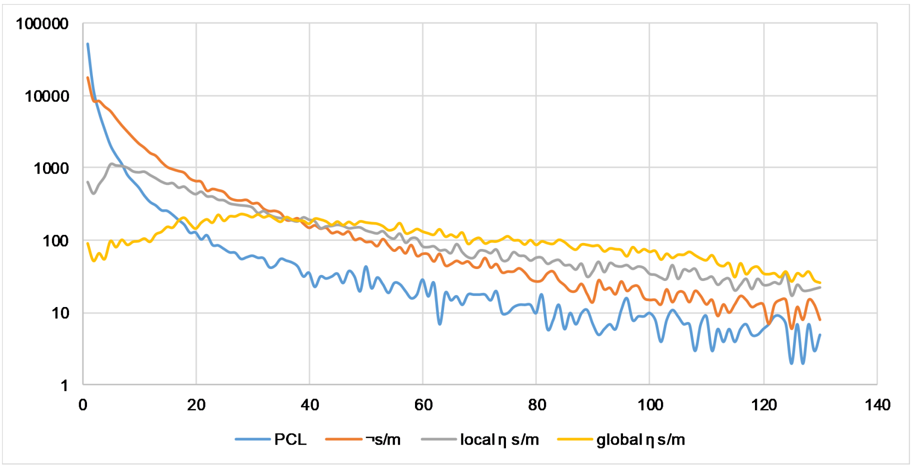


Figure 4.5: Histogram over all proxy-sizes up to 131 among all 499 geometries for all four experiments. The upper bound of 131 is given by the sum of the mean (11.74) and corresponding standard deviation (120.27) regarding cluster sizes obtained from *PCL*. Note the logarithmic scale on the y-axis.

not only interested in small error values, but also in representations of the geometry that reduce its complexity, i.e. that have a low number of proxies. Towards this end, it is remarkable that *local* η s/m and *global* η s/m attain MSE values comparable to those of $\neg s/m$ while only using 31.6% and 19.1% of the proxies on average, respectively. The close error results are especially of interest for *global* η s/m, as this runs without any dependency or information provided by *PCL*, as a global parameter is applied to all geometries equally. Hence, the assignment of points to proxies is on one hand optimized in terms of the MSE errors measure, while providing significantly fewer proxies on the other hand. Note that the lowest (and therefore optimal) MSE of 0 is given for a segmentation in which every point is represented by its own proxy.

We proceed to further investigate the proxies obtained by the different experiments. In Figure 4.5, we show a histogram over the attained proxy-sizes taken over all geometries in the experiment. Note that the y-axis has a logarithmic scale. We show all proxy sizes up to 131, where this bound is given by the sum of the mean (11.74) and corresponding standard deviation (120.27) regarding proxy sizes obtained from *PCL*. We can see that both *PCL* and $\neg s/m$ create a significantly larger number of small proxies when compared with the segmentation results of *local* η s/m and *global* η s/m. In fact the average proxy sizes are 11.74 (*PCL*,$\neg s/m$), 37.15 (*local* η s/m), and 64.00 (*global* η s/m). As segmentation—in our setup—should create few regions that still reflect the geometry attributes, extremely small regions as exposed by *PCL* and $\neg s/m$ are undesirable. The availability of splits and merges in *local* η s/m and *global* η s/m results in a bell-curve-like behavior in Figure 4.5, as both curves first increase and show a small descent with minor oscillations after their peaks. Hence, in the critical area of small sized proxies, the availability of splits and merges not only reduces their required number, but also balances their sizes, causing for more uniformly sized proxies.

To summarize the quantitative analysis of the segmentation part, we conclude:

- The proposed method outperforms the segmentation approach of [RVDHV06] as well as variational shape approximation without splits and merges, as used by [LB16] in regard of MSE.

- Without any knowledge of seed numbers or error values, a globally set η, availability of splits as well as merges, and treatment of all points as initial seeds provides segmentation results that have MSE comparable to [RVDHV06, LB16] but a significantly reduced number of proxies.
- The availability of splits and merges not only optimizes for small proxy numbers, but also causes more uniform region sizes.

Interpretation of parameter η In Section 4.3.3, we introduced the user-chosen parameter η. It relates to the energy $\mathcal{L}^{2,1}$ as presented in Equation (4.4). From the definition of $\mathcal{L}^{2,1}$, it is clear that two factors contribute to the value $\mathcal{L}^{2,1}(P_i, N_i)$ a given proxy P_i can achieve. These are:

- The number of points p_j assigned to the proxy and
- the Euclidean distance of the proxy normal N_i to the respective point normals n_j.

That is, a proxy can achieve a low energy by either exhibiting low deviation in its normals or by containing a low number of points. In particular the latter aspect highly depends on the number of points and the point densities in the considered model. Therefore, no general values of η can be presented in this work, but the user has to choose an appropriate value for the given setup. In the following, we present a simple heuristic how to make an (initial) choice for η.

Any point on a smooth surface can be approximated via a quadric, i.e. a (hyperbolic) paraboloid [DN98]. As we handle mostly (locally) convex objects, we consider an elliptic paraboloid as a simple model for a curved surface, parameterized as

$$\mathcal{P} = (u, v, \frac{u^2}{a^2} + \frac{v^2}{b^2}),$$

then it has mean curvature

$$H(u,v) = \frac{a^2 + b^2 + \frac{4u^2}{a^2} + \frac{4v^2}{b^2}}{a^2 b^2 \sqrt{\left(1 + \frac{4u^2}{a^4} + \frac{4v^2}{b^4}\right)^3}}. \tag{4.8}$$

A normal to $\mathcal{P}$ at (u, v) is given as

$$\mathcal{P}_u \times \mathcal{P}_v = \begin{pmatrix} 1 \\ 0 \\ 2u/a^2 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 2v/b^2 \end{pmatrix} = \begin{pmatrix} -2u/a^2 \\ -2v/b^2 \\ 1 \end{pmatrix}.$$

Hence, after normalization, the point parameterized at (u, v) contributes the following value to $\mathcal{L}^{2,1}$, when assuming that the points are distributed uniformly on the paraboloid and therefore the proxy normal is just $N_i = (0, 0, 1)^T$:

$$\left\| \frac{1}{\sqrt{4u^2/a^4 + 4v^2/b^4 + 1}} \begin{pmatrix} -2u/a^2 \\ -2v/b^2 \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\|^2$$
$$= 2 - \frac{2}{\sqrt{4u^2/a^4 + 4v^2/b^4 + 1}}.$$

Placing a number $\aleph_i$ of points regularly on the domain $[-1,1] \times [-1,1]$, i.e. choosing $u = j/\nu_i$, $v = \ell/\nu_i$ for $j, \ell = 1, \dots, \nu_i$ and $\nu_i := \lceil \frac{\sqrt{\aleph_i}-1}{2} \rceil$, we can compute the total value of $\mathcal{L}^{2,1}$ for these points, depending on the curvature prescribed by (a, b) as

$$\mathcal{L}^{2,1}(P_i, N_i) = 2(2\nu_i + 1)^2 - \sum_{j=-\nu_i}^{\nu_i} \sum_{\ell=-\nu_i}^{\nu_i} \frac{1}{\sqrt{\left(\frac{j}{\nu_i a^2}\right)^2 + \left(\frac{\ell}{\nu_i b^2}\right)^2 + \frac{1}{4}}}. \tag{4.9}$$

Now Equation (4.9) provides a heuristic to compute an (initial) value of η: A user of the algorithm first chooses a desired curvature, to be covered by the proxies. From this choice and a distribution on the two main curvature directions, via Equation (4.8), the parameters (a, b) can be computed. As the user also knows the models to which the algorithm will be applied and therefore the resolution, i.e. the number of points to be included, a second choice is the number of points $\aleph_i$ that is roughly to be covered by a single proxy. From these two choices, using Equation (4.9), a first estimate for η can be computed. If the output of the algorithm is not satisfactory, the user is of course free to tune the parameter towards the desired result.

4.4 Simplification

We will now investigate the creation of a simplified mesh based on the segmentation generated before. Both works of [CSAD04] and [LB16] present simplified meshed geometries with m faces, one representing each proxy. The work of [YLW06] also presents simplified meshes, utilizing their proxy quadrics. However, the approaches of [CSAD04] and [YLW06] are not suitable for our context as they work on meshes. The authors of [LB16] do not elaborate on the computation of their meshes. They only state that

> *(...) a polygonal mesh is easily generated by computing intersections of proxy planes of neighboring clusters of data points.*

In the following, we will see that only simple cases allow for this approach while the general case is more involved. First, we will discuss the creation of vertices for the simplified mesh (Section 4.4.1). Subsequently, we will connect these vertices to faces in order to obtain the complete mesh (Section 4.4.2). In both sections we also address corresponding challenges as-well-as-possible solutions.

4.4.1 Vertices for a Simplified Mesh

The intuitive way to determine simplified mesh vertices is the intersection of neighboring proxies, as used by [LB16]. In the following, we will use the notion of neighborhood for proxies as introduced in Remark 6 in Section 4.3.3.

Intersecting Planes A first naive solution for the creation of vertices for the simplified mesh is to consider the intersection of neighboring proxies and the construction of vertices in these intersection points. In the general case, where $q > 3$ proxy planes meet, we cannot simply consider the intersection as it will be mostly empty.

We call such a situation obtained via the proxy-neighborhood relation a *q-tuple*. Let I be the index set labeling the proxies given by subset $P_i \subseteq P$, proxy center C_i, and proxy normal N_i. Then we build a set of subsets $\mathcal{I}$ on I as in Equation (3.1) in Section 3 w.r.t. the neighborhood relation on proxies mentioned in Remark 6. Now a *q-tuple* is an element in $\mathcal{I}_J = \{I_j\}_{j \in J}$ for an index set J holding inclusion maximal candidates for intersection points with proxy indices contributing to the intersection.

Now, for the case of $q > 3$, we select for each tuple three indices at random, intersect them, and make the resulting vertex known to all proxy members of the tuple. As this might cause degenerate faces in the face creation stage—as the vertex does not lie within all of the proxies it is associated to—, we use a triangulation of all created faces (see Section 4.4.2) to obtain triangles, which are planar.

Intersecting Point Optimization A second, more involved solution for the creation of simplified mesh vertices is based on optimization. The intersection of more than three planes is numerically unstable as discussed above. In the work of [ZCHK12], the authors turn to a variational approach, start from a triangle mesh, and aim at computing the intersection points x_j of the vertex tangent planes of all triangles. Thus, exactly three tangent planes—corresponding to each of the three vertices v_i of a triangle—contribute to an intersection point. Denoting the normal at v_i by n_i, they solve the following minimization problem

$$\begin{aligned} \text{minimize:} \quad & \sum_{t_j} \left(\sum_{v_i \in t_j} \|x_j - v_i\|_2^2 \right) \\ \text{subject to:} \quad & n_i^T (v_i - x_j) = 0 \quad \forall t_j, \quad \forall v_i \in t_j \\ & \|n_i\|_2^2 = 1 \quad \forall v_i \end{aligned} \tag{4.10}$$

where the normals are variables in the minimization. Note that the original normals at the vertices are not taken into account at all during the minimization. The requirement of unit-length normals is necessary, however, as otherwise $n_i = 0$ would trivially satisfy all conditions.

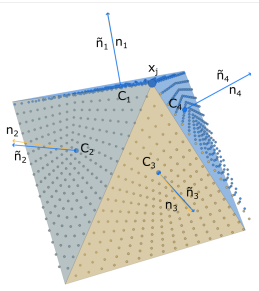
We generalize this approach in the following way to our setup. First of all, we use the concept of and notation for *q-tuples* introduced in the previous paragraph. Then, we consider the following energy

$$F(\{x_1, \ldots, x_j\}) = \sum_{j \in J} \left(\sum_{i \in I_j} \|x_j - C_i\|_2^2 \right) + \sum_{i \in I} \tilde{w}_i \|N_i - \tilde{n}_i\|_2^2, \tag{4.11}$$

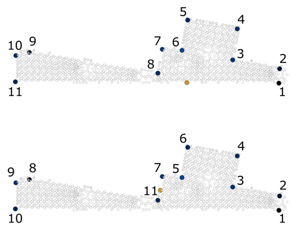
with sought-for intersection points x_j for each maximal tuple I_j, known proxy-centers C_i, weighting terms $\tilde{w}_i \in \mathbb{R}_{\geq 0}$, unknown normal deviations $\tilde{n}_i$, and known proxy-normals N_i for all proxies $i \in I$. Ultimately, we want to solve

$$\begin{aligned} \text{minimize:} \quad & F(\{x_1, \ldots, x_j\}) \\ \text{subject to:} \quad & \tilde{n}_i^T (x_j - C_i) = 0 \quad \forall j \in J \quad \forall i \in I_j \\ & \|\tilde{n}_i\|_2^2 = 1 \quad \forall i \in I. \end{aligned} \tag{4.12}$$

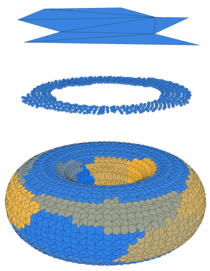
This generalizes the problem of [ZCHK12] as stated in Equation (4.10) in several ways. First, we allow for more than three, namely for an arbitrary number of planes to intersect. This arises already at a simple geometry like the octahedron, which



(a) Intersection of more than three proxies



(b) top: star-convex; bottom: barycenter



(c) Reconstructing a non-convex proxy

Figure 4.6: (a) Illustration of point optimization with deviation allowance (represented by $\tilde{n}$) of $n_1, \ldots, n_4$ to find the optimal mesh vertex x_j satisfying the constraints $\langle C_i - x_j, \tilde{n}_i \rangle = 0$ and $\|\tilde{n}_i\|_2^2 = 1$ for $i = 1, \ldots, 4$. The C_i are the normal-corresponding proxy-centers and n_i the original proxy normals. (b) Star-convex proxy (front part of the Fandisk, Figure 4.11). Top shows an ordering of the vertices around a star-convex center point, bottom shows the corresponding ordering around the barycenter.
(c) Bottom: Segmentation of the Torus with 24 final proxies. Middle: Non-convex proxy representing the upper part of the torus after projection of the points onto their proxy plane. Top: Resulting mesh face after connecting the vertices.

has valence 4 vertices, see Figure 4.6(a) for an illustration. Second, we do allow the normals $\tilde{n}_i$ to deviate from the proxy normals N_i, but a large deviation is punished, where the severity can be steered by the weights $\tilde{w}_i$.

In contrast to the first naive solution, this approach guarantees all vertices of the mesh to lie within the proxies that they are derived from. That is, if the optimization problem (4.12) yields a feasible point, after correcting the proxy normals from n_i to $\tilde{n}_i$ all vertices associated to a planar proxy lie completely within the corrected proxy plane.

4.4.2 Faces for a Simplified Mesh

After creating the mesh vertices, we need to connect them in order to generate faces for the mesh. In [SZP20] we focused on star-convex faces which we will discuss next. Afterwards we use the complex presented in Section 3 obtained from the segmentation to treat non-convex faces.

Star-convex Faces The general idea is to represent every proxy region with a single star-convex face. All vertices associated to a proxy are sorted around the barycenter of the proxy w.r.t. an arbitrary reference direction. This yields correct results, when the barycenter of the proxy is also a star-convex center point, see Figure 4.6(b). If we sort the vertices with a non-star-convex center, they are connected in wrong order and the resulting faces will degenerate. This approach obviously fails if a proxy represents a non-convex part of the geometry, see Figure 4.6(c).

General Faces We want to use the information encoded in the complex set up in Equation (3.2), especially the cycles per region, to construct even non-convex

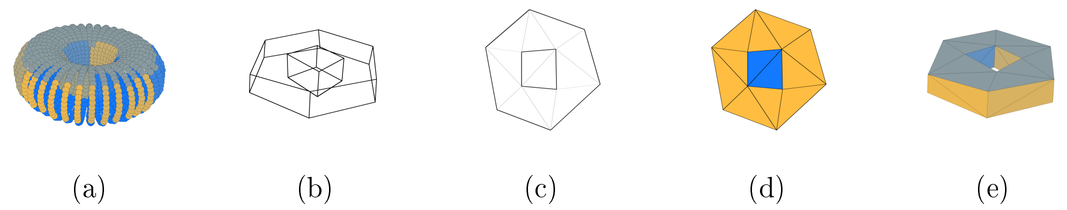


Figure 4.7: The pipeline illustrates how we start with a segmentation (a) and retrieve a graph representation (b) from the segmentation. Then we send all cycles belonging to a region to a local coordinate system (here top two cycles), generate a constrained triangulation respecting the cycles' information (c), using a bipartition argument to find the final face (the one in orange) (d) and send this information back to the graph representation. Applying this to all regions yields our final simplified mesh (e).

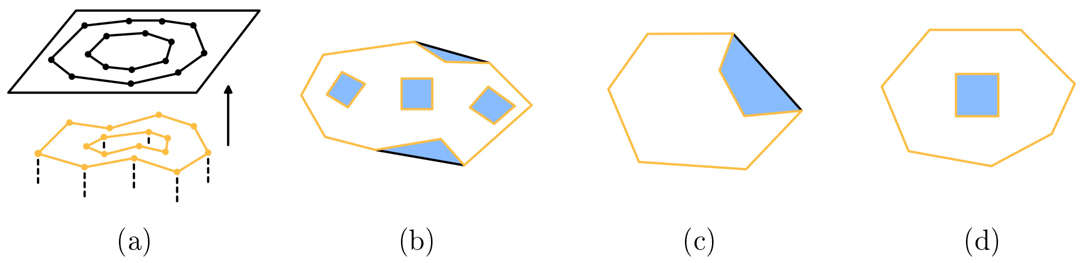


Figure 4.8: (a) Mapping of representation of cycles to proxy plane preserving topology. (b)-(d) Decision on segment consisting of triangles counting as final face. The cycle edges are displayed in orange and edges caused by the (convex) triangulation in black. The blue (disjoint) segments do not provide the final faces as their boundary edges do not agree with the whole set of cycle edges, yet the orange bounded white areas do.

faces per region. Figure 4.7 demonstrates the whole pipeline. Suppose we are given a complex $K = \mathcal{V} \cup \mathcal{E} \cup \mathcal{F}$ representing a segmentation. Then according to Section 4.4.1 we obtain vertices for the simplified mesh using the respective regions contributing to a vertex $v \in \mathcal{V}$, i.e. the indices in $\Psi^{-1}(v) \in \mathcal{I}$.

As each region shall reflect a face we collect the corresponding cycles in $\mathcal{F}$, i.e. all which include a vertex in $\mathcal{V}$ for which its pre-image under Ψ^{-1} includes the region's index, see Section 3. These cycles are mapped to the linear proxy associated to the region in such a way that the topology of each cycle is preserved, see Figure 4.8(a). Afterwards we apply a constrained triangulation in the proxy plane, i.e. a triangulation in 2d respecting the edges of the mapped cycles.

As we might have multiple cycles bounding a regions' face, we need to decide which triangles belong to the face. It is even the case for just one cycle, as the triangulation yields a convex set but the face in question does not need to be convex. The cycle information now provides a possibility to split the triangles into segments, i.e. a segment is bounded by the cycles, see Figure 4.8(b)-4.8(d). As our face consists of a single component possibly including several holes we select the one segment whose boundary edges are equal to the set of all cycle edges.

Finally we pull back the triangles belonging to the segment of the final face to the related vertices in $\mathbb{R}^3$. Applied to every region gives us the final simplified mesh.

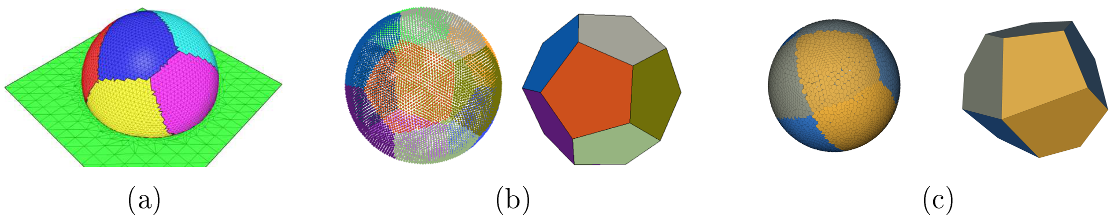


Figure 4.9: A visual comparison of the output of (a) [CSAD04] showing a segmentation of the half-sphere into six proxies, (b) [LB16] with a segmentation of the sphere into 12 proxies, and (c) the results of our algorithm applied to the sphere deducing 12 proxies.

4.4.3 Experimental Results

In the following, we present different experiments regarding the simplification as obtained from the proxy segmentation. Each experiment addresses different aspects of the simplification pipeline. First, we consider how the parameter η influences the obtained simplification. Next, on the one hand we turn to the Fandisk model, to discuss difficulties arising due to face generation in terms of star-convex faces and on the other hand we present results dealing with general faces. The last experiment deals with a noisy geometry and robustness of our algorithm.

Throughout our experiments, in order to solve the minimization problem in Equation (4.12), we turn to the build-in solver of Matlab. Note that the minimization problem has a non-linear target function with non-linear constraints and can thus not be solved by any LP or even ILP solver. Hence, we follow the example from the Matlab manual [MM], which is supported by all versions newer than *R2019b*[7]. The solver asks for starting points from which to run the optimization. We initialize the normals $\tilde{n}_i$ by the proxy normals N_i. As first guesses for any intersection point x_j, we chose the barycenter of the centers C_i of those proxies that contribute to this intersection.

Threshold η-dependency on the Sphere Model Our first simplification model is a sphere sampled with $\aleph = 5,122$ points. We chose this model as it also appears in [CSAD04, LB16]. By running our algorithm with 12 initial centers without split and merge we obtain a segmentation into 12 planar faces, shown together with the simplification done by optimization in Figure 4.9 coupled with results of [CSAD04, LB16].

In Figure 4.10, we show segmentation and simplification results w.r.t. different values of η taken from $\{500, 200, 100, 50, 25\}$. The utilized geometry is the sphere used above. The simplification points are calculated via both approaches introduced in Section 4.4, i.e. via intersection of the proxy planes and via the optimization problem given in Equation (4.12). All results are obtained from the same set of six randomly selected seed points. For the optimization case $\eta = 500$ we increased the weight $\tilde{w}_i = 3$, as otherwise the simplification points would have produced a smaller version of the cube. Hence, in this case we forced the optimization putting more emphasis on less proxy normal deviation. Figure 4.10 also shows the amount of final proxies in correspondence to the chosen value of η. It is not surprising that

[7]The reference directs to the updated version of release R2023a compared to the one mentioned in [SZP20].

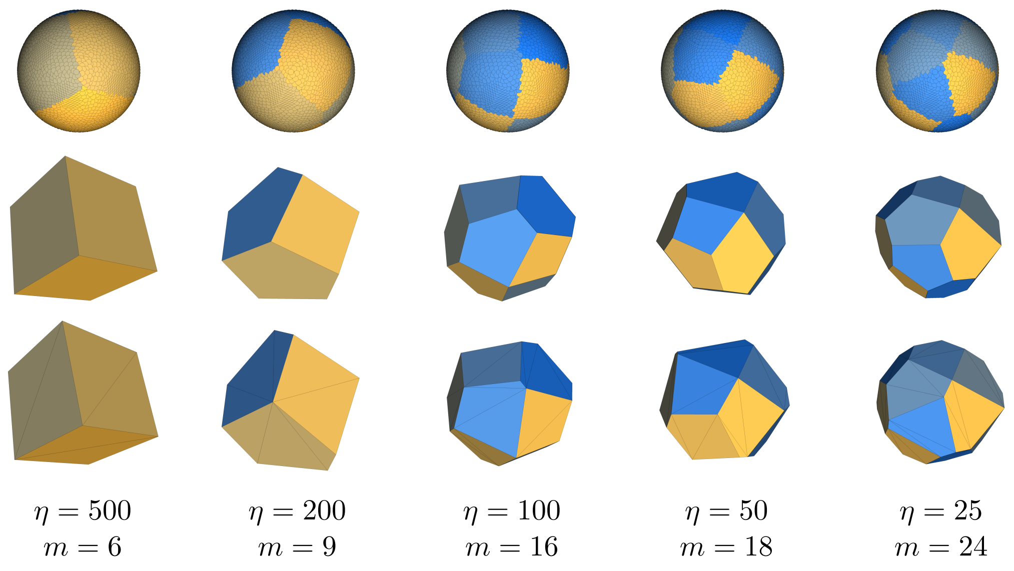


Figure 4.10: The segmentation and simplification for different η values. The first row shows the segmented point sets, the second and third rows the meshes deduced via optimization and plane intersection, respectively.

with a decreasing number of η, the number of proxies increases as this decreases the error measure (4.5) in order to meet the prescribed threshold. Note further that the resulting meshes contain vertices where more than three proxies meet, see the fourth and fifth column in Figure 4.10. While it is not problematic in this case, it does cause problems for a different model, see Section 4.4.3.

Face Reconstruction on the Fandisk Model We proceed to discuss a more involved geometry, namely the Fandisk model (CAD) with $\aleph = 38,840$ vertices, shown in Figure 4.11. Here, we started the segmentation with 36 manually selected seeds, $\eta = 75$, and without using splits or merges. We consequently obtained $m = 36$ proxy regions, shown in Figure 4.11(a).

Reconstructing this model is challenging to our algorithm in two aspects. First, our simplification procedure could use star-convex faces, see Section 4.4.2. However, the orange front plate of the Fandisk model is not star-convex with respect to its barycenter (Figure 4.6(b), bottom) and thus a first automatic reconstruction is slightly faulty (Figure 4.11(b)). These errors are easily identified and fixed by assigning a correct order to the contributing face vertices. The second challenging aspect is caused by the sensitivity of neighborhood notions for different densities in the point set. For example, the blue region fits between the dark-blue and gray and hence, they see each other (Figure 4.11(c), right mark). However, their planes are almost parallel, and so their intersection appears as an outlier. This could be avoided, if we forbid intersection points built by almost parallel planes or if we forbid those intersections that lie too far away from either one of the proxies. The behavior of a misplaced intersection point is also the case for the one produced by the beige, blue, and dark-blue proxies (Figure 4.11(b) left mark), whereas in consequence a gap results between the blue and gray areas, which should not be there, according to the segmentation. As before, we manually removed faulty intersection points and reset face incidences to obtain a clean mesh for visual representation (Figure 4.11(d)).

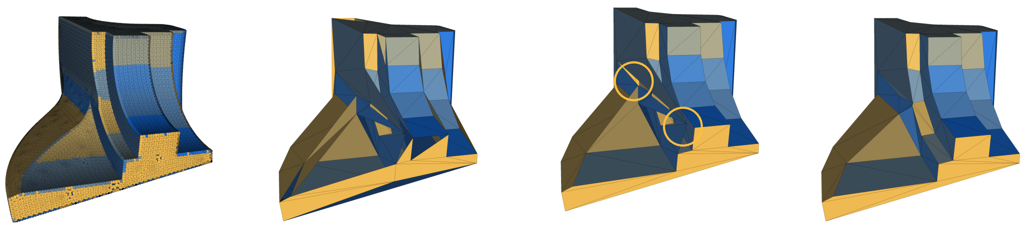

(a) Initial Segmentation
(b) Faulty Vertex Order in Faces
(c) Faulty Proxy Intersection Points
(d) Corrected Simplification

Figure 4.11: Segmentation and simplification (plane intersection) of the Fandisk.

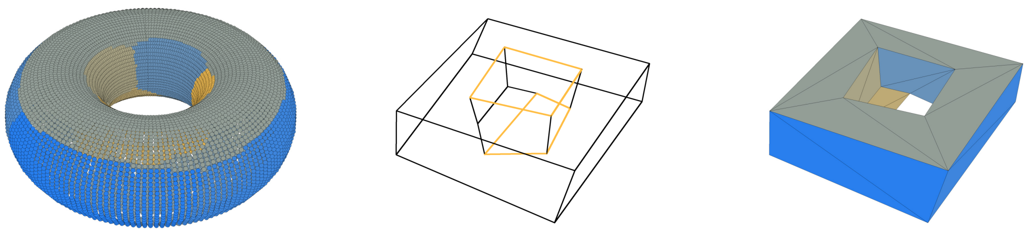


Figure 4.12: Left: Segmentation result from 11 manually selected seeds. Center: The complex' graph representation with the top and bottom cycle highlighted in orange. Right: Simplification obtained by the complex information.

Note that these challenges are unique to the setting of point sets. In the context of meshed geometries, the intersection vertices can simply be ordered along the boundary of their respective proxy region, yielding a feasible face. Also, neighborhood relations in the mesh setting can be computed via shared edges and do not require an additional neighborhood parameter k. Hence, the works of [CSAD04, YLW06] did not have to tackle these issues, while the work of [LB16] does not contain any description of how they solved these problems.

Face Reconstruction for General Faces For the reconstruction of general faces we used three geometries to showcase two different directions. The first one uses the segmentation of the point set on a Torus model with $\aleph = 13,503$ vertices with 11 manually selected seeds and without splitting or merging see Figure 4.12. A correct determination of cloud neighborhoods is crucial to identify the cycles bounding faces, where in this example a fairly large number of points and a manual seed selection supports our goal. But with it we are able to generate faces with non-trivial first fundamental group, i.e. punctured regions, as can be seen in the top face of the simplification shown in Figure 4.12.

The second direction shows two simplification results on manually selected regions on a 2-torus model ($\aleph = 3,838$) and the Fandisk model ($\aleph = 6,475$), see Figure 4.13. The coloring of the region representation is chosen w.r.t. Example 2.

All of the simplification results are simplicial pseudo-surfaces. In our case they don't even have non-trivial intersection, i.e. only in their faces.

Robustness against Noise on the Dodecahedron Model As our final experiment, we consider the simplification of a Dodecahedron model equipped with

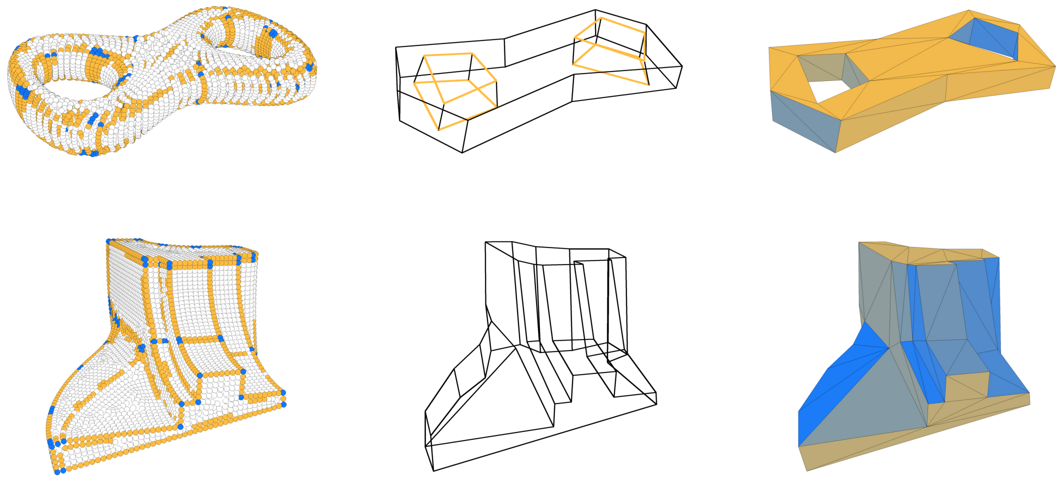


Figure 4.13: Shows face generation examples on 2-torus (top row) and Fandisk (bottom row). Left column: Region representation of manually selected regions. Center column: The complex' graph representation (top row shows top and bottom cycles highlighted in orange). Right column: Simplification obtained by the complex information.

Gaussian noise in normal direction with an amplitude of 25% of the average neighbor distance (taken as averaged sum over all points and their 12 nearest neighbors.). This geometry is not easily translated into a clean mesh and therefore, the methods of [CSAD04, YLW06] cannot be applied here straightforward. The model consists of $\aleph = 962$ and we started with 12 randomly chosen seeds and a threshold of $\eta = 50$. Here, in contrast to the other experiments, we use a neighborhood size of $k = 12$, because of the involved noise components. The otherwise used value of $k = 8$ caused points to not be associated to any proxies. Furthermore, we allowed for splits and merges. The algorithm converged after 8 iterations with $m = 11$ final proxies, see Figure 4.14. Observe that the faces reflecting the top proxy in the third image are not planar, which is a possible occurrence outlined in the intersection of planes when finding the simplified mesh vertices. In contrast, the optimization provides planar patches (rightmost image).

The segmentation reflects the different parts of the geometry correctly. This probably results from the normal differences caused by the noise still being smaller than the normal differences between the different faces of the Dodecahedron. Hence, if the noise level and its influence in normal deviation still lies beyond the normal deviation of neighboring geometry regions, its segmentation will reflect the geometric structure well. However, this still depends on initial seed placements and therefore also on performing splits and merges.

With a segmentation reflecting the correct structure of the geometry, the simplification should not cause any additional issues, as it is the result of proxy plane intersections. Only the neighborhood relation between proxies might be more involved, as point locations now deviate more because of additional noise components.

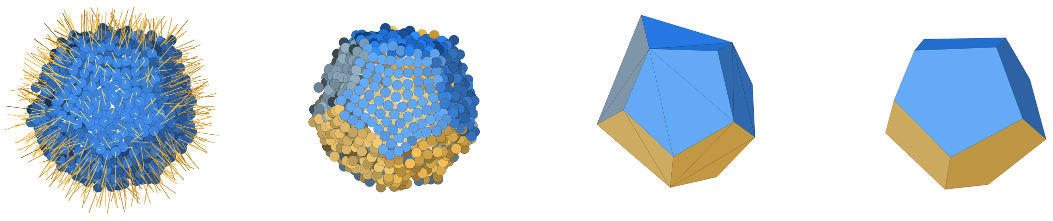


Figure 4.14: Noisy Dodecahedron, its segmentation and simplifications (planar intersection and optimization).

4.5 Conclusion and Further Research

We have shown that variational shape approximation is an effective approach to obtain a simplified mesh not only from meshed input, but also from geometries sampled by point sets. The presented example for non-convergence of the VSA method as used in [CSAD04, YLW06, LB16] was successfully circumvented by the introduction of a new *switch* operation for which we proved convergence. Furthermore, by two more operations in the pipeline, namely *split* and *merge*, we eliminate the dependency of both the number and placement of initial seed points. Finally, we give a detailed description on how to obtain a simplified mesh from the segmented point set by building on the method of tangent plane intersection as well as constructing faces for regions which are not necessarily star convex using an underlying complex. This addressed the third direction of future work outlined in [SZP20], namely that up to this point only star-convex regions could be handled. The face generation however raises a more basic challenge, i.e. the determination of neighborhoods of regions outlined in Remark 6. But as this is the same problem points and their lack of connectivity entail, it remains an ill posed one. Other open directions are listed in the following.

- On a theoretical level, we have shown that the introduction of the *switch* operation results in guaranteed convergence. However, it remains unclear whether other alterations of the pipeline exist that came with the same result and do not affect the runtime of the algorithm as heavily as the *switch* operation does.

- Concerning the parameters, we currently do not provide any theoretical reasons for the choice of weights ω_j in Equation (4.4) or weights $\tilde{w}_j$ in Equation (4.11). A better understanding of these weights, aside from the experimental values used in the work, is desirable. Similarly, the sum of normal differences η is not directly related to the curvature of a proxy, as it depends on the number of points contributing to the sum. Here, a threshold should be found that is independent of the number of points.

- We have presented an experiment on a noisy point set. We assume that the treatment of meshes equipped with noise should be equally possible and yield even better results because of the explicit connectivity. To investigate this behavior is also left as future work.

Chapter C

Inside a Cloud

So far we investigated clouds on a macroscopic level and how they interact to form larger systems with certain properties. In this chapter we are going to discuss some aspects inside a cloud, i.e. a microscopic viewpoint. As a cloud consists of multiple instances, in our case points, we can explore the points' behavior inside the cloud to obtain insights or guide contributions to certain tasks.

A cloud could be perceived as the neighborhood of the points it contains, not only from a topological point of view[1], but also from the discussion of how a local perspective and the ill posed problem of point neighborhoods lead to the approach of selecting a neighborhood for each point, see Chapter A. These are then used to steer certain processes for that specific point. Clouds are therefore versatile, as they might contain a subset for each point reflecting its local vicinity or they contain way less sets, such that points rely on the same cloud.

Something we can make use of is the distribution of points inside a cloud where the points are possibly equipped with normal information. This distribution can be evaluated via Principal Component Analysis (PCA) (cf. [WJM14]). For example a cloud reflecting the vicinity of a point might appear planar and thus the point could be treated differently compared to a volumetric extended neighborhood. We are researching the influence of sigmoid-weighted point normals on the extend of point distributions—which we then call normal weighted neighborhoods—, obtained via PCA in Section 5 in a large scale experiment, which is published, see [SZ21]. We included minor changes in text and layout to improve the readability.

The necessity of point normals in the normal weighted neighborhoods and the fact that certain layouts got favored brought up the question whether we could gain insights solely on the point distribution and its PCA (leaving out point normals) with an entity which reflects a desired surface-like appearance. These considerations are covered in Section 6.

[1]In the sense of open sets a neighborhood of a point is an open set (w.r.t. the spaces' topology) containing that point, see [Lee00].

5 Normal Weighted Neighborhoods

Many definitions of point neighborhoods, combinatorial or geometric, with global or local parameters, have been proposed and discussed (see Section 5.1). Furthermore, the concept of weighting neighboring points is not new. For example, the pure selection of a neighborhood causes an equal treatment of all neighbors. Aside from this, isotropic weighting is one common way, evaluating Euclidean distances via a Gaussian weighting function. This provides closer points with higher influence (see, e.g., [ABCO$^+$01]). Additionally, other point set information can be incorporated, like density or distribution (see, e.g., [PLL12] or [SJP18]). The inclusion of normal deviation in the area of anisotropic weighting has also been considered and discussed before (see [YRS$^+$18, Skr19]).

The research work presented here aims at investigating anisotropic weighting terms in a broad framework, which includes usual weighting choices such as equal weights or sharp cut-off weights[2] (Section 5.2). Our evaluation is processed via a Shannon entropy model (Section 5.3), which is based on the work of [DMDV11, WJM14]. Furthermore, we aim at evaluating the weighting scheme on a large scale. This is to prevent over-interpretation of findings obtained from a very small set of models. Overall, the contributions of this work are:

- Definition of a shape-aware neighborhood weighting utilizing sigmoid function weights based on normal variation.
- Presentation of a Shannon entropy evaluation model that can be proven to be non-degenerate on our inputs.
- Large scale experimental evaluation of the proposed neighborhood weighting concept.
- Discussion of the results with respect to both neighborhood weighting and neighborhood sizes.

While the content of this work is deeply routed in the field of traditional computer-aided design, in our concluding Section 5.5, we will provide an outlook and several thoughts on the application of the presented techniques within the context of machine learning.

5.1 Related Work

Neighborhoods are very important in point set processing, as almost all algorithmic approaches rely on them. A common choice is to use heuristics to determine sufficient notions like the size of a combinatorial or metric neighborhood. In the following, we recall works discussing heuristic neighborhood definitions. Several works have advanced from simple heuristics and derive more involved notions for better fitting neighborhood definitions in different contexts. These are mainly obtained from error functionals, which we will also discuss.

[2]We consider the case of cut-off weights if starting from a given deviation, all points with greater or equal deviation are attributed weight 0.

5.1.1 Heuristics

Most works consider either a combinatorial k-nearest neighborhood $\mathcal{N}_k(\cdot)$ or a metric ball $B_r(\cdot)$ inducing a neighborhood, see Examples 1 iv) and v). Both of these notions have parameters to be tuned, namely the number of neighbors k or the radius r of the neighborhood. Several works have been presented introducing heuristics to find appropriate values for k or r in different scenarios. The authors of [ABCO$^+$01] for instance use a global radius and change it to affect the running time of their algorithm. In [PGK02], the authors fix a combinatorial number k of neighbors to be sought. Then, for each point p_i from the considered point set P, these k neighbors are found, which fixes a radius r_i to the farthest of them. Finally, the neighbors within radius $r_i/3$ are used. Therefore, their approach resembles the geometric neighborhood in a local manner.

The method used in [PKKG03] is more involved. The authors recognize that both a too large or too small radius r lead to problems and thus aim for a local adaption like [PGK02]. A local density estimate δ_i around each point $p_i \in P$ is computed from the smallest ball centered at p_i, containing $\mathcal{N}_k(p_i)$, where k is found experimentally to be best chosen from $\{6, \dots, 20\} \subset \mathbb{N}$. Given the radius r_i of this ball, the local density is set to be $\delta_i = k/r_i^2$. In a second step, a smooth density function δ is interpolated from the local density estimates δ_i, hence this weighting involves the incorporation of density-information into the weight assignment.

In the context of surface reconstruction, the authors of [FR01] discuss several choices for neighborhoods and corresponding weights. While two of the three presented methods simply use geometric neighborhoods, the third method takes a different approach. Namely, the authors collect all neighbors of p_i in a "large" ball ([FR01, page 7]) around p_i. Then, they fit a plane to this preliminary neighborhood and project all neighbors and p_i onto this plane. On the projections, a Delaunay triangulation is built and the induced neighborhood of the triangulation is used in the following computations, which localizes their approach and respects different point distributions.

A completely different route is taken by [BL12]. The authors first calculate features of a point set based on differently sized neighborhoods. Then, they use a training procedure to find the combination of neighborhood sizes that provides the best separation of different feature classes.

The inclusion of normal deviation and hence anisotropic weighting into neighborhood concepts is part of the work [YRS$^+$18]. The approach of the authors is to use a weighted principal component analysis, which fits our evaluation model. However, they rely on a global neighborhood size and assign sharp cut-off weights while we allow for changing neighborhood sizes and smooth weighting terms.

5.1.2 Error Functionals

While the approaches presented above are based on heuristics, some works try to deduce an optimal k for the k nearest neighborhoods based on error functions. For instance, the authors of [LCOL06] work in the context of the Moving Least Squares (MLS) framework (see [ABCO$^+$01, Lev98, Lev04, SL16]) for function approximation. The authors perform an extensive error analysis to quantify the approximation error both independent and depending on the given data. Finally, they obtain an error functional. This is then evaluated for different neighborhood sizes k. The neighborhood $\mathcal{N}_k$ yielding the smallest error is finally chosen to be used in the actual MLS approximation.

In contrast, the authors of [MNG04] deduce an error bound on the normal estimation obtained from different neighborhood sizes. Utilizing this error functional, they obtain the best suited neighborhood size for normal computation. The work of [LCOL06] heavily depends on the MLS framework in which the error analysis is deduced, while the work of [MNG04] depends on the framework of normal computation.

The authors of [WJM14] take a more general approach in the context of segmentation of 3D point sets. They also use the concept of combinatorial neighborhoods, going back to results of [LP01, DMDV11]. In order to choose an optimal value for k, the authors turn to the covariance matrix, which is symmetric and positive-semi-definite. Thus, the matrix has three non-negative eigenvalues. Following an idea of [HDD$^+$92], in the work of [PKKG03], the authors grow a neighborhood and consider a surface variation as a measure to grow a neighborhood around each point p_i. The same quantity is used by [BL06]. However, the authors of [PKKG03] do not grow a neighborhood, but choose a size k for it according to a consistent curvature level. The authors of [WJM14] do not stop at these information, but proceed to consider three more quantities derived from the eigenvalues of the covariance matrix reflecting point set features, see [DMDV11, WJM14]. Afterwards, following the concept of entropy by Shannon [Sha48], they evaluate combinatorial and geometric neighborhood sizes via two error measures (see Section 5.3 for a detailed discussion).

5.2 Sigmoids

In contrast to the works listed above, our approach aims at integrating the shape of the geometry, i.e., the normal information, into the neighborhood definition. We will do so by enriching a given combinatorial neighborhood with a set of weights that are dependent on the normal variation within the neighborhood. To ensure a smooth transition of weights, we apply a sigmoid function to the angle deviation of the normals.
Given a set of points $P = \{p_i \mid i \in [n]\}$, $n \in \mathbb{N}$ and $[n] = \{1, 2, \ldots, n\}$, corresponding oriented unit-length normals $n_i \in \mathbb{S}^2$, and local neighborhoods $\mathcal{N}_i \subset [n]$ for every $i \in [n]$, see Example 1 v). For a given weighting function

$$\phi : [0, \pi] \to [0, 1], \tag{5.1}$$

we obtain the following weights

$$w_{ij} = \phi\left(\angle(n_i, n_j)\right) \qquad \text{for } i \in [n],\ j \in \mathcal{N}_i. \tag{5.2}$$

The argument of ϕ is the deviation of the normals measured by their angle, which ranges from 0 to π. We turn to this formulation, because it has an obvious geometric interpretation. In order to have an efficient implementation of the presented techniques, the scenario can be reformulated in terms of the scalar product of the normals, which avoids the costly computation of arccos.

Note that by the symmetry of the angle, the weights are symmetric, i.e., $w_{ij} = w_{ji}$. The weighting function ϕ shall assign non-negative weights between 0 and 1. These weights should correspond to the similarity of the corresponding normals, i.e., a

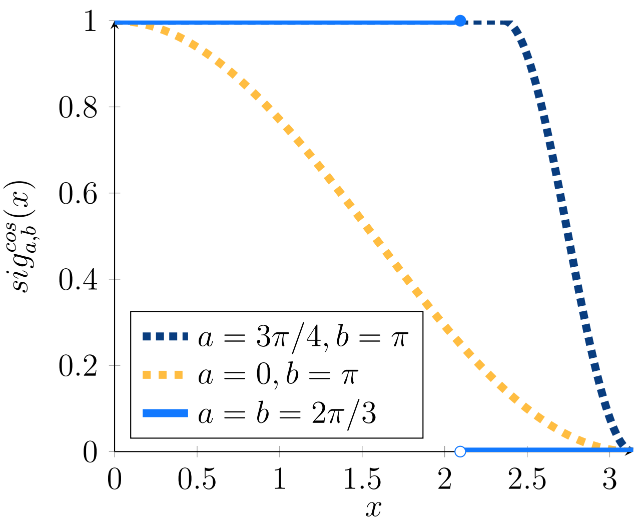


Figure 5.1: Plots of the sigmoid $sig^{cos}_{a,b}(x)$ for three parameter choices on the domain $[0, \pi]$.

small angle should result in weights close to or equal 1, while a large angle should yield weights close to or equal 0.

Our choice for the weighting function is a sigmoid. A sigmoid function is visually characterized by its shape of an "S"-curve, even though it is mirrored in our scenario, see Figure 5.1. We will consider a family of sigmoid functions that provide different interpolations between 1 and 0. The family is based on the trigonometric cosine function. It is related to the sigmoid used in [MGMAI06], however, we alter it to be a monotonic falling curve between 1 and 0 on the interval $[0, \pi]$.

Definition 15 (Cosine-Sigmoid). *Consider two given thresholds $a \in [0, \pi]$ and $b \in [a, \pi]$. Then, we define the sigmoid weighting function $\mathrm{sig}^{\cos}_{a,b} : [0, \pi] \to [0, 1]$ as*

$$x \mapsto \begin{cases} 1 & \text{if} \quad 0 \leq x < a \\ \frac{1}{2}\cos\left(\frac{\pi(x-a)}{b-a}\right) + \frac{1}{2} & \text{if} \quad a \leq x \leq b \\ 0 & \text{if} \quad b < x \leq \pi. \end{cases} \tag{5.3}$$

Note that for $a \neq b$ this function is $\mathcal{C}^1$ and smoothly transitions from 1 to 0. In particular, both boundary values are included, i.e., points can be given both weights 1 and 0, which corresponds to fully taking them into account or to not taking them into account at all. The threshold parameter $a \in [0, \pi]$ translates the curve along the x-axis and controls where the cosine curve starts. Similarly, the threshold parameter $b \in [a, \pi]$ controls where the cosine curve ends, i.e., the curve's decline is controlled by the distance between these two thresholds. In particular, when choosing $a = b = \pi$, all inputs obtain uniform weight 1 while for $a = b$, the function models a sharp cut-off at the chosen threshold. This allows us to relate our weights to the uniform weights used in [WJM14] and to the sharp cut-off weights of [YRS+18], respectively.

5.3 Evaluation Model

Having presented the set of neighborhood weights in Equation (5.2) and the corresponding weighting function in Equation (5.3) in the previous section, we will now describe the mathematical background of our evaluation process. For this, we turn to the information measures originally introduced by Shannon [Sha48]. Specifically, we will use a variation of the quantities derived in [DMDV11, WJM14] as we will present in Section 5.3.1. First, we will establish the necessary notation and preliminary results.

Consider the covariance matrices $C_i \in \mathbb{R}^{3\times 3}$ given by

$$C_i := \sum_{j\in\mathcal{N}_i} w_{ij}(p_j - \bar{p}_i)(p_j - \bar{p}_i)^T, \tag{5.4}$$

with $i \in [n]$, where $\bar{p}_i = \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} p_j$ is the barycenter of the neighborhood of p_i, thus $(p_j - \bar{p}_i)$ is a column vector in $\mathbb{R}^3$ and $(p_j - \bar{p}_i)^T \in \mathbb{R}^{1\times 3}$ its transpose, i.e., a row-vector. The weights w_{ij} are chosen according to Equation (5.2). The covariance matrix C_i is symmetric and positive-semi-definite. Thus, it has three non-negative eigenvalues, which in the following we will denote by

$$\lambda_i^1 \geq \lambda_i^2 \geq \lambda_i^3 \geq 0. \tag{5.5}$$

Depending on the neighborhood $\mathcal{N}_i$ and the assigned weights w_{ij}, we can prove the following theorem about the covariance matrix C_i.

Proposition 7 (Non-degenerate Covariance Matrix)**.** *For $P = \{p_i \mid i \in [n]\}$ a set of points, fix a point $p_i \in P$ and its neighborhood $\mathcal{N}_i \subseteq [n]$, and consider the function $\mathrm{sig}_{a,b}^{\cos}$ from Equation (5.3) as well as the covariance matrix C_i given in Equation (5.4). Assume there are $\ell_1, \ell_2 \in \mathcal{N}_i$, $\ell_1 \neq \ell_2$ such that $p_{\ell_1} \neq p_{\ell_2}$ and $n_{\ell_1} \neq -n_{\ell_2}$. Then, for a value $b \in [a, \pi]$, the sum of all eigenvalues of C_i is strictly positive, independent of the choice of $a \in [0, \pi]$.*

Note that a non-degenerate covariance matrix can trivially be obtained by setting $a = \pi$. However, the proposition makes an even stronger statement, namely that degeneracy can be obtained independent from the choice of a. Its proof follows from the observation that the weights w_{ij} are non-negative, as are all eigenvalues of C_i since C_i is positive semi-definite. Thus, the sum of the eigenvalues is 0 if and only if all eigenvalues are. By a case distinction on the zero set of the function $\mathrm{sig}_{a,b}^{\cos}$, we can then prove that there exists a value $b \in [a, \pi]$ which results in strictly positive weights, which proves the proposition.

5.3.1 Non-Degenerate Covariance Matrix

Given the assumptions of Proposition 7, we can assume that $C_i \neq 0 \in \mathbb{R}^{3\times 3}$. Therefore, we can derive certain quantities from the eigenvalues of the covariance matrix. In our context, we will consider the linearity L_λ, planarity P_λ, and scattering S_λ. These are given by

$$L_i^\lambda = \frac{\lambda_i^1 - \lambda_i^2}{\lambda_i^1}, \qquad P_i^\lambda = \frac{\lambda_i^2 - \lambda_i^3}{\lambda_i^1}, \qquad S_i^\lambda = \frac{\lambda_i^3}{\lambda_i^1} \tag{5.6}$$

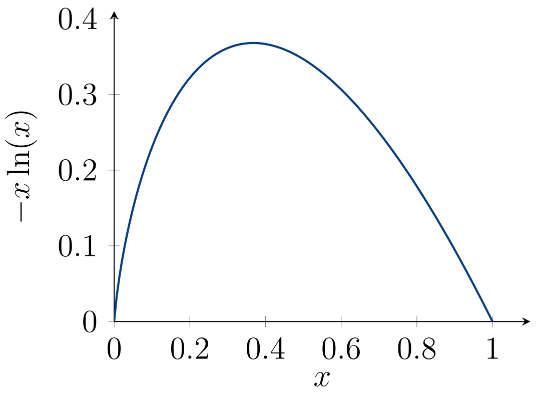


Figure 5.2: Plot of the summand $-x\ln(x)$ from Equation (5.7) for $x \in [0,1]$ as all arguments L_i^λ, P_i^λ, and S_i^λ are taken from $[0,1]$.

and represent 1D, 2D, and 3D features in the point set, respectively. See [DMDV11] for a derivation and a detailed explanation of these quantities. As $C_i \neq 0$, we have $\lambda_i^1 \neq 0$, therefore the quantities in Equation (5.6) are well-defined. Furthermore, because of the ordering of the eigenvalues given in Equation (5.5), we have $L_i^\lambda, P_i^\lambda, S_i^\lambda \in [0,1]$. Hence, as

$$L_i^\lambda + P_i^\lambda + S_i^\lambda = 1,$$

each of these three quantities can be interpreted as the probability of the considered point to be part of an intrinsic 1D, 2D, or 3D part of the geometry. The authors of [DMDV11, WJM14] consider the error

$$E_i^{\text{dim}} = -L_i^\lambda \ln(L_i^\lambda) - P_i^\lambda \ln(P_i^\lambda) - S_i^\lambda \ln(S_i^\lambda). \tag{5.7}$$

See Figure 5.2 for a plot of each summand of the equation. Note that while $\lim_{x\to 0} \ln(x) = \infty$ we have $\lim_{x\to 0} x\ln(x) = 0$, which follows from rewriting it as quotient and applying L'Hôpital's rule. Practically, the error measure E_i^{dim} assesses to what extent the neighborhood $\mathcal{N}_i$ indicates a corner, an edge point, or a planar point of the geometry. In particular, the extreme cases

$$(\lambda_i^1, \lambda_i^2, \lambda_i^3) \in \{(\rho,0,0), (\rho,\rho,0), (\rho,\rho,\rho) \mid \rho \in \mathbb{R}_{>0}\} \tag{5.8}$$

all obtain $E_i^{\text{dim}} = 0$. That is to say that if a point p_i can be clearly classified as part of a linear, planar, or scattered segment of the point cloud, the classification error E_i^{dim} will indicate this.

Note that in general applications, these extreme cases are unlikely to occur. In particular in the presence of noise, the quantities L_i^λ, P_i^λ, and S_i^λ will generally not satisfy Equation (5.8). Thus, the classification error E_i^{dim} will be larger and therefore indicate that the point could not clearly be classified as part of a linear, planar or scattered segment of the point cloud.

We will use the classification error (5.7) in our quantitative experiments in Section 5.4. Aiming for as-clear-as-possible classification of points, we pursue as-small-as-possible values of E^{dim}. However, the above discussion depends on the assumptions provided in Proposition 7. In the following we will discuss cases in which these assumptions are not satisfied.

5.3.2 Degenerate Covariance Matrix

In practical applications, the assumptions of Proposition 7 are not always satisfied. Note here that the classification error E_i^{dim} is evaluated on a single point p_i of the point set P. The following reasons can hinder the correct evaluation:

i) If the point set contains multiple duplicates of a point, more than the sought-for number of neighbors k, all points in the reported neighborhood collapse into a single point equal to the barycenter of the neighborhood. Thus, C_i becomes 0.

ii) If a point p_i has a flipped normal in comparison to all its neighboring points p_j, the argument x in the weight equation $w_{ij} = \mathrm{sig}_{a,b}^{\cos}(x)$ becomes π and therefore, all weights degenerate to 0. This happens in particular for very small or thin geometries as well as for faulty normal fields.

iii) Even if the assumptions of Proposition 7 are satisfied, it only states the existence of a suitable parameter $b \in [a, \pi]$. Therefore, choosing parameter b too small can cause all weights in the covariance matrix (5.4) to degenerate to 0.

In the following evaluation, we prevent case *i)* by requiring the point sets to only contain distinct points. Furthermore, we orient the normal field to prevent case *ii)*. Concerning a too small parameter b, we report a failure in the computation of the error values for the point p_i if $\sum_{\ell=1}^{3} \lambda_i^\ell = 0$. By including the choice $a = \pi$ for the parameters, we ensure that each model has at least one correctly evaluated error value E_i^{dim} at each point $p_i \in P$.

5.4 Experimental Results

In this section, we present our quantitative evaluation of the weights presented in Equation (5.2). For the evaluation, we utilize the classification error E^{dim} as defined in Equation (5.7). Our clean models are taken from a data set described in [HZG+18]. The authors provide ten thousand clean and manifold surface meshes, which are obtained by exporting only the boundary of the tetrahedral meshes used in [HZG+18]. From these, we randomly select a subset of 1,000 meshes with uniform probability. Furthermore, we use 100 meshed models each from the real-world object scans provided by [CZMK16] and by [BRLB14]. Finally, to test the scalability of our approach, we also include the model “Pan et Oursons” from [Lar12].

For all these models, we use the mesh information and its manifold property to obtain oriented face normals. From these, we compute vertex normals and then use these and the vertices as point sets for our experiments. For each such point set P, we consider the parameter sets

$$\mathfrak{A} := \mathfrak{B} := \left\{0, \frac{\pi}{6}, \frac{\pi}{3}, \frac{\pi}{2}, \frac{2\pi}{3}, \frac{5\pi}{6}, \pi\right\}$$

for the choice of a and b, respectively, where we ensure that $a \leq b$. We choose this range as a reasonable trade-off between complexity of the experiments and exploration of the parameter space. Note that in the application scenario at the end, we consider parameter values that are reasoned there. We use the combinatorial neighborhood notion, so that for every pair $(a, b) \in \mathfrak{A} \times \mathfrak{B}$ and every point $p_i \in P$, we calculate its E_i^{dim} value over the range of k, taken from

$$\mathfrak{K} := \{6, \dots, 20\}.$$

We assume this range for k, as it reflects typical, heuristic choices for neighborhood sizes in the area of point set processing, see the works discussed in Section 5.1, in particular [PKKG03]. For each point p_i in each point set P, we obtain an optimal parameter triple (a_i^*, b_i^*, k_i^*) as

$$(a_i^*, b_i^*, k_i^*) = \underset{(a,b,k)\in\mathfrak{A}\times\mathfrak{B}\times\mathfrak{K}}{\arg\min} E_i^{\text{dim}}. \tag{5.9}$$

Following the discussion from Section 5.3.2, we set $E_i^{\text{dim}} = \infty$ if the covariance matrix C_i for the point $p_i \in P$ degenerates for all choices $(a, b, k) \in \mathfrak{A} \times \mathfrak{B} \times \mathfrak{K}$.

See Figure 5.3 for an illustration of the classification error E^{dim} on the Fandisk geometry as well as for a comparison of different parameter choices (a, b). The top row shows the classification error E_i^{dim} from Equation (5.7) on each point of the geometry, colored from low error to large error. Note that when fixing parameters (a, b), it is possible that the covariance matrix C_i degenerates for every choice $k \in \mathfrak{K}$. This happens for the specific choice $a = b = \arccos(0.9)$ as used in [YRS+18]. We have colored the respective points red. The optimal triple from Equation (5.9) achieves significantly lower classification error E_i^{dim} than the equal weights of [WJM14] or the cut-off weights of [YRS+18]. The observed fluctuation in planar areas is due to (a) the utilization of combinatorial neighborhoods, which do not always provide symmetrically shaped neighborhoods on a synthetic geometry like the Fandisk, as well as to (b) the sensitivity of E_i^{dim} to slight changes in the covariance matrix.

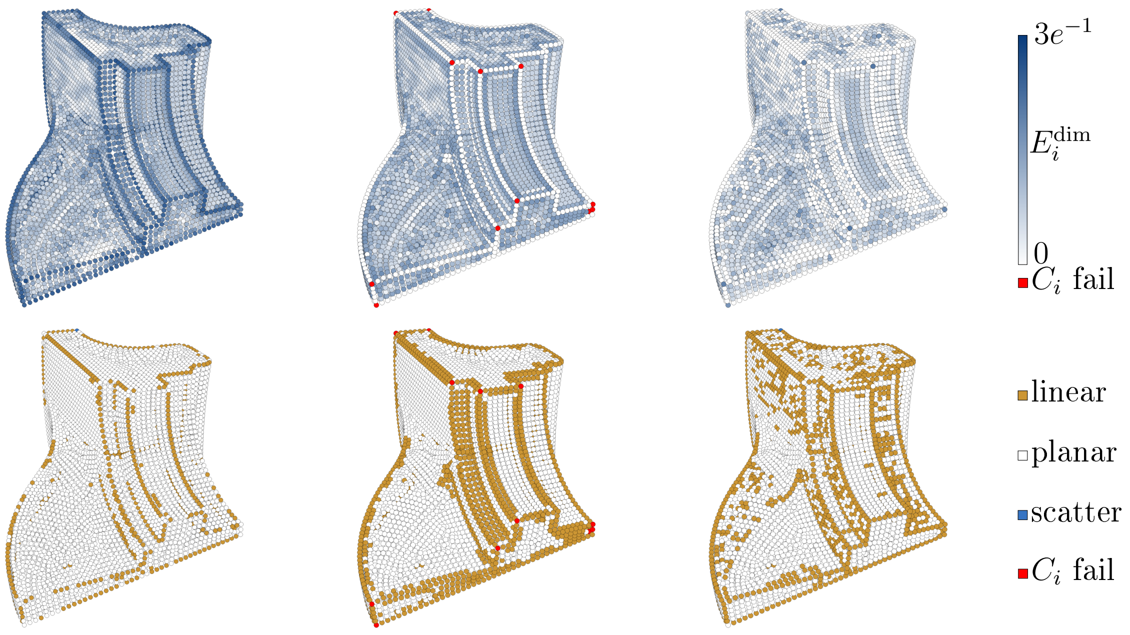
The bottom row of Figure 5.3 shows a feature classification according to the maximum value out of linearity, planarity, and scattering as defined in Equation (5.6). Note how the equal weights of [WJM14] classify almost all elements as planar and fail to identify edge structures. In contrast, the cut-off weights of [YRS+18] identify all edges, but over-pronounce them. The optimal weight choice from (5.9) takes a middle ground between these two extremes, on the cost of identifying several clearly planar points as linear. Again, this stems to a certain extend from processing a synthetic geometry. Observe that the equal weights and the optimal weights do identify scattered points (one example being the topmost corner of Fandisk) while the cut-off weights rather fail to create a covariance matrix at corner points.

The images in Figure 5.3 summarize our following experiments. In order to compare with the findings of [WJM14], we compute the classification error E_i^{dim} for each point of every point set of the three chosen model repositories [HZG+18, BRLB14, CZMK16] as well as of the single, large model from [Lar12].

In the following we report and interpret our findings.

Global (a, b, k) Analysis

We analyze the total amount of (a, b, k) choices for all model repository selections. Here, we count all points of all point sets with their respective optimal parameter triple (a^*, b^*, k^*). The corresponding four global histograms for the three model repositories and the model from [Lar12] are given in Figure 5.4. There, each point of each geometry contributes one unit in the histograms, which report the number

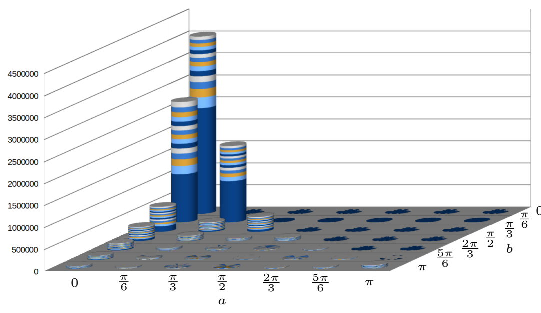


(a) Equal Weights: $a = b = \pi$, as in [WJM14]

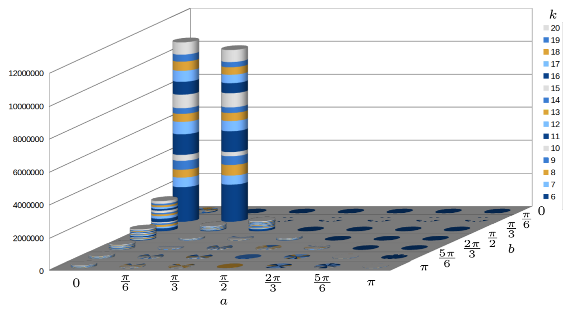
(b) Cut-Off Weights: $a = b = \arccos(0.9) \approx 0.451$ as in [YRS+18], red points show degenerate covariance matrix

(c) Optimal a_i^*, b_i^*, k_i^* for each point p_i according to Equation (5.9)

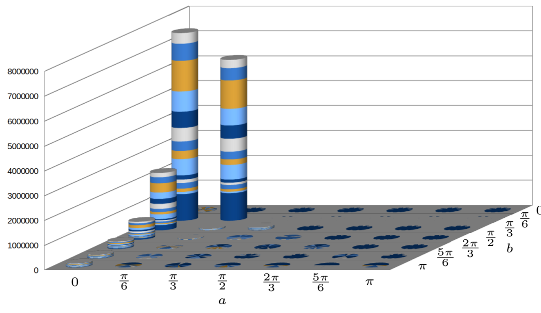
Figure 5.3: The effect of the different parameters on the Fandisk model. The top row shows the classification error E_i^{dim} from Equation (5.7) for each point of the model, from low (⬜) to high error (🟦). Note how the optimal weights from Equation (5.9) have drastically reduced error in comparison to both equal weights (used by [WJM14]) and sharp cut-off weights (used by [YRS+18]). The red points indicate elements of the point set, for which the covariance matrix from Equation (5.4) degenerates given the chosen weights.
Bottom row shows a feature classification according maximum value out of linearity (🟧), planarity (⬜), and scattering (🟦) as defined in Equation (5.6). Note how equal weights fail to consistently identify edge structures. The cut-off weights manage to identify planar areas well while over-pronouncing edge structures. These are identified well by the weights from Equation (5.9).



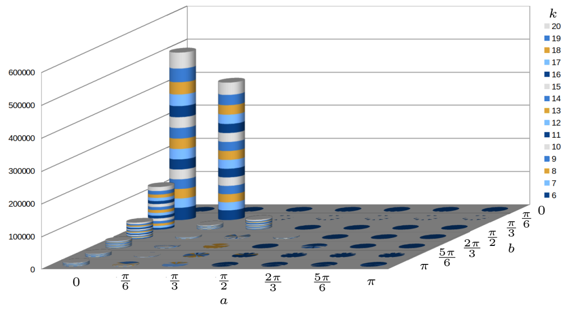
(a) Applied to 1,000 geometries randomly selected from the data set used in [HZG+18], with 7,213,429 total points.



(b) Applied to 100 geometries taken from [CZMK16], with 25,929,256 total points.



(c) Applied to the 100 geometries from the data set presented in [BRLB14], with 17,918,016 total points.



(d) Applied to the "Pan et Oursons" scan from [Lar12], with 1,199,992 total points.

Figure 5.4: Histograms of preferred sigmoid parameters (a^*, b^*, k^*) (Equation (5.9)) with respect to minimal error values for $E^{\dim}$ (Equation (5.7)) over the range $\mathfrak{K}$ when applied to several large model repositories. Each point of the respective point set(s) corresponds to one unit in the histogram. Additionally, each such bar is colored according to the chosen optimal neighborhood size k, from the lowest at the bottom to the highest at the top.

of points that choose a parameter combination (a, b), with a on the x-axis and b on the y-axis. Additionally, each such bar is colored according to the chosen optimal neighborhood size k, from the lowest at the bottom to the highest at the top. In summary, the classification error acts similar on all data sets, i.e., in the comparison between clean and real-world models.

On the large scale of 1,000 point sets with a total of 7,213,429 points (Figure 5.4(a)), we observe, that on average, a small choice for parameter a and a similarly choice for parameter b are preferred. This can be interpreted to say that it is desirable to take only normals into account that exhibit a small deviation. In particular, Figure 5.4(a) suggests that the majority of points from the clean models choose a neighborhood without any room for normal deviation ($a = b = 0$). This is one notable difference to the histograms on scanned models, Figures 5.4(b) to 5.4(d), where these drastic weights are almost never chosen.

It is particularly noteworthy that almost no points chose equal weights $a = b = \pi$ which highlights the benefit of our approach over that chosen by [WJM14]. Furthermore, choosing a sharp cut-off along the lines of [YRS+18], by $a = b = \pi/6$, occurs for about a quarter of the points. However, about 38% rather go with a softer

		$\min_i(E_i^{\text{dim}})$	$\frac{1}{N}\sum_{i=1}^{N} E_i^{\text{dim}}$	$\max_i(E_i^{\text{dim}})$	$\text{sd}_i(E_i^{\text{dim}})$
Clean [HZG+18]	Weights [WJM14]	0	0.6309891	1.071584	0.1896867
$7,213,429$ points	Ours, Eq. (5.9)	0	0.2477968	1.012825	0.1882686
Scanned [CZMK16]	Weights [WJM14]	$2.946194 \cdot 10^{-8}$	0.3348262	1.071796	0.1967012
25,929,256 points	Ours, Eq. (5.9)	0	0.2526083	0.9649821	0.1569379
Scanned [BRLB14]	Weights [WJM14]	0.001344446	0.2497551	1.048021	0.1144664
17,918,016 points	Ours, Eq. (5.9)	0	0.2109502	0.9379431	0.09631257
Scanned [Lar12]	Weights [WJM14]	0.003978099	0.3827366	0.9805029	0.1456077
1,199,992 points	Ours, Eq. (5.9)	0	0.3200202	0.7774453	0.1337365

Table 5.1: Quantitative comparison of the classification error E_i^{dim} computed over different model repositories with weights by [WJM14] and our weights from Equation (5.9). Note that our weighting scheme always obtains lower minimal, average, and maximum error as well as a lower standard deviation.

decrease by choosing $a = 0, b = \pi/6$. A localized, i.e., model-depended, discussion about the possibility to increase a and b for better results is given in the upcoming paragraph.

In terms of scanned real-world models (Figures 5.4(b) to 5.4(d)), we analyzed 100 point sets from each [CZMK16, BRLB14] and one large model from [Lar12]. In comparison to the clean models, we do observe a different behavior. Namely, while small values for a and b are still favored, the choice of $a = 0$, which was most prominent on clean models, is almost never made for scanned models. The chosen weights indicate that mostly neighborhoods with a normal deviation of up to $\pi/6$ are taken into account. These are either all weighted uniformly ($a = \pi/6$) or with gradually deteriorating weights ($a = 0$). We interpret the parameter b to reflect the noise components caused by the acquisition process. Therefore, choosing the lowest possible choice of b causes several points $p_i \in P$ to have degenerate covariance matrices C_i, independent of the chosen neighborhood size k. We will give a more detailed discussion on this in one of the following paragraphs.

In conclusion, we see that weight-determination generally favors a narrow window between parameters a and b. This corresponds to using a neighborhood with an overall small normal deviation. The value b however depends on the geometry. Clean models mostly attain smaller error values for very small values of b, whereas real-world models require slightly larger values of b to obtain non-degenerate covariance matrices. All models from all repositories have in common that they almost never report equal weights as preferred weight assignment. Hence, when regarding the classification error E_i^{dim}, the equal weighting scheme of [WJM14] is inferior to the family of weights presented here. This becomes obvious when comparing the values obtained from our experiments, see Table 5.1. The classification error computed with our weights in Equation (5.9) has lower minimum, average, maximum, and standard deviation than the error computed with the equal weights of [WJM14].

Sharp cut-off weights are only chosen as optimal weighting by a subset of the real-world scans. As [YRS+18] used sharp cut-off weights in the context of denoising, our results hint that this weight set might be beneficial in the presence of noise. However, for about 75% of the scanned models, when considering the classification error E_i^{dim} our weighting family still chooses weights superior to the cut-off weights.

	$a = 0$	$a-$	$\neg a-$	$b = a$	$b-$	$\neg b-$	# Points
Clean [HZG+18]	60.19%	39.81%	0%	36.80%	25.49%	37.72%	7,213,429
Scanned [CZMK16]	54.30%	45.70%	0%	43.50%	14.21%	42.29%	25,929,256
Scanned [BRLB14]	62.44%	37.56%	0%	36.86%	21.03%	42.11%	17,918,016
Scanned [Lar12]	60.70%	39.30%	0%	37.33%	20.46%	42.21%	1,199,992

Table 5.2: Distribution of (a^*, b^*) choices into the three cases of (a) an attained minimum $(a = 0, b = a)$, (b) a possible decrease of the parameter without failure $(a-, b-)$, and (c) impossibility of decreasing the parameter because it would cause a degenerate covariance matrix $(\neg a-, \neg b-)$.

Local (a,b) Analysis

In this paragraph, we will discuss the (a^*, b^*) choices presented in the previous paragraph from a local, i.e., point-set-dependent, perspective. The respective results are presented in Table 5.2. There, the first row corresponds to the clean models from [HZG+18] while the other three rows correspond to the scanned real-world models from [CZMK16, BRLB14, Lar12]. The columns present information about the amount of points accepting minimal value $a = 0$, allowing $(a-)$ or forbidding $(\neg a-)$ a decrease of a, accepting minimal value $b = a$, and allowing $(b-)$ or forbidding $(\neg b-)$ a decrease of b. In this scheme, the columns $\neg a-$ and $\neg b-$ denote the percentage of those points for which a decrease of the respective parameter results in a degenerate covariance matrix C_i, see Section 5.3.2. Observe that we cover all possible cases. For easy comparability, we provide the respective case numbers in percent, with the total number of points for the respective repository given in the last column.

Having all values in one chart, we directly observe the behavior assessed for parameter a in the previous paragraph. There, we stated that especially in the case of clean models, an as-small-as-possible value for a is favorable over larger values for a. Indeed, Table 5.2 confirms this statement, as almost[3] none of the points allows for an decrease of parameter a (cf. column $\neg a-$). This justifies the small values for a attained in the real-world scenarios presented in Figures 5.4(b) to 5.4(d) when compared to the values of a attained in the clean scenarios in Figure 5.4(a). Semantically, this opts for including just enough neighbors in the computation to make it feasible, i.e., to prevent a degenerate covariance matrix, but focus on those that are as-similar-as-possible with regard to the normal field.

The reported numbers on the parameter b also support the observation drawn before. It is chosen to be as-small-as-possible, i.e., as close to the chosen a without creating a degenerate covariance matrix. Over all repositories, b is chosen to create a sharp cutoff $(b = a)$ in about 36% of the considered points. A notable exception is the scanned data set [CZMK16], which allows for 43.5% of the points to choose a sharp cut-off. This is possibly due to lower noise levels and different geometry types in this data set when compared to the other scanned data sets [BRLB14, Lar12]. Furthermore, in about 42% of the scanned points and 37% of the clean points, b is at least chosen to be as-close-as-possible to a, i.e., the weighting scheme is chosen to

[3]There are $< 0.01\%$ for each scanned repository that would allow for a decrease, which is not shown here due to rounding.

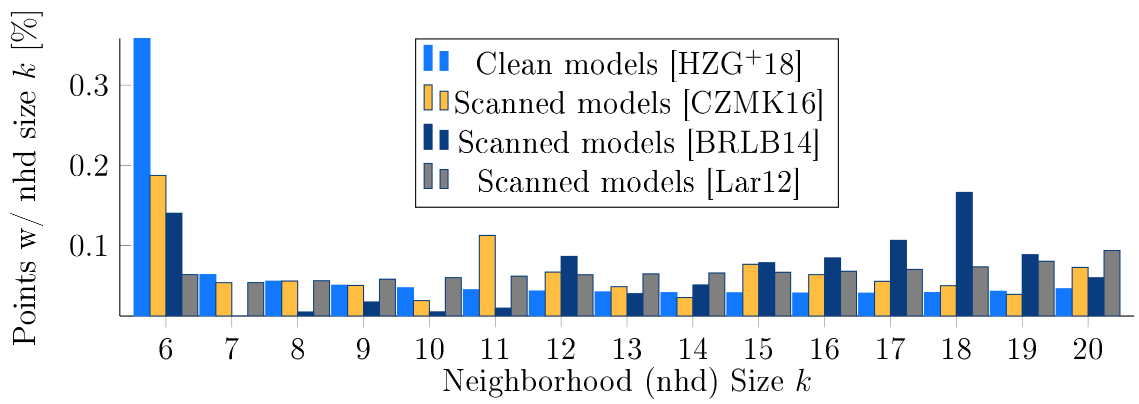


Figure 5.5: Histogram of preferred neighborhood sizes k^* with respect to minimal error value E_i^{dim}. To ensure compatibility over the different data repositories, we normalize by the total number of points and report the percentage of points choosing the respective neighborhood size.

be as-close-as-possible to a sharp cut-off, which cannot be realized because a further decrease of b would cause a degenerate covariance matrix. These observations justify the general weighting choice of sharp cutoff, as chosen by [YRS+18], although the particular chosen values only prove to be most effective in about one fourth of all models from the data set used here.

Summarizing the global and local analysis of the parameter choices (a^*, b^*), we draw the following conclusions:

- The utilized classification error favors weight determination with as-small-as-possible values for both parameters a and b. That is, only points with as-similar-as-possible normals are considered, but out of these, all are allowed to influence the computation as-evenly-as-possible.
- Equal weights ($a = b = \pi$), as used by [WJM14], are never chosen as optimal parameters to obtain a minimal classification error E_i^{dim}.
- Sharp cut-off weights as widely used in the literature, e.g., in [YRS+18], attain minimal classification error for 36.8% of the clean points and for up to 43.5% of the scanned points. This proves their relevance in particular for real-world scenarios.

Global k Analysis

As stated in the beginning of Section 5.4, for each point in the utilized point sets, we also obtain a preferred neighborhood size $k^* \in \mathfrak{K}$ yielding smallest classification error E_i^{dim} among all choices (Equation (5.9)). In Figure 5.5, we present a histogram plotting this data, i.e., for each neighborhood size $k \in \mathfrak{K}$, we show what percentage of points from the respective model repository use this k.

Note that the plot for the clean models shows a favor for an as-small-as-possible neighborhood size k over larger neighborhoods. In contrast, the scanned models show a different behavior. Whereas the repository [CZMK16] also attains its peak at $k = 6$, it is more equally distributed among the whole range, with a notable second peak at $k = 11$. The models in [BRLB14], however, exhibit an almost

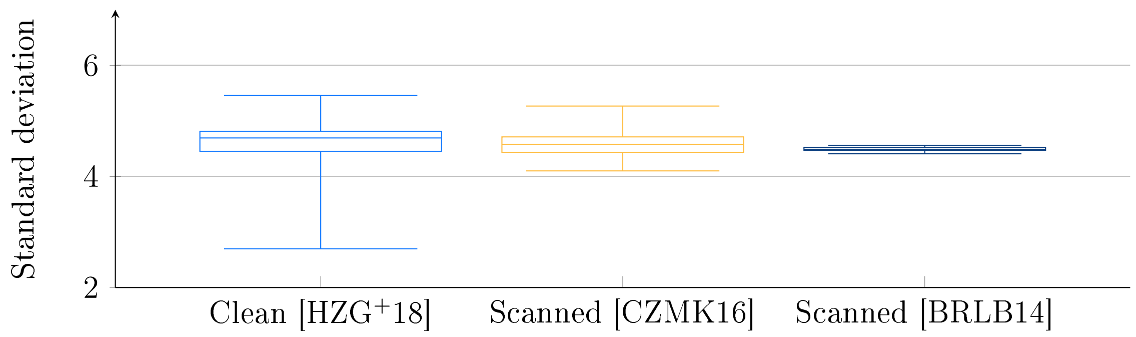


Figure 5.6: Box-whisker plot for the standard deviations obtained by the different models. Each model contributes its own standard deviation as a data point for the diagram. Therefore, the leftmost column represents 1,000 data points (from [HZG+18]), the center column represents 100 data points (from [CZMK16]), and the rightmost column represents 100 data points (from [BRLB14]).

Gaussian bump around their maximal value $k = 18$. Finally, the results for the model chosen from [Lar12] are almost uniform over the entire range, with a slight increase for growing values of k.

In order to investigate scalability effects, we have included one significantly larger model from [Lar12] in our analysis. Note that the general observations as made on Figure 5.4, Table 5.1, Table 5.2, and Figure 5.5 particularly hold for this model. Furthermore, despite the fact that [WJM14] focuses on these large-scale models, our weighting approach from Equation (5.9) still provides smaller classification errors as indicated in Table 5.1. Thus, we cannot report any scaling issues.

For the clean models, we obtain an average neighborhood size of $\bar{k} = 10.55$ with a standard deviation of $\sigma = 4.77$. For the scanned models, those quantities are:

- $\bar{k} = 12.09489$, $\sigma = 4.618431$ ([CZMK16]),
- $\bar{k} = 14.22883$, $\sigma = 4.487704$ ([BRLB14]),
- and $\bar{k} = 13.54247$, $\sigma = 4.398983$ ([Lar12]).

These findings suggest that variable neighborhood sizes yield smaller error values with regard to the classification error E_i^{dim}. If a global neighborhood size has to be chosen, then the average values provided by this analysis serve as reasonable choices, as they provide a good trade-off between a fixed neighborhood size and a low classification error. In order to further investigate the benefit of varying neighborhood sizes, in the following paragraph, we turn to a point-set-dependent perspective.

Local k Analysis

We will now consider the standard variation of the neighborhood sizes taken over a single model for E^{dim}. We aim to better understand and investigate the hypothesis formulated above, i.e., the statement that a variable neighborhood size contributes to lower classification error.

In order to interpret the neighborhood sizes, we consider a box-whisker plot over all standard deviations within the respective models in Figure 5.6. That is to say, the boxes indicate the first, second (median), and third quartile of the standard deviations of neighborhood sizes for the indicated model repository. In particular, most approaches in the literature use—and are evaluated on—a setting with a fixed neighborhood size k. In our analysis, this would correspond to a standard deviation around 0, indicating no or small changes to the neighborhood size within a geometry. However, it is obvious from Figure 5.6 that all standard deviations are located well away from 0. Even considering the minima, i.e., the lower whiskers of the boxes, they reside at 2.7, 4.1, and 4.4, respectively, indicating that at least these small variations in neighborhood size are necessary to minimize the classification error. Note that the variation of neighborhood size is most notable for the clean model repository, where the standard deviation of chosen neighborhood sizes goes up to 5.46. This is in contrast to the scanned data from [BRLB14], where the models are not very diverse, which is reflected in the almost uniform standard deviation of the chosen neighborhood sizes.

In summary, from the global and local analysis of the obtained neighborhood sizes k, we draw the following conclusions:

- All standard deviations lie well above 0, i.e., the considered classification error favors variable neighborhood sizes over constant-size neighborhoods.
- This behavior is more pronounced for scanned models (see [CZMK16] and [BRLB14]) than for clean models (see [HZG+18]).
- The classification error favors smaller neighborhood sizes for clean models, however for scanned models this behavior is not preserved.

Application Scenario

Building on the observation from the previous paragraph that varying neighborhood sizes can contribute to better performance and in order to evaluate our proposed methodology in an application scenario, we turn to the normal filtering stage of the point set denoising algorithm proposed in [YRS+18]. This first stage is part of a larger, iterative process of three stages that removes noise from an input geometry. We focus on the first stage to not have the effect of our weighting scheme be confounded by procedures within the more complex pipeline (we provide the names of the parameters of the algorithms in brackets in the following). In each iteration (parameter p), a weighted covariance matrix is built. The algorithm is using a sharp cut-off weight function (parameter ρ), optimizes the eigenvalues of the covariance matrix (parameter τ), and uses those to update the respective point normals afterwards.

For the experiment, we followed the experimental pipeline of the original article [YRS+18]. Namely, we took the four geometries (Cube, Fandisk, Octahedron, Rockerarm) as discussed in the original publication [YRS+18], together with three more geometries (Bearing, Sharp Sphere, Fertility), see Figure 5.7. The models were given as meshes and provided the oriented point normals, as they were obtained from the mesh information as weighted vertex normals that served as ground truth normals in the experiment. Afterwards, we applied Gaussian noise in normal direction with amplitudes 0.3ℓ and 0.6ℓ, where ℓ denotes the mean of all distances from points to their respective six nearest neighbors. To measure the deviation from the ground

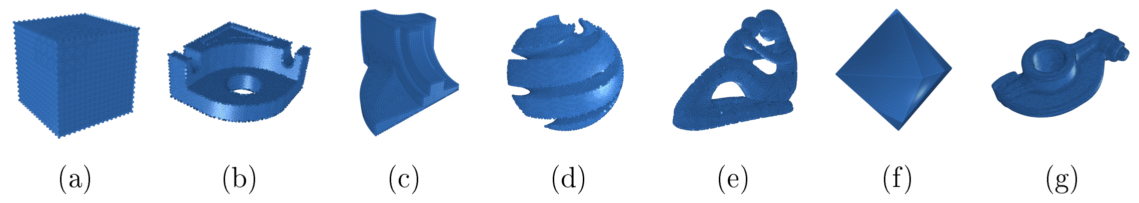


Figure 5.7: Geometries from left to right with number of vertices in brackets: Cube (1,906), Bearing (3,475), Fandisk (6,475), Sharp Sphere (8,354), Fertility (9,239), Octahedron (16,395), and Rockerarm (24,106).

truth normals, we make use of the mean squared error (MSE) given as

$$\mathrm{MSE}(N, \tilde{N}) = \frac{1}{n}\sum_{i=1}^{n} \|n_i - \tilde{n}_i\|_2^2, \tag{5.10}$$

where $N = \{n_i \mid p_i \in P\}$ are the ground truth and $\tilde{N} = \{\tilde{n}_i \mid p_i \in P\}$ are the changed normals of the geometry P.

The default values for the normal filtering were taken from [YRS+18] and the parameters (p, ρ, τ) are set as follows:

- Cube with $(150, 0.95, 0.3)$,
- Fandisk with $(150, 0.9, 0.3)$,
- Octahedron and Rockerarm with $(80, 0.9, 025)$,
- and Bearing, Sharp Sphere, and Fertility with $(100, 0.9, 0.25)$.

For the allowed neighborhood range, we first let the default normal filtering of [YRS+18] identify the neighborhoods. Then, for each point p_i, we took the number of neighbors k_i identified and let $\mathfrak{K}_i := \{k_i - 10, \ldots, k_i + 10\}$. If this caused negative values, we omitted them. Afterwards, each point chose a neighborhood according to the least error (Equation (5.7)) before and in each iteration. We set parameters $a = b = \arccos(\rho)$ w.r.t. the processed geometry, as this algorithm is tailored to benefit from a sharp distinction. In the cases where the covariance matrix degenerates as all weights become 0, we use the default neighborhood size and equal weights for such points.

This allows us to study the effect of including our weighting scheme into a larger application by comparing to the results obtained by [YRS+18]. Furthermore, we can compare both the original results of [YRS+18] and the results of our enhanced weighting pipeline to the ground truth normals provided by the noiseless models. In Table 5.3, the MSE results are shown, where MSE(Noise) compares ground truth and noisy input, MSE([YRS+18]) compares ground truth and the result of [YRS+18] with its default parameters, and MSE($\mathfrak{K}$) compares ground truth and the result of the enhanced normal filtering pipeline.

From Table 5.3, an immediate observation is that allowing individual neighborhood ranges reduces the overall error compared to the default values used in the normal filtering. This becomes obvious as all values in the column MSE($\mathfrak{K}$) are smaller than those in column MSE([YRS+18]). Thus, the algorithm benefits from

	0.3ℓ			0.6ℓ		
	MSE(Noise)	MSE([YRS+18])	MSE(ours)	MSE(Noise)	MSE([YRS+18])	MSE(ours)
Cube	0.01817	0.00472	**0.00381**	0.06778	0.04101	**0.01718**
Bearing	**0.01914**	0.09220	0.06311	**0.08796**	0.10588	0.09971
Fandisk	**0.01598**	0.04710	0.03304	0.05907	0.06702	**0.05285**
Sharp Sphere	**0.03395**	0.15219	0.12499	**0.09194**	0.16233	0.14309
Fertility	**0.13404**	0.22036	0.19741	**0.23215**	0.25364	0.25100
Octahedron	0.28785	0.28234	**0.27533**	0.30471	0.27754	**0.27162**
Rockerarm	**0.01715**	0.08137	0.06715	**0.06368**	0.08407	0.07817

Table 5.3: Mean squared errors (MSE) for several models corrupted by noise 0.3ℓ and 0.6ℓ with ℓ being the mean distance of all distances among six nearest neighbors. Our approach outperforms that of [YRS+18] in all examples considered.

individual neighborhood ranges as discussed in the previous paragraphs, instead of setting one global neighborhood size parameter.

One interesting observation can be made by incorporating the MSE(Noise) values into the comparison. For most of the models considered, these values outperform the normal filtering results. To explain this, consider that the point normals are influenced by the mesh properties, i.e., the faces and their areas affect the normals provided as ground truth. The filtering procedures thus sometimes worsens the results as it tries to clearly arrange normals corresponding to points which represent a planar area, whereas points lying on (sharp) edges in a mesh receive normals neither belonging to the areas meeting at that edge. This points towards future work and improvements of [YRS+18], but is irrelevant in the context of this work as the main message to be taken from these experiments is that utilizing optimized neighborhood selection over a wider range of k does positively influence the performance of the algorithm.

5.5 Conclusion and Further Research

We investigated a family of weights (Equation (5.2)) for point set processing. These weights are based on the normal similarity. The family includes common choices such as equal weights or sharp cut-off weights at a given threshold. Furthermore, we presented an evaluation model for neighborhood weights based on a Shannon entropy classification error (Equation (5.7)). We have performed a large-scale evaluation of our weight family on four data sets. The first set consisted of $1,000$ clean surface meshes from the work of [HZG+18]. The second and third set consisted of 100 real-world scans taken from each [CZMK16] and [BRLB14]. Additionally, we included a scanned model from [Lar12] with over one million points.

A statistical analysis revealed that the optimal weight parameters should lead to a neglect of non-similar normals, yet include mid-range normal points with a low weight. Specifically, equal weights, as used in the literature discussed in Section 5.1 and in particular in [WJM14] do not obtain minimal error values. Furthermore, sharp cut-off weights as used, e.g., by [YRS+18] do perform well on certain scanned models, but are also generally inferior to more flexible weighting terms. Finally, it became obvious in the evaluation that neighborhood sizes have to be variable over a point set as only these variable sizes attain minimal error values. The potential of this variability of neighborhood sizes was shown by incorporating point-wise neighborhood size ranges within a normal filtering stage in a point denoising algorihtm ([YRS+18]), yielding smaller mean squared errors compared to the original pipeline.

While this work addresses a variety of possible weighting choices and neighborhood sizes, to cover the most widely used versions from the literature, several aspects are left as future work. Further research consists of running the large-scale analysis on a broader range of neighborhood sizes, comparable to [WJM14].

Not only with the publication of the versatile *PointNet* architecture, machine- and deep learning techniques gradually conquered the realm of point set processing [QSMG17]. Subsequently, a wide variety of publications arose that tackle several problems related to point set processing. These touch on multiple areas discussed in this work. For instance, the *PointCleanNet* architecture was designed for denoising and outlier removal of point sets [RLBG+20]. Another representative of machine learning technology for point sets is the *NormNet* architecture, which derives point-wise normal estimations for three-dimensional point sets [HLKD19]. A problem naturally approached using normal information on a point cloud is (semantic) segmentation. In a sense, our optimized neighborhood weights do segment the point set into several, small parts with consistent normal information. In contrast, large-scale segmentation approaches via machine learning are available, e.g., utilizing edge-convolution networks [CD19]. All these machine learning approaches have in common that they depend on large sets of well labeled training data. In its generality, our approach can add to these developments in many ways. Aside from improving the (semi-)automated generation of training data, it can support the selection of neighborhood weights and sizes that serve as architecture input. Additionally, the direct incorporation of normal information on the neighborhood-level supports topological consistence, outlier resistance, and coherence of both the considered point set as well as the algorithmic procedure applied to it. Investigating these potentials of our methodology in the machine learning context is, however, left for future work.

As a closing remark, we would like to point out that several algorithms developed within the geometry processing community are heavily dependent on one or more parameters. These parametric values are often fine-tuned during an experimentation phase that is run on a small and limited data set. Our large-scale analysis in this work highlights the importance of a systematic setup for parameter evaluation. In particular, we were able to show that the usually globally chosen neighborhood size parameter yields better results when used within a bilateral weighting scheme that incorporates normal information and varying neighborhood sizes. We hope that this work encourages further research on the applicability of algorithms and their parameters “in the wild” to ensure applicability and robustness of the developed methods.

6 Flatness Model

In the previous section we investigated shape-aware neighborhoods in terms of size and incorporation of weighted point normals. In this process we made use of the three point classification types linearity, planarity, and scattering. These three entities are a kind of measure to elaborate on spatial extent of points while describing three different appearances and there are more feature classification types see [WJM14]. In the shape-aware neighborhoods we pushed the decision towards a certain feature type by utilizing point normal weights, so that the feature gets attained more clearly. Thus the classification of a feature at a point could tell us something about the local vicinity of that point inside the point set.

In the following we want to present the flatness model applied to a (subset of a) point set. The motivation is to evaluate a points' vicinity and as we are interested in points taken from surfaces, we want the model either to tend to surface appearance or not. Such an appearance would include bend or irregular shaped 2-dimensional regions and not distributions representing curves or volumes. Thus besides the utilization for point feature classification the model helps us to find a point neighborhood for a point in question with 2-dimensional extent. The contributions are therefore:

- Definition of the flatness model as a continuous model ranging between 2-dimensional and non 2-dimensional point distributions.
- Discussion of results in terms of point feature classification.
- Incorporation of the model into an iterative point set denoising algorithm.

6.1 Description of Flatness Model

Despite a possible weighting of the covariance matrix' entries done in Section 5.2 and the follow-up analysis on the entities built from the eigenvalues, we want to discuss another energy model based on the eigenvalues and its behavior. As they might reflect point distributions from dimensions 0 up to 3, we are interested to set them in relation so that 2-dimensional distributions are favored as these shall reflect portions of a surface. From the three eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ of the covariance matrix built on a set of points in $\mathbb{R}^3$ (cf. Section 5.3) we therefore want $\lambda_1 > 0$ (to guarantee the upcoming model and prevent degeneracy), λ_2 tending to λ_1 (to achieve two major distribution directions), and λ_3 going to 0 (thus diminishing volumetric extend). Observe further that we could incorporate weights into the setup of the covariance matrix, yet we do not need to as we do not rely on point normal information anymore compared to the setting in Section 5.2. The flatness energy is given by

$$E\left(\lambda_2, \lambda_3\right) = \left(1 - \frac{\lambda_2}{\lambda_1}\right)^{m_1} + 1 - \left(1 - \frac{\lambda_3}{\lambda_1}\right)^{m_2}, \quad m_1, m_2 \in 2\mathbb{N}_0. \tag{6.1}$$

The idea is to use λ_1 as a normalization factor and let λ_2 and λ_3 depend on it, as the eigenvalues are all sorted. Then the first term (blue curves in Figure 6.1) favors flat behavior, as a value of λ_2 closer to λ_1 receives smaller energy values. In contrast, the right term (orange curves) has a reversed appearance forcing λ_3 going to 0.

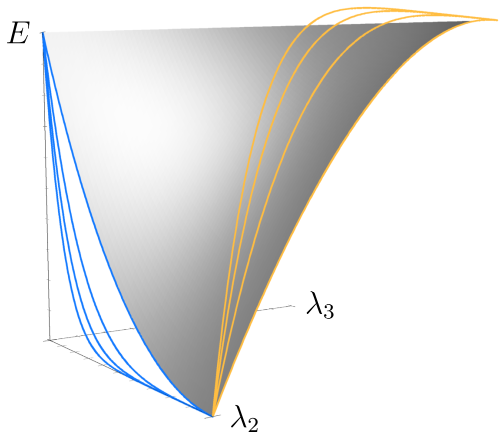


Figure 6.1: Shows function plot of energy $E(\lambda_2, \lambda_3)$, here denoted E, given in Equation (6.1) for values $m_1 = m_2 = 2$. The blue curves illustrate the first term of E with values $m_1 \in \{2, 4, 6, 8\}$ where larger values let the curves decrease faster and consequently causing smaller errors for a wider range of λ_2. In contrast the orange curves represent the second term of E with m_2 in the same range with the difference that the curves' slope increases which narrows the choices for λ_3 providing small energies in E.

The parameters m_1 and m_2 influence the focus on strengthening the surface-like or weaken the volumetric behaviors.

Observe further that each term of the form $(1-x)^m$ lies in $[0, 1]$ with $m \in 2\mathbb{N}_0$ and $x \in [0, 1]$ thus bounding the energy E in $[0, 2]$. We do not introduce normalization here, as in the experiments we will face other entities not necessarily bound between 0 and 1 and even the visual representation will range between minimal and maximal energy values for better perception.

6.2 Experimental Results

In the following discussion we want to focus on several aspects of the flatness model. At first glance those energies could provide insight into a local points' neighborhood marking the surface behavior in its vicinity. For instance a point with large energy has a neighborhood with high curvature. Thus these energies could be used to distinguish and treat points differently. In the first paragraph we focus on a parameter analysis on the two exponents arising in the energy m_1 and m_2 and their influence on energy value distribution. In the second paragraph we want to figure out flatness energy values taken from various point distributions and compare them to other entities relying on eigenvalues. Third, we want to compare the flatness energies on a model with other energy entities relying on eigenvalues and which could be used to lead to a similar point distinction on the geometry. Fourth, we discuss the influence of noise towards the generation of the flatness energy. Fifth and finally, we incorporate the energy into an application dealing with point set denoising.

As all eigenvalues are taken from a covariance matrix we chose a combinatorial neighborhood of size $k = 14$ (cf. Example 1 v))[4], with deviations mentioned, to

[4]This size shall provide a small counteract to symmetric point distributions.

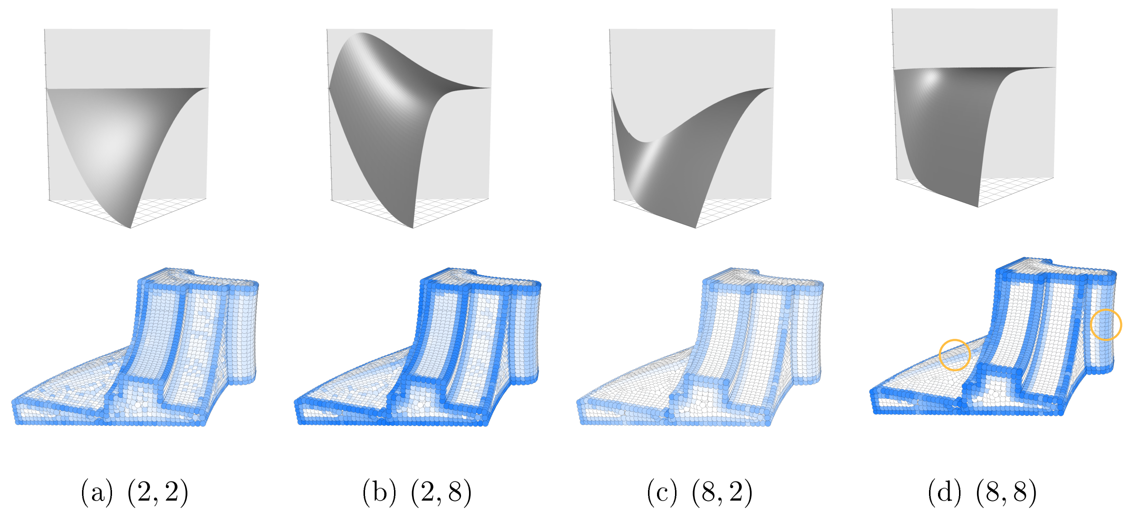


Figure 6.2: Top row shows function plots of energies given in Equation (6.1) for values (m_1, m_2) described in the respective caption. The linear subspace with normal $(-1/\sqrt{2}, 1/\sqrt{2}, 0)$ displayed in gray shall represent how the energy behaves along the line when $\lambda_2 = \lambda_3$ and the grid at the bottom represents the $\lambda_2\lambda_3$-plane. The labels of the coordinate axes are equal to those shown in Figure 6.1. The bottom row shows respective color plots on the Fandisk model ranging from low (□) to high (🟦) energy interpolating between the minimal and maximal obtained values. The orange circles in (d) set focus on the ridge (left) and curvy edge (right).

determine a point distribution around each point and assign an energy value per point and color it accordingly for visual representation.

Parameter Analysis We consider the Fandisk model on 6,475 points and the energy written in Equation (6.1) with four choices of (m_1, m_2). Plots of the energy and respective color coded representations are illustrated in Figure 6.2. Note that with a larger m_1 we put emphasis on planar behavior which means in contrast that we widen the range of point distributions which then would be counted as planar. See for instance the examples for the choices $(8, 2)$ and $(8, 8)$ where the bent regions at the front of the Fandisk receive small energy values compared to the other choices. Increasing the value m_2 instead narrows the choices of point distributions being accounted as planar, really increasing the energy value of spatial distributions even more, which can be seen that along the feature lines, i.e. areas of high mean curvature, of the Fandisk the points receive larger energies. The change of energy value distributions is shown in Figure 6.3, where we can observe how energy value distributions change w.r.t. m_1 and m_2 either get pulled to or pushed from value 0, respectively. This is due to the fact that higher values in m_1 allow us to forgive a point distribution not being totally flat, and thus working towards planarity, whereas a higher m_2 narrows spatial distributions being perceived as planar. This might give a possibility to cut at a certain threshold and extract points whose neighborhoods account for a certain error above or below that threshold.

Energies on Point Distributions In this experiment we deviate from $k = 14$ points for the combinatorial neighborhood, but use all points in the distributions shown in Figure 6.4. They shall reflect possible (yet synthetic) distributions on

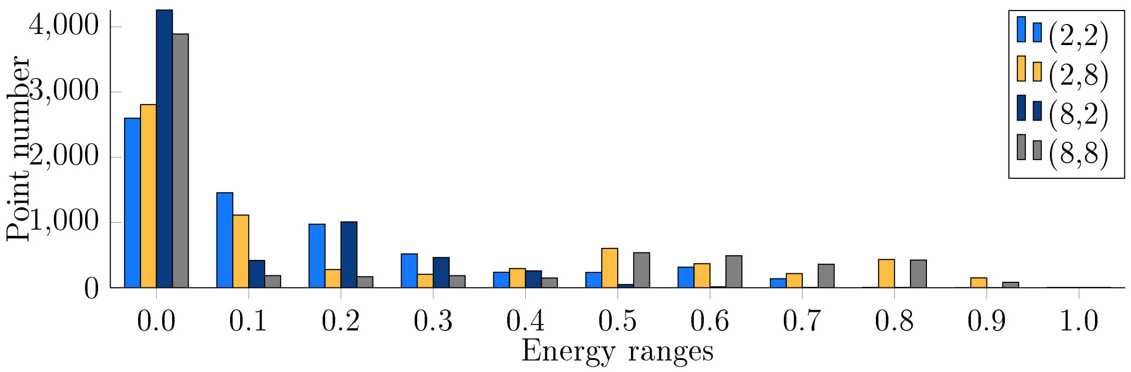


Figure 6.3: Shows distributions of energy values of examples presented in Figure 6.2 within energy ranges, for instance bars at 0.0 represent the number of points within range $[0.0, 0.1)$. The last bin refers to energy equal to 1. Note all maximal energies are lower or equal to 1 for this experiment. The energy ranges are normalized w.r.t. the maximal energy value of the respective type, i.e. 0.929 (🟦), 1.211 (🟨), 0.895 (🟦), and 0.999 (⬜).

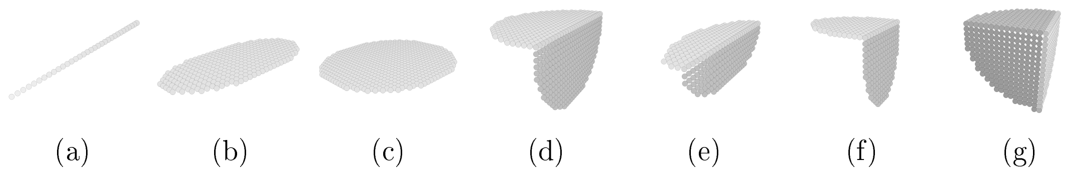


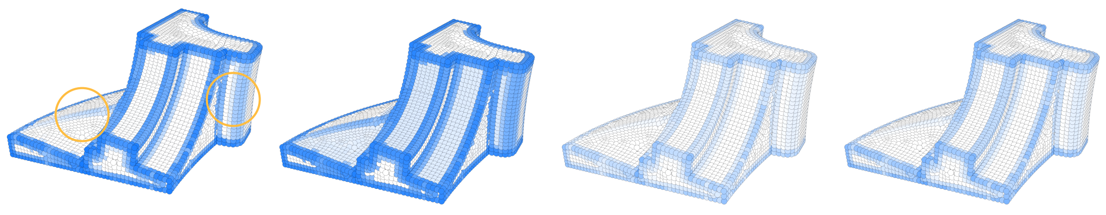
Figure 6.4: Shows point distributions in $\mathbb{R}^3$.

which we want to investigate how flatness behaves and also compare it to the three entities linearity, planarity, and scattering introduced in Equation (5.6). Our primary goal is to see how these entities could encode a feature point with its energy, i.e. whenever such distributions reflect a points' vicinity. For the flatness model we set $m_1 = m_2 = 8$ and we would assume that it produces small energy values for the cases (b) and (c), as these represent planar regions, while all other point distributions should cause larger energies. In Table 6.1 we find all energy values where we observe the desired behavior and it supports our motivation for the flatness energy mentioned in the introduction, as we can see how the entities linearity, planarity, and scattering work towards certain distributions more strictly, while the flatness energy majorly tries to distinguish two cases: being flat or not. We also did the calculation of the three competitors without point normal information, as we really wanted to measure how they behave on the given distributions and we did not focus on an energy like the one given in Equation (5.7), as this model was needed to decide about combinatorial neighborhood sizes as well as sigmoid parameters while keeping a low mixture of linearity, planarity, and scattering.

In terms of feature classification we refer back to Figure 5.3 as comparison where we either report too few points as features which should be and too many in almost planar regions are recorded as features which should not. In addition, some points might have not been classified due to covariance matrix degeneracy because of the point normal weights. The flatness model outweighs these challenges, i.e. better represents whether a point belongs to a feature or not, does not incorporate weights or normal information, and therefore causes no degenerate cases as long as $\lambda_1 > 0$.

	(a)	(b)	(c)	(d)	(e)	(f)	(g)
Flatness	1.0	0.10830	0.0	0.70701	0.58412	0.92815	0.93434
Linearity	1.0	0.75740	0.0	0.49232	0.86968	0.51985	0.02799
Planarity	0.0	0.24259	1.0	0.36667	0.09388	0.20614	0.68348
Scattering	0.0	0.0	0.0	0.14100	0.03642	0.27400	0.28852

Table 6.1: Energy values for Flatness (cf. Equation (6.1) with $m_1 = m_2 = 8$), Linearity, Planarity, and Scattering (cf. Equation (5.6)) applied to point distributions shown in Figure 6.4.



(a) (b) (c) (d)

Figure 6.5: Shows results of energies Flatness (a) (cf. Equation (6.1) with $m_1 = m_2 = 8$), Omnivariance (b), Anisotropy (c), and Change of Curvature (d) (cf. Equation (6.2)) on the Fandisk model ranging from low (□) to high (🟦) energy interpolating between the minimal and maximal obtained values. The orange circles in (a) set focus on the ridge (left) and curvy edge (right).

Feature Classification Comparison For the comparison in terms of feature classification we consider the three entities omnivariance O_λ, anisotropy A_λ, and change of curvature C_λ all mentioned in [WJM14], i.e.

$$O_\lambda = \sqrt[3]{e_1 e_2 e_3}, \quad A_\lambda = 1 - \frac{e_1 - e_3}{e_1}, \quad \text{and} \quad C_\lambda = \frac{e_3}{e_1 + e_2 + e_3}, \tag{6.2}$$

where e_i is the normalized eigenvalue $e_i = \lambda_i/(\lambda_1 + \lambda_2 + \lambda_3)$, $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ and $\lambda_1 > 0$. Note that we reversed the anisotropy (compared to the one in [WJM14]) to streamline (visual) comparisons. From a visual perspective the flatness energy with values $m_1 = m_2 = 8$ is able to capture feature lines while holding flat areas, even if they are bent, with lower energy. Also the ridge and edges without sharp cuttings are highlighted, cf. Figure 6.5(a). The other entities also capture important features yet either with much more emphasis (cf. ridge or bent flat regions in Figure 6.5(b)) or even less (cf. edges in Figures 6.5(c) and 6.5(d)). Note that the color interpolation ranges between the minimal and maximal determined values, therefore lighter blue tones are really to be perceived as lower energy values.

In Figure 6.6 we count points in the geometry lying within some energy range, where all of these ranges are normalized w.r.t. the maximal recorded error value (the minimal values are close to 0). This normalization is done, because flatness ranges up to 2 while change of curvature is smaller 1 and therefore we can extract how energies are distributed among the whole range. A direct comparison between two entities in contrast seems not advisable, but for instance from the behavior of

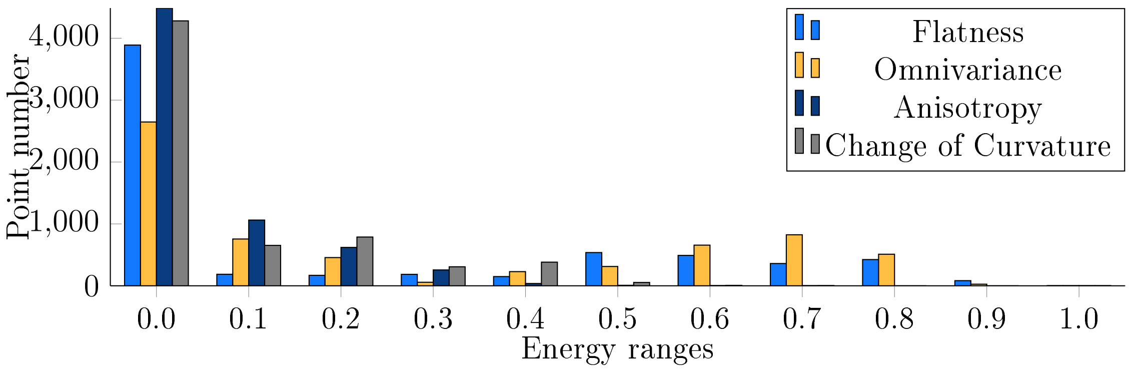


Figure 6.6: Shows distributions of energy values of examples presented in Figure 6.5 within energy ranges, for instance bars at 0.0 represent the number of points within range $[0.0, 0.1)$. The last bin refers to energy equal to 1. The energy ranges are normalized w.r.t. the maximal energy value of the respective type, i.e. 0.999 (flatness), 0.329 (omnivariance), 0.676 (anisotropy), and 0.271 (change of curvature).

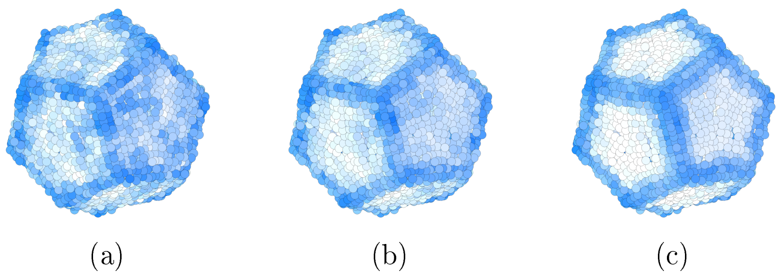


Figure 6.7: Shows results of flatness energy (cf. Equation (6.1) with $m_1 = m_2 = 8$) with three scales for the combinatorial neighborhood $k \in \{14, 28, 56\}$ (f.l.t.r.) on the noisy Dodecahedron model ranging from low (□) to high (🟦) energy.

the flatness values we can see that a large portion falls into low energies in $[0, 0.1)$ with a larger increase in the area around $[0.5, 0.8)$. This might be of interest when we want to decouple features by energy values w.r.t. a threshold and it could further lay the starting point for segmentation, i.e. here grouping connected[5] points into a segment lying in some energy range. Omnivariance shows a similar behavior, yet being normalized by a rather small maximal value. For the last two entities such a clear gap for distinction does not arise, because their bars somehow decrease without a further bump later.

Flatness on Noisy Geometries Here we want to investigate the impact of noise towards the flatness energy. We assume that small spatial variations in the distributions cause significant deviations in the error values which therefore makes the distinction into flat and non-flat parts worse. In Figure 6.7 we visually represented the energy values on a Dodecahedron model with 3,842 points equipped with Gaussian noise in normal direction with an amplitude of 25% of the average neighbor distance (taken as averaged sum over all points and their 12 nearest neighbors). For the energy we took $m_1 = m_2 = 8$ and let the combinatorial neighborhood size k range among 14, 28, and 56. Respective energy distributions can be seen in Fig-

[5]Here connection can be again thought of w.r.t. the local vicinity around a point.

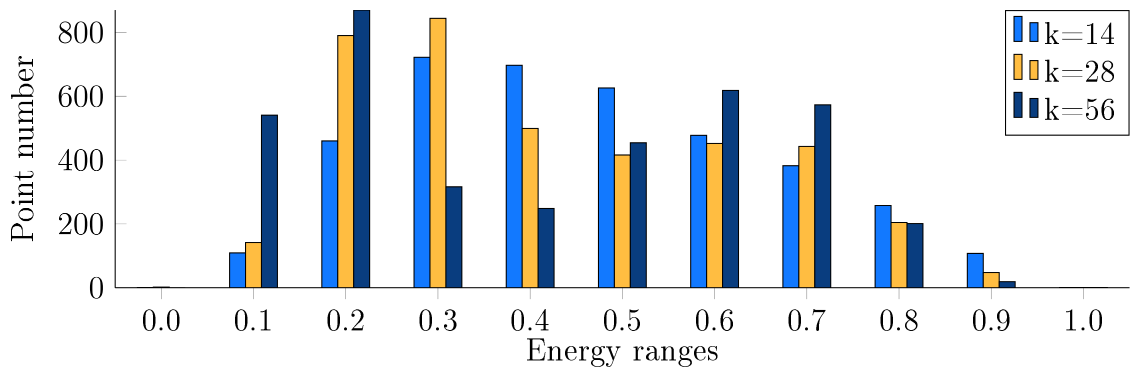


Figure 6.8: Shows distributions of energy values of examples presented in Figure 6.7 within energy ranges, for instance bars at 0.0 represent the number of points within range $[0.0, 0.1)$. The last bin refers to energy equal to 1. The energy ranges are normalized w.r.t. the maximal energy value of the respective size, i.e. 0.984 ($k = 14$), 0.919 ($k = 28$), and 0.826 ($k = 56$).

ure 6.8. Here our assumption is confirmed, i.e. lower portions of point distributions with more spatial deviations due to noise have a greater impact on energy ranges. But a way to counteract it is to increase the points to be considered, therefore a size of $k = 56$ still does not recover (almost) flat regions with low energy, but it somehow decouples flat from non-flat points more clearly (see energy distribution histogram and also the visual representation) compared to lower sizes. Here it seems to be a balance between amplitude of noise versus sampling density plays a major role into a more reliable usage of the flatness energy, i.e. larger amplitudes demand for more dense samplings so that we can increase the distribution sizes we take the energies from.

Flatness in Denoising After a brief investigation in the behavior on noisy geometries we want to think of an application scenario, namely the denoising of point set geometries. As the flatness energy could be used as a continuous model for feature classification, we want to recall what we have done in the point set denoising algorithm [YRS+18]. This iterative scheme consists of three stages with a point normal filtering, feature detection, and point update w.r.t. classified features. We already did an experiment to improve the results of the normal filtering, see the application scenario in Section 5.4. With the flatness energy being a continuous model without any necessity of point normals we do not want to improve on the feature classification in that algorithm, as normals are provided there and using those for a cut-off in terms of feature detection is natural. We aim to couple all steps into two, i.e. determine energies and update w.r.t. to these energies, and try to use less parameters with a significant impact.

Suppose we are given the points P of our noisy geometry. In an iterative scheme we determine the energy value E_p (cf. Equation (6.1)) for each $p \in P$ w.r.t. the combinatorial neighborhood (cf. Example 1 v)) and then send

$$p \mapsto p + \varepsilon \left(\eta \left(2 - E_p\right)\left(-d\bar{v}_3\right) + \left(1 - \eta\right) E_p \left(\bar{p} - p\right)\right). \tag{6.3}$$

Thus we add to the point p a sum of weighted terms scaled with $\varepsilon \in \mathbb{R}_{>0}$ to adjust the distance of movement. The left term shall send the point p towards the plane

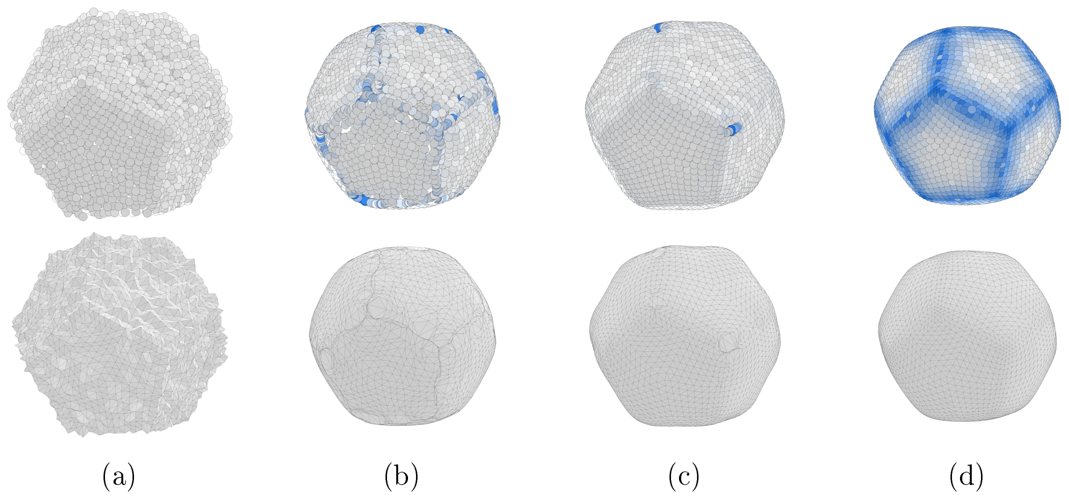


Figure 6.9: Top row shows points representing a Dodecahedron with the respective mesh representation below. The first column gives the noisy model and the other columns display results after 60 iterations with $k = 14, 8, 28$ and $\eta = 0.5, 0.5, 1$ from (b) to (d), respectively. The color interpolation is done w.r.t. the flatness energy (cf. Equation (6.1) with $m_1 = m_2 = 8$) ranging from low (⬜) to high (🟦) energy.

represented by the barycenter b of the combinatorial neighborhood and normalized eigenvector $\bar{v}_3$ to eigenvalue λ_3 (used for the flatness energy), thus d is the signed distance w.r.t. this plane. The right term sends p towards $\bar{p}$, which is the orthogonal projection of p onto the line given by b and the normalized eigenvector $\bar{v}_1$ to eigenvalue λ_1. The first term is scaled by $(2 - E_p)$, i.e. if the energy is lower we consider the point to belong to a flat region, thus we send it towards the plane, otherwise it is higher and we send it with the weight E_p onto the line. Additionally we artificially weight both terms with $\eta \in [0, 1]$.

Figure 6.9 shows examples on a noisy Dodecahedron model with 3,842 points equipped with Gaussian noise in normal direction with an amplitude of 25% of the average neighbor distance (taken as averaged sum over all points and their 12 nearest neighbors.). For the flatness energy we set $m_1 = m_2 = 8$, $\varepsilon = 0.1$, and apply 60 iterations. One such iteration determines a new combinatorial neighborhood for the values $k \in \{8, 14, 28\}$, the flatness energy for every point, and then sends every point according to Equation (6.3). After an iteration all points are updated to the new points. In Figure 6.9(b) we used a medium size of neighbors ($k = 14$), deviating from the insights gained in the previous analysis on noisy geometries. With a larger number of iterations we can see that with $\eta = 0.5$, i.e. weighting both terms equally, points in feature regions are slowly moved towards each other, i.e. tangential drift, causing holes in the point set. This effect can be narrowed selecting an even smaller size $k = 8$ in Figure 6.9(c). Here for some points of the Dodecahedron we can observe the behavior of tangential drift. In contrast, with $\eta = 1$, i.e. we ignore the second term in the update and send points only towards the plane, an even larger size $k = 28$ generates updates of better quality without moving points towards each other, see Figure 6.9(d). Thus, at the moment $\eta = 1$ provides the possibility to make use of higher neighborhood sizes k which are important when dealing with noisy geometries.

Note that while we have some parameters (iterations, k, m_1, m_2, η, and ε), currently only k and η might have a more significant impact and yet only if $\eta \neq 1$. Thus this scenario incorporating the flatness energy continuously into an denoising scheme acts as a first step and future directions could be the following: First, the second term in the update is sensitive to neighborhood sizes and causes tangential drift. Here we could change the update to apply dynamically, i.e. we start with larger neighborhood sizes k and a focus on the first term ($\eta = 1$), decreasing η and k while the number of iterative steps increase. Second, currently feature regions are not reconstructed with sharp creases if those were originally present in the ground truth model. Thus, making the update scheme sensitive to these geometric details is another open question.

6.3 Conclusion and Further Research

In this section we proposed the flatness model working as an energy and consisting of eigenvalues taken from a covariance matrix generated by a distribution of points in $\mathbb{R}^3$. This energy uses the latter two eigenvalues normalized by the first and sets its emphasis on flat or non-flat appearances. As such it applies for feature classification or a continuous model for point set denoising without the necessity of point normals or influential thresholds. Some open questions are comprised in the following list.

- The parameter analysis and respective histogram results gave some insight of the distributions of energies and a possible way to introduce thresholds to cut-off ranges and work towards point segmentation. Here we can assume that points close to each other do not provide a large deviation in energy in general and as the energy measures flatness the approach is similar to the one discussed in Section 4. Hence, we could think of incorporating the flatness energy into a segmentation algorithm without the necessity of (oriented) point normals. Such segmentation could be achieved after a single step as energies do not change. However, reassignments at region boundaries could benefit a regions' total energy, the latter is yet to be defined.

- We have seen first examples on feature classification. An open question is the quality and benefits the flatness energy model can achieve thoroughly compared to other classification entities and application scenarios.

- We incorporated the flatness energy into a point set denoising algorithm. The partition into 2- and 1-dimensional update schemes seems necessary to account for surface-like and feature line regions, respectively. The point update model introduces tangential drift whenever we put an emphasis on point updates towards the 1-dimensional affine subspaces, at least when the point distributions are larger. Thus a future direction is a proper movement of points reflecting features of the geometry.

Conclusion

In this thesis we investigated set systems on geometric point data in Euclidean space from a macroscopic (first two chapters) and microscopic (last chapter) perspective.

In Chapter A we described cloud sets, which are set systems build on points which give rise to simplicial complexes as control structures. The preferred instance of such structure is a simplicial surface (making the cloud complex a cloud surface), so that the system of clouds mimics the properties of a collection of subsets we find in an atlas describing a 2-manifold. We discussed the generation of cloud surfaces on one hand starting with a simplicial surface, generating points, and placing them in clouds. On the other hand we focused on the reverse direction, obtaining the clouds from a point set which led to topolgical questions on coverings. In practice we used the cloud surfaces' underlying simplicial surface as a skeleton to perform point motions in Euclidean space.

In Chapter B we derived a second set system which is dual to the one in the first chapter under certain conditions. For the resulting complex we conjectured them to be surface-like, i.e. having combinatorial properties so that each point has at least an abstract vicinity which is perceivable to be 2-dimensional. For their construction we presented a *k-means* clustering approach for point sets coupled with an application of generating a simplified polygonal surface out of the segmentation also reconstructing polygonal faces which are not necessarily simply connected.

In Chapter C we partitioned the microscopic investigation inside a cloud in two. First we set up a large scale evaluation to determine a neighborhood around a point accounting as an approximation of a 2-dimensional vicinity using an energy model on feature classification entities built from eigenvalues obtained by a weighted point distribution analysis. The evaluation explores the parameter spaces of the weighting function and combinatorial neighborhood sizes of the point distributions. Secondly we proposed the flatness model as a feature classification entity favoring point distributions which appear to be 2-dimensional and not curve-like or volumetric. This continuous model then is incorporated into a point set denoising application to evaluate its potency using less parameters and preserving geometric features without the necessity of point normal information.

Bibliography

[AB98] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, SCG '98, pages 39–48, New York, NY, USA, 1998. Association for Computing Machinery.

[AB99] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. *Discrete & Computational Geometry*, 22:481–504, 12 1999.

[ABCO$^+$01] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Point Set Surfaces. In *VIS'01: Proceedings of the conference on Visualization '01*, pages 21–28. IEEE Computer Society, 2001.

[ABCO$^+$03] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.

[AP10] Marco Attene and Giuseppe Patanè. Hierarchical structure recovery of point-sampled surfaces. In *Computer Graphics Forum*, volume 29 (6), pages 1905–1920. Wiley Online Library, 2010.

[BL06] David Belton and Derek D. Lichti. Classification and segmentation of terrestrial laser scanner point clouds using local variance information. *The International Archives of the Photogrammetry, Remote Sensing, and Spatial Information Sciences*, 36(5):44–49, 2006.

[BL12] Nicolas Brodu and Dimitri Lague. 3D terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology. *ISPRS Journal of Photogrammetry and Remote Sensing*, 68:121–134, 2012.

[BNPS17] Karl-Heinz Brakhage, Alice Niemeyer, Wilhelm Plesken, and Ansgar Strzelczyk. Simplicial surfaces controlled by one triangle. *Journal for Geometry and Graphics*, 21:141–152, 01 2017.

[BRLB14] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. FAUST: Dataset and evaluation for 3D mesh registration. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3794–3801, Piscataway, NJ, USA, 6 2014. IEEE.

[BSW09] Mikhail Belkin, Jian Sun, and Yusu Wang. *Constructing Laplace Operator from Point Clouds in* $\mathbb{R}^d$, pages 1031–1040. Society for Industrial and Applied Mathematics, 2009.

[BTS+17] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Gaël Guennebaud, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva. A survey of surface reconstruction from point clouds. *Comput. Graph. Forum*, 36 (1):301–329, 2017.

[CD19] Jhonatan Contreras and Joachim Denzler. Edge-convolution point net for semantic segmentation of large-scale point clouds. In *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 5236–5239, 2019.

[CSAD04] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational Shape Approximation. In *ACM Transactions on Graphics (TOG)*, volume 23 (3), pages 905–914. ACM, 2004.

[CW07] Barbara Cutler and Emily Whiting. Constrained planar remeshing for architecture. In *Proceedings of Graphics Interface 2007*, pages 11–18, 2007.

[CZMK16] Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. A large dataset of object scans. *arXiv:1602.02481*, 2016.

[DMDV11] Jerome Demantké, Clément Mallet, Nicolas David, and Bruno Vallet. Dimensionality based scale selection in 3D lidar point clouds. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-5/W12:97–102, 2011.

[DN98] Min Dai and Timothy S. Newman. Hyperbolic and parabolic quadric surface fitting algorithms–comparison between the least squares approach and the parameter optimization approach. Technical report, University of Alabama, 1998.

[FCD21] Tong Fu, Raphaëlle Chaine, and Julie Digne. Baseline skinning for point sets of articulated bodies. *arXiv:2106.03514*, 2021.

[FR01] Michael S. Floater and Martin Reimers. Meshless parametrization and surface reconstruction. *Computer Aided Geometric Design*, 18(2):77–92, 2001.

[GH97] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997.

[GMR17] Eleonora Grilli, Fabio Menna, and Fabio Remondino. A review of point clouds segmentation and classification algorithms. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2 (W3):339–344, 2017.

[GP07] Markus Gross and Hanspeter Pfister, editors. *Point-based graphics [electronic resource] / edited by Markus Gross and Hanspeter Pfister.* The Morgan Kaufmann series in computer graphics. Morgan Kaufmann, Amsterdam, 1st edition edition, 2007.

[Hat10] Allen Hatcher. *Algebraic topology / Allen Hatcher, Cornell University.* Cambridge University Press, Cambridge, first published, 13. printing edition, 2010.

[HBC11] Paul Harris, Chris Brunsdon, and Martin Charlton. Geographically weighted principal components analysis. *International Journal of Geographical Information Science*, 25(10):1717–1736, 2011.

[HDD$^+$92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface Reconstruction from Unorganized Points. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 71–78. ACM, 1992.

[HHL15] Zhen Hua, Zilong Huang, and Jinjiang Li. Mesh simplification using vertex clustering based on principal curvature. *International Journal of Multimedia and Ubiquitous Engineering*, 10(9):99–110, 2015.

[HLKD19] Janghun Hyeon, Weonsuk Lee, Joo Hyung Kim, and Nakju Doh. Normnet: Point-wise normal estimation network for three-dimensional point cloud data. *International Journal of Advanced Robotic Systems*, 16(4), 2019.

[HZG$^+$18] Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. Tetrahedral meshing in the wild. *ACM Trans. Graph.*, 37(4), 2018.

[JHPS21] Mohd Javaid, Abid Haleem, Ravi Pratap Singh, and Rajiv Suman. Industrial perspectives of 3d scanning: Features, roles and it's analytical applications. *Sensors International*, 2, 2021.

[Lar12] Oliver Laric. `http://threedscans.com/`, 2012. Online, accessed 14 September 2023.

[LB16] Kai Wah Lee and Pengbo Bo. Feature curve extraction from point clouds via developable strip intersection. *Journal of Computational Design and Engineering*, 3(2):102–111, 2016.

[LCOL06] Yaron Lipman, Daniel Cohen-Or, and David Levin. Error Bounds and Optimal Neighborhoods for MLS Approximation. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 71–80. Eurographics Association, 2006.

[Lee00] John M. Lee. *Introduction to topological manifolds / John M. Lee.* Graduate texts in mathematics 202. Springer, New York [u.a.], 2000.

[Lev98] David Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67(224):1517–1531, 1998.

[Lev04] David Levin. Mesh-independent Surface Interpolation. In Guido Brunnett, Bernd Hamann, Heinrich Müller, and Lars Linsen, editors, *Geometric modeling for scientific visualization*, pages 37–49. Springer, 2004.

[Llo82] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[LP01] Lars Linsen and Hartmut Prautzsch. Local Versus Global Triangulations. In *Proceedings of EUROGRAPHICS*, volume 1, pages 257–263, 2001.

[LW85] Marc Levoy and Turner Whitted. The use of points as a display primitive. Technical report, University of North Carolina, Department of Computer Science, Chapel Hill, NC, USA, 1985.

[MGMAI06] Matthew R. Marler, Philip Gehrman, Jennifer L. Martin, and Sonia Ancoli-Israel. The sigmoidally transformed cosine curve: a mathematical model for circadian rhythms with symmetric non-sinusoidal shapes. *Statistics in medicine*, 25(22):3893–3904, 2006.

[MM] Mathworks and Matlab. Solve a constrained nonlinear problem, problem-based. `https://de.mathworks.com/help/optim/ug/solve-nonlinear-optimization-problem-based.html`. Online, accessed 14 Septembers 2023.

[MNG04] Niloy J. Mitra, An Nguyen, and Leonidas Guibas. Estimating Surface Normals in Noisy Point Cloud Data. *International Journal of Computational Geometry & Applications*, 14(04n05):261–276, 2004.

[NL13] Anh Nguyen and Bac Le. 3d point cloud segmentation: A survey. In *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pages 225–230, 2013.

[PGK02] Mark Pauly, Markus Gross, and Leif Kobbelt. Efficient simplification of point-sampled surfaces. In *Proceedings of the conference on Visualization'02*, pages 163–170. IEEE Computer Society, 2002.

[PKKG03] Mark Pauly, Richard Keiser, Leif Kobbelt, and Markus Gross. Shape Modeling with Point-Sampled Geometry. *ACM Transactions on Graphics*, 22, 3:641–650, 2003.

[PLL12] Min Ki Park, Seung Joo Lee, and Kwan H Lee. Multi-scale tensor voting for feature extraction from unstructured point clouds. *Graphical Models*, 74(4):197–208, 2012.

[PRYZ] Konrad Polthier, Ulrich Reitebuch, Sunil K. Yadav, and Eric Zimmermann. Javaview 5.03.003. `www.javaview.de`. Online, accessed 14 September 2023.

[QSMG17] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In

Proceedings of the IEEE conference on computer vision and pattern recognition, pages 652–660, 2017.

[RC11] Radu Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6324–6328, 05 2011.

[RLBG$^+$20] Marie-Julie Rakotosaona, Vittorio La Barbera, Paul Guerrero, Niloy J Mitra, and Maks Ovsjanikov. Pointcleannet: Learning to denoise and remove outliers from dense point clouds. In *Computer Graphics Forum*, volume 39, pages 185–203. Wiley Online Library, 2020.

[RVDHV06] Tahir Rabbani, Frank Van Den Heuvel, and George Vosselmann. Segmentation of point clouds using smoothness constraint. *International archives of photogrammetry, remote sensing and spatial information sciences*, 36(5):248–253, 2006.

[RZP18] Ulrich Reitebuch, Eric Zimmermann, and Konrad Polthier. Two-layer woven surfaces with planar faces. In Eve Torrence, Bruce Torrence, Carlo Séquin, and Kristóf Fenyvesi, editors, *Proceedings of Bridges 2018: Mathematics, Art, Music, Architecture, Education, Culture*, pages 147–154, Phoenix, Arizona, 2018. Tessellations Publishing. Available online at `http://archive.bridgesmathart.org/2018/bridges2018-147.pdf`.

[SFC10] Batchimeg Sosorbaram, Tadahiro Fujimoto, and Norishige Chiba. Simplification of point set surfaces using bilateral filter and multi-sized splats. *The Journal of the Society for Art and Science*, 9 (3):140–153, 2010.

[Sha48] Claude E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[SJP18] Martin Skrodzki, Johanna Jansen, and Konrad Polthier. Directional density measure to intrinsically estimate and counteract non-uniformity in point clouds. *Computer Aided Geometric Design*, 64:73–89, 2018.

[Skr19] Martin Skrodzki. *Neighborhood Data Structures, Manifold Properties, and Processing of Point Set Surfaces.* PhD thesis, Freie Universität Berlin, Berlin, Germany, 8 2019.

[SL16] Barak Sober and David Levin. Manifold Approximation by Moving Least-Squares Projection (MMLS). *arXiv:1606.07104*, 2016.

[SRZ21] Martin Skrodzki, Ulrich Reitebuch, and Eric Zimmermann. Investigations of structures in the parameter space of three-dimensional Turing-like patterns. In *AUTOMATA2021*, Marseille, France, 7 2021.

[SWK07] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26 (2), pages 214–226. Wiley Online Library, 2007.

[SZ20] Martin Skrodzki and Eric Zimmermann. Large-scale Evaluation of Neighborhood Weights and Sizes. In *Joint SPM/SMI 2020 Conference – Poster Proceedings*, pages 14–17, 2020.

[SZ21] Martin Skrodzki and Eric Zimmermann. A large-scale evaluation of shape-aware neighborhood weights and neighborhood sizes. *Computer-Aided Design*, 141, 2021.

[SZP19] Martin Skrodzki, Eric Zimmermann, and Konrad Polthier. Variational Shape Approximation of Point Set Surfaces. In *IGS 2019 International Geometry Summit – Poster Proceedings*, pages 54–57, 2019.

[SZP20] Martin Skrodzki, Eric Zimmermann, and Konrad Polthier. Variational shape approximation of point set surfaces. *Computer Aided Geometric Design*, 80, 2020.

[Toe17] Fridtjof Toenniessen. *Topologie : ein Lesebuch von den elementaren Grundlagen bis zur Homologie und Kohomologie / Fridtjof Toenniessen.* Springer Spektrum, Berlin, Germany, 2017.

[Wes01] Douglas B. West. *Introduction to graph theory / Douglas B. West.* Prentice Hall, Upper Saddle River, NJ, 2. ed. edition, 2001.

[WJM14] Martin Weinmann, Boris Jutzi, and Clément Mallet. Semantic 3D scene interpretation: a framework combining optimal neighborhood size selection with relevant features. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3:181–188, 2014.

[WK05] Jianhua Wu and Leif Kobbelt. Structure recovery via hybrid variational surface approximation. In *Computer Graphics Forum*, volume 24 (3), pages 277–284. Wiley Online Library, 2005.

[XTZ20] Yuxing Xie, Jiaojiao Tian, and Xiao Xiang Zhu. Linking points with labels in 3d: A review of point cloud semantic segmentation. *IEEE Geoscience and Remote Sensing Magazine*, 8(4):38–59, 2020.

[Yan22] Yanan Yang. The skinning in character animation: A survey. *Academic Journal of Computing & Information Science*, 5 (4):4–17, 2022.

[YHXL15] Li Yao, Shihui Huang, Hui Xu, and Peilin Li. Quadratic error metric mesh simplification algorithm based on discrete curvature. *Mathematical Problems in Engineering*, 2015.

[YLW06] Dong-Ming Yan, Yang Liu, and Wenping Wang. Quadric surface extraction by variational shape approximation. In Myung-Soo Kim and Kenji Shimada, editors, *Geometric Modeling and Processing - GMP 2006*, pages 73–86, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[YRS$^+$18] Sunil Kumar Yadav, Ulrich Reitebuch, Martin Skrodzki, Eric Zimmermann, and Konrad Polthier. Constraint-based point set denoising using normal voting tensor and restricted quadratic error metrics. *Computers & Graphics*, 74:234–243, 2018.

[YSZP21] Sunil Kumar Yadav, Martin Skrodzki, Eric Zimmermann, and Konrad Polthier. Surface denoising based on normal filtering in a robust statistics framework. In Jin Cheng, Xu Dinghua, Osamu Saeki, and Tomoyuki Shirai, editors, *Proceedings of the Forum "Math-for-Industry" 2018*, pages 103–132, Singapore, 2021. Springer Singapore.

[ZCHK12] Henrik Zimmer, Marcel Campen, Ralf Herkrath, and Leif Kobbelt. Variational tangent plane intersection for planar polygonal meshing. In Lars Hesselgren, Shrikant Sharma, Johannes Wallner, Niccolo Baldassini, Philippe Bompas, and Jacques Raynaud, editors, *Advances in Architectural Geometry 2012*, pages 319–332. Springer, 2012.

Declaration of Authorship

Name: Zimmermann
First name: Eric

I declare to the Freie Universität Berlin that I have completed the submitted dissertation independently and without the use of sources and aids other than those indicated. The present thesis is free of plagiarism. I have marked as such all statements that are taken literally or in content from other writings. This dissertation has not been submitted in the same or similar form in any previous doctoral procedure. I agree to have my thesis examined by a plagiarism examination software.

Date: Signature:

Zusammenfassung

In dieser Arbeit werden Punktmengen, welche wie in unserem Fall zumeist von Flächen im 3-dimensionalen Raum stammen und als Repräsentation für diese Flächen dienen, mit einer Struktur versehen und zwei Perspektiven untersucht.

Die erste ist dabei eine makroskopische Sichtweise. Im ersten Kapitel werden Punkte derart durch ein Mengensystem (genannt Wolken oder Punktwolken) organisiert, dass eine simpliziale Fläche als Kontrollstruktur abgeleitet werden kann. In der Praxis betrachten wir zwei Möglichkeiten der Generierung einer solchen Kontrollstrutkur: zum einen startend von einer simplizialen Fläche, bei der wir Punkte erzeugen und in Wolken verteilen, zum anderen ausgehend von Punkten im Raum, bei der wir Teilmengen gegebener Punkte durch eine Überlagerung ausschneiden, um die simpliziale Fläche zu erhalten. Die Wolken erinnern durch ihre Beziehung zueinander an Teilmengen der Karten eines Atlas für eine Mannigfaltigkeit und mit dieser Struktur untersuchen wir im Bereich der Anwendungen, inwiefern Bewegungen der Punkte im Raum durch Positionsänderungen einer Einbettung der Kontrollstruktur im Raum hervorgerufen werden können, wobei der Einfluss auf die Punkte durch die Wolken gesteuert wird. Im zweiten Kapitel betrachten wir ein zweites Mengensystem, welches sich unter Umständen dual zum ersten verhält. Hierbei werden in der Kontrollstruktur Wolken nicht durch Punkte repräsentiert, sondern durch 2-dimensionale Entitäten berandet durch möglicherweise mehrere Kurven. Die Generierung eines solchen Mengensystems erreichen wir durch eine Segmentierung der Punkte, welche orientierte Punktnormalen respektiert. Im Bereich der Anwendung betrachten wir die Erzeugung eines polygonalen Netzes für die Kontrollstruktur, sodass wir eine simplifizierte Repräsentation und somit Approximation der Punkte erhalten.

Die zweite Perspektive untersucht eine mikroskopische Sichtweise. Im Gegensatz zur Betrachtung der Verbindung der Wolken untereinander, konzentrieren wir uns hier auf eine Wolke und die Punkte, die sie enthält. Dabei betrachten wir eine Wolke in diesem Sinne als eine Nachbarschaft der Punkte, die sie enthält und zumeist in einem lokalen Szenario, sodass für einen Punkt gezielt nach einer individuellen Nachbarschaft gesucht wird. Im ersten Teil wird eine solche Nachbarschaft bewertet durch eine Energie basierend auf geometrischen Merkmalen und zusätzlich gewichtet durch Punktnormalen. Der Parameterraum für die Gewichtung und Größen der Nachbarschaften wird dann in einem breit, d.h. basierend auf einer Vielzahl von Modellen, angelegten Experiment analysiert. Der zweite Teil widmet sich einer Beschreibung einer neuen Energie im 3-dimensionalen Raum zur Ermittlung von geometrischen Merkmalen. Diese ist für Punkte in Flächen konzipiert, wobei flächige Regionen durch niedrige Energiewerte bevorzugt und kurvige oder volumetrische Regionen durch hohe Energiewerte ausgesondert werden. Das daraus enstehende kontinuirliche Modell wird folgend integriert in eine Anwendung zum Entrauschen von Punktmengen ohne weitere Informationen wie zum Beispiel Punktnormalen.