

Natural Neighbor Interpolation  
-  
Critical Assessment and New Contributions

Dem Fachbereich Informatik  
der Technischen Universität Kaiserslautern  
zur Erlangung des akademischen Grades  
Doktor der Naturwissenschaften  
(Dr. rer. nat.)  
vorgelegte

Dissertation  
von  
Tom Bobach

Kaiserslautern, im April 2008



# Zusammenfassung

In den Ingenieur- und Naturwissenschaften treten zahlreiche Probleme auf, die eine inhärent geometrische Struktur aufweisen. Um diese rechnergestützt behandeln zu können, werden entsprechende Datenstrukturen und Zugriffsalgorithmen für die digitale Modellierung benötigt. Eine der am weitesten verbreiteten räumlichen Datenstrukturen ist die Delaunaytriangulierung, welche im kanonischen Sinne dual zum Voronoidiagramm ist. Während das Voronoidiagramm eine elegante Möglichkeit bietet, räumliche Abhängigkeiten zu modellieren, welches sich im Kernkonzept der “natürlichen Nachbarn” zusammenfassen lässt, erlaubt die Delaunaytriangulierung den robusten und effizienten Zugriff darauf. Diese nützliche Kombination führt dazu, daß Methoden basierend auf diesen Datenstrukturen in allen Bereichen der Ingenieur- und Naturwissenschaften große Bedeutung haben.

Diese Arbeit beschäftigt sich mit Teilproblemen aus einer Vielzahl von Anwendungen, deren Gemeinsamkeit ihre Verbindung zu Voronoidiagrammen und natürlichen Nachbarn ist. Zuerst wird eine Idee untersucht, welche die Verallgemeinerung von B-Splineflächen auf allgemeine, nichtreguläre Knotenstrukturen verspricht. Danach werden explizite Schritte vorgestellt, welche die Implementierung eines kürzlich eingeführten neuen Verfahrens zur  $C^2$ -stetigen Interpolation über natürlichen Nachbarn erlauben. Die glatte Interpolation mit natürlichen Nachbarn setzt das Vorhandensein von Ableitungsinformationen an den Datenpunkten voraus. Zwei neue Verfahren zur Ableitungsschätzung in unstrukturierten Daten werden vorgestellt, welche auf dem Konzept der natürlichen Nachbarn basieren. In einem verwandten Kontext beschäftigt sich die Arbeit mit der Berechnung diskret harmonischer Funktionen in Punktwolken. Eine wichtige Beobachtung, die hierbei gemacht wurde, zeigt den Zusammenhang zwischen der Approximation des Laplace-Operators mittels lokaler Koordinaten basierend auf natürlichen Nachbarn und der kontinuierlichen Abhängigkeit diskret harmonischer Funktionen von den Koordinaten der Punktwolke. Im Kontext der Datenextrapolation wird eine allgemeine Konstruktion vorgestellt, die mit Hilfe des Konzeptes der natürlichen Nachbarn eine algorithmisch transparente und glatte Erweiterung von Interpolanten über die konvexe Hülle des Datensatzes hinaus ermöglicht. Letztendlich werden in einer ausführlichen Behandlung der Eigenschaften einer vor kurzem eingeführten Netzgenerierungsmethode aus dem Bereich der Finiten Elemente Beweise für eine Reihe vorher angenommener Eigenschaften geliefert.





# Abstract

In engineering and science, a multitude of problems exhibit an inherently geometric nature. The computational assessment of such problems requires an adequate representation by means of data structures and processing algorithms. One of the most widely adopted and recognized spatial data structures is the Delaunay triangulation which has its canonical dual in the Voronoi diagram. While the Voronoi diagram provides a simple and elegant framework to model spatial proximity, the core of which is the concept of natural neighbors, the Delaunay triangulation provides robust and efficient access to it. This combination explains the immense popularity of Voronoi- and Delaunay-based methods in all areas of science and engineering.

This thesis addresses aspects from a variety of applications that share their affinity to the Voronoi diagram and the natural neighbor concept. First, an idea for the generalization of B-spline surfaces to unstructured knot sets over Voronoi diagrams is investigated. Then, a previously proposed method for  $C^2$  smooth natural neighbor interpolation is backed with concrete guidelines for its implementation. Smooth natural neighbor interpolation is also one of many applications requiring derivatives of the input data. The generation of derivative information in scattered data with the help of natural neighbors is described in detail. In a different setting, the computation of a discrete harmonic function in a point cloud is considered, and an observation is presented that relates natural neighbor coordinates to a continuous dependency between discrete harmonic functions and the coordinates of the point cloud. Attention is then turned to integrating the flexibility and meritable properties of natural neighbor interpolation into a framework that allows the algorithmically transparent and smooth extrapolation of any known natural neighbor interpolant. Finally, essential properties are proved for a recently introduced novel finite element tessellation technique in which a Delaunay triangulation is transformed into a unique polygonal tessellation.



# Acknowledgements

This thesis emerged under formidable supervision by Prof. Georg Umlauf and Prof. Gerald Farin, with further valuable guidance from Prof. Dianne Hansford. Their constant dependability, both professionally and personally, deserves my deepest gratitude and respect.

I also need to specifically thank Prof. Hans Hagen, whose dedication in creating new and interesting interdisciplinary research opportunities enabled a very intense experience during my time in the international research and training group. The open-minded and welcoming atmosphere of the whole Kaiserslautern work group enables productive and pleasant work.

Further thanks go to my collaborators Prof. Martin Hering-Bertram, Alexandru Constantiniu, and Prof. Paul Steinmann, with whom I am happy to have shared such interesting and challenging research interests.

The invaluable offer of friendship and experienced advice from Prof. Xavier Tricoche and Dr. Christoph Garth was a big help in all phases of this thesis. Their fruitful discussions and feedback constantly enriched my work.

In my research I also benefited from the cooperative and helpful correspondence with a number of people, whom I would like to thank here. Prof. Hisamoto Hiyoshi was so keen to provide me with preprints of unpublished research results of his. From Prof. Peter Alfeld I received a very insightful technical report that greatly supported my work on extrapolation. While investigating possibilities to generalize natural neighbor coordinates, Dr. Julia Flötto kindly gave me source code on higher order Voronoi diagrams. The discussion with Prof. Oleg Davidov about scattered data interpolation provided many insights, and discussion with Prof. N. Sukumar offered interesting information about practical applications of natural neighbor interpolation in finite elements.

When having needed to reflect on current issues, I greatly enjoyed the company and exchange with Dr. Ariane Middel, who always succeeds in brightening the most worrisome situation (thank you, Ariane), Dr. Burkhard Lehner, Dr. Kerstin Müller, Dr. Christoph Fünfzig, and many more I fail to mention. I thank all my colleagues in the computer graphics and geometry group in Kaiserslautern and in the PRISM lab in Phoenix, especially Mady, Inga, Pushpak, Mahesh, and Roger.

Last but not least, the support of my family and friends was essential to every part of my work. They deserve my thanks especially for putting up with me during all those times when work life got the better of me. I love my wife Sophia. Your incredible patience, your love, and your faith in me kept me going. I deeply thank my parents, who supported me all the years and paved the way to where and what I am now.



# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>                                     | <b>1</b>  |
| 1.1      | Overview . . . . .                                      | 1         |
| 1.2      | Contributions . . . . .                                 | 2         |
| <b>2</b> | <b>Foundations</b>                                      | <b>5</b>  |
| 2.1      | Linear Algebra . . . . .                                | 5         |
| 2.1.1    | Affine Geometry . . . . .                               | 6         |
| 2.1.2    | Generalized Barycentric Coordinates . . . . .           | 7         |
| 2.1.3    | Singular Value Decomposition . . . . .                  | 7         |
| 2.1.4    | Linear Least Squares . . . . .                          | 8         |
| 2.2      | Vector Analysis . . . . .                               | 8         |
| 2.2.1    | Derivatives . . . . .                                   | 9         |
| 2.2.2    | Multi-Indices . . . . .                                 | 9         |
| 2.2.3    | Multivariate Taylor Series . . . . .                    | 10        |
| 2.3      | Spatial Tessellations . . . . .                         | 10        |
| 2.3.1    | Planar Polygons . . . . .                               | 11        |
| 2.3.2    | Polyhedral Complexes . . . . .                          | 11        |
| 2.3.3    | Tessellations . . . . .                                 | 12        |
| 2.3.4    | The Delaunay Triangulation . . . . .                    | 13        |
| 2.3.5    | The Voronoi Diagram . . . . .                           | 14        |
| 2.3.6    | The Power Diagram . . . . .                             | 16        |
| 2.4      | Graphs . . . . .  | 17        |
| 2.5      | Bézier and B-Spline Functions . . . . .                 | 18        |
| 2.5.1    | Bernstein Polynomials . . . . .                         | 18        |
| 2.5.2    | Bézier Functions . . . . .                              | 19        |
| 2.5.3    | Multivariate Bernstein Polynomials . . . . .            | 20        |
| 2.5.4    | Bézier Simplices . . . . .                              | 20        |
| 2.5.5    | De Casteljau's Algorithm . . . . .                      | 21        |
| 2.5.6    | B-Spline Functions . . . . .                            | 22        |
| 2.5.7    | De Boor's Algorithm . . . . .                           | 23        |
| <b>3</b> | <b>Related Work</b>                                     | <b>25</b> |
| 3.1      | Scattered Data Interpolation . . . . .                  | 25        |
| 3.1.1    | Inverse Distance Methods . . . . .                      | 26        |
| 3.1.2    | Partition of Unity Method . . . . .                     | 27        |
| 3.1.3    | Radial Basis Functions . . . . .                        | 27        |
| 3.1.4    | Finite Element Methods . . . . .                        | 29        |
| 3.1.5    | Generalized Polygonal Barycentric Coordinates . . . . . | 29        |
| 3.2      | Natural Neighbor Interpolation . . . . .                | 31        |

|          |  |           |
|----------|--|-----------|
| 3.2.1    | Natural Neighbor Concepts . . . . .  | 31        |
| 3.2.2    | Properties of Natural Neighbor Interpolation . . . . .                           | 32        |
| 3.2.3    | Steps of Natural Neighbor Interpolation . . . . .                                | 33        |
| 3.2.4    | Smoothness of Natural Neighbor Interpolation . . . . .                           | 34        |
| 3.2.5    | Natural Neighbor Coordinates in Point Clouds . . . . .                           | 34        |
| 3.2.6    | Transfinite Natural Neighbor Coordinates . . . . .                               | 43        |
| 3.2.7    | Smooth Natural Neighbor Interpolation . . . . .                                  | 46        |
| 3.2.8    | Manifold Natural Neighbor Interpolation . . . . .                                | 52        |
| 3.2.9    | Implementation of Natural Neighbor Interpolation . . . . .                       | 53        |
| 3.2.10   | Taxonomy of Natural Neighbor Interpolants . . . . .                              | 55        |
| <b>4</b> | <b>Splines over Iterated Voronoi Diagrams</b>                                    | <b>59</b> |
| 4.1      | Background . . . . .   | 59        |
| 4.2      | Farin's Splines over Iterated Voronoi Diagrams . . . . .                         | 60        |
| 4.2.1    | Iterated Sibson's Interpolation . . . . .  | 60        |
| 4.2.2    | Representation as Local Coordinates . . . . .                                    | 61        |
| 4.3      | Equivalence with B-splines in the Univariate Case . . . . .                      | 62        |
| 4.3.1    | Evaluating Quadratic B-Splines Using de Boor . . . . .                           | 63        |
| 4.3.2    | Evaluating Quadratic B-Splines Using Repeated Knot Insertion . . . . .           | 63        |
| 4.3.3    | Evaluating Quadratic B-Splines Using Iterated Sibson's Interpolation . . . . .   | 64        |
| 4.3.4    | Difference for Degree Greater Two . . . . .                                      | 64        |
| 4.4      | Failure in the Bivariate Case . . . . .  | 65        |
| 4.5      | González' Voronoi Splines . . . . .  | 66        |
| 4.5.1    | Method Description . . . . .   | 66        |
| 4.5.2    | Discussion of González' Method and Future Research . . . . .                     | 68        |
| 4.6      | Conclusion . . . . .   | 68        |
| <b>5</b> | <b>Practical Implementation of Higher Order Natural Neighbor Coordinates</b>     | <b>71</b> |
| 5.1      | Background . . . . .   | 71        |
| 5.2      | Algebraic Volume Computation . . . . .   | 72        |
| 5.2.1    | Triangulation of the Convex Hull . . . . .                                       | 72        |
| 5.2.2    | Lasserre's Method . . . . .  | 72        |
| 5.2.3    | Voronoi (Sub-)Tiles in $\mathcal{H}$ -Representation . . . . .                   | 75        |
| 5.2.4    | Weight Dependent Power Voronoi Tile . . . . .                                    | 76        |
| 5.3      | Laplace and Sibson Natural Neighbor Coordinates with Lasserre's method . . . . . | 77        |
| 5.3.1    | Laplace Coordinates . . . . .  | 77        |
| 5.3.2    | Sibson Coordinates . . . . .   | 78        |
| 5.4      | Parametric Recursive Volume Representation . . . . .                             | 78        |
| 5.4.1    | Recursion with Variable Right Side . . . . .                                     | 78        |
| 5.4.2    | Descending the Recursion . . . . .   | 79        |
| 5.4.3    | Ascending the Recursion . . . . .  | 79        |
| 5.5      | Implementation of Hiyoshi Coordinates in 2D . . . . .                            | 81        |
| 5.5.1    | Geometric Computation . . . . .  | 81        |
| 5.5.2    | Computation with Lasserre's Method . . . . .                                     | 82        |
| 5.5.3    | Integral Evaluation . . . . .  | 84        |
| 5.6      | Conclusion . . . . .   | 85        |

|          |   |            |
|----------|---|------------|
| <b>6</b> | <b>Derivative Generation for Natural Neighbors</b>  | <b>89</b>  |
| 6.1      | Background . . . . .  | 89         |
| 6.2      | Direct Approach for Derivative Generation . . . . .                                       | 91         |
| 6.2.1    | Fitting Arbitrary Derivatives . . . . .   | 91         |
| 6.2.2    | Choosing the Neighborhood . . . . .   | 92         |
| 6.2.3    | Polynomial Precision for Natural Neighbor Interpolants . . . . .                          | 93         |
| 6.3      | Iterative Derivative Generation . . . . .   | 93         |
| 6.3.1    | A Univariate Example . . . . .  | 93         |
| 6.3.2    | Stages One and Two . . . . .  | 95         |
| 6.3.3    | An Explicit Construction for the General Setting . . . . .                                | 96         |
| 6.3.4    | Discussion . . . . .  | 98         |
| 6.4      | Results: Generated Derivatives Applied in Smooth Natural Neighbor Interpolation . . . . . | 99         |
| 6.4.1    | Visual Examples for Franke’s Function . . . . .   | 99         |
| 6.4.2    | Numerical Assessment for Franke’s Function . . . . .                                      | 108        |
| 6.5      | Complexity Issues for Quintic Bézier Simplices in Hiyoshi’s Method . . . . .              | 112        |
| 6.6      | Conclusion . . . . .  | 114        |
| <b>7</b> | <b>Discrete Harmonic Functions from Local Coordinates</b>                                 | <b>115</b> |
| 7.1      | Background . . . . .  | 115        |
| 7.1.1    | Related Work . . . . .  | 116        |
| 7.1.2    | Harmonic Functions and Their Discretization . . . . .                                     | 117        |
| 7.2      | Discrete Harmonic Functions from Local Coordinates . . . . .                              | 118        |
| 7.2.1    | Discrete Harmonic Functions over Triangulations . . . . .                                 | 118        |
| 7.2.2    | Laplacian Discretizations Based on Local Coordinates . . . . .                            | 120        |
| 7.2.3    | Experimental Comparison . . . . .   | 121        |
| 7.2.4    | Dynamic Aspects . . . . .   | 123        |
| 7.3      | Conclusion . . . . .  | 124        |
| <b>8</b> | <b>Natural Neighbor Extrapolation</b>   | <b>127</b> |
| 8.1      | Introduction . . . . .  | 127        |
| 8.2      | Related Work . . . . .  | 128        |
| 8.2.1    | Scattered Data Interpolation . . . . .  | 128        |
| 8.2.2    | Scattered Data Extrapolation . . . . .  | 131        |
| 8.3      | Taxonomy of Scattered Data Extrapolation . . . . .  | 134        |
| 8.4      | Weighted Triangular Exterior Coordinates . . . . .  | 136        |
| 8.4.1    | Method Description . . . . .  | 136        |
| 8.4.2    | Alfeld’s Extrapolation . . . . .  | 138        |
| 8.4.3    | Increasing the Continuity Away from the Convex Hull . . . . .                             | 138        |
| 8.4.4    | Restrictions and Shortcomings . . . . .   | 138        |
| 8.4.5    | Implementation Notes . . . . .  | 140        |
| 8.5      | Extrapolating Watson’s Construction . . . . .   | 141        |
| 8.6      | Projective Exterior Domain Coordinates . . . . .  | 142        |
| 8.6.1    | Extrapolation with Brown Coordinates . . . . .  | 142        |
| 8.6.2    | Extension of Circumcircles to Points at Infinity . . . . .                                | 143        |
| 8.6.3    | Extrapolation Using Brown’s Approach on Unbounded Circumcircles . . . . .                 | 145        |
| 8.7      | Ghost Points for Natural Neighbor Interpolation . . . . .                                 | 147        |

|           |  |            |
|-----------|--|------------|
| 8.7.1     | Ghost Point Idea . . . . .   | 147        |
| 8.7.2     | Assigned vs. Dismissed Ghost Points . . . . .                        | 149        |
| 8.7.3     | Static Ghost Point Placement . . . . .                               | 150        |
| 8.7.4     | Dynamic Ghost Point Placement . . . . .                              | 152        |
| 8.8       | Visual Comparison of Extrapolation Methods . . . . .                 | 157        |
| 8.8.1     | Tame Data Set . . . . .  | 157        |
| 8.8.2     | Oscillation Data Set . . . . .                                       | 159        |
| 8.8.3     | Flip Data Set . . . . .  | 161        |
| 8.8.4     | Sliver Data Set . . . . .  | 164        |
| 8.8.5     | Artifacts and Asymptotic Behavior . . . . .                          | 166        |
| 8.9       | Conclusion and Future Work . . . . .                                 | 172        |
| <b>9</b>  | <b>Adaptive Delaunay Tessellation</b>                                | <b>173</b> |
| 9.1       | Background . . . . .   | 173        |
| 9.2       | Introduction . . . . .   | 173        |
| 9.2.1     | Linear Elasticity and Nodal Integration in the Finite Element Method | 174        |
| 9.2.2     | Voronoi Tile Coverage in the ADT . . . . .                           | 175        |
| 9.3       | The Adaptive Delaunay Tessellation . . . . .                         | 175        |
| 9.4       | Geometric Properties of the ADT . . . . .                            | 177        |
| 9.4.1     | Uniqueness of the ADT . . . . .                                      | 177        |
| 9.4.2     | Connectedness of the ADT . . . . .                                   | 178        |
| 9.4.3     | An Alternative Characterization for the ADT . . . . .                | 180        |
| 9.4.4     | Coverage of Voronoi Tiles . . . . .                                  | 183        |
| 9.4.5     | Further Remarks . . . . .  | 185        |
| 9.5       | Discussion . . . . .   | 186        |
| 9.6       | Conclusion . . . . .   | 187        |
| <b>10</b> | <b>Conclusion</b>  | <b>189</b> |
|           | <b>Bibliography</b>  | <b>193</b> |
|           | <b>Curriculum Vitae</b>  | <b>203</b> |



# 1 Introduction

There's beauty in simplicity.

*~common saying*

## 1.1 Overview

A great many scientists and engineers are concerned with the analysis and solution of problems that possess an inherently geometric nature. The use of computer-aided methods in the assessment of such problems has become a standard tool in their research and work, providing insights that would otherwise be impossible. One fundamental prerequisite for such tools are data structures and algorithms that allow to model these problems on computers, and some of the most challenging types of input are data defined over unorganized, inhomogeneous point clouds, called *scattered data*.

An extremely successful geometric structure in the handling of scattered data is the classical Voronoi diagram. Indeed, it is conceptually simple and elegant, and naturally captures spatial relations. Most of its success in computational geometry is due to its underlying dual data structure, the Delaunay triangulation, for which there exist robust, proven, and widely available algorithms. Sometimes, even abstract problems without any obvious geometric interpretation are transformed into a geometric setting to permit their manipulation in this data structure.

Among the many necessary operations on scattered data is the evaluation of a smooth function that fits the given data, called *scattered data interpolation*, or that approximates it sufficiently well, in which case it is called *scattered data approximation*. This vast field has produced many powerful techniques, with solutions readily available for most problems. However, only few of the available methods can be considered conceptually simple and applicable to data distributions with an arbitrarily inhomogeneous distribution. Probably the most interesting methods standing out by these attributes are *natural neighbor interpolants*, which are defined with respect to the geometry of the Voronoi diagram.

These interpolants are elegant because they naturally generalize to higher dimensions, allow for an easy computation based on the Delaunay triangulation, and – most importantly – implicitly capture the spatial constellation of the data. This latter property is due to the Voronoi diagram which has applications far beyond scattered data interpolation.

In this general context, this thesis considers some open problems of natural neighbor concepts, with its main focus on scattered data interpolation, as well as on closely related aspects such as spatial tessellations and the application of natural neighbor coordinates to discrete function modelling.

## 1.2 Contributions

In this thesis we provide a comprehensive survey on the field of natural neighbor interpolation, which includes generalizations to data sites of arbitrary shape, to manifolds, and a presentation of available methods for the computation of natural neighbor interpolants in Chapter 3. As part of this survey, we offer insightful visual comparison between the basic natural neighbor based interpolants for data in point clouds.

In an unpublished work by Farin, an equivalence between the evaluation of B-spline functions and repeated application of Sibson's interpolant in iterated Voronoi diagrams has been shown, and conjectured to yield smooth functional surfaces in the bivariate case over unstructured knot sets. We have implemented his idea, and we were able to disprove this conjecture, which we further explain in Chapter 4.

Smooth natural neighbor interpolation depends on the computation of smooth natural neighbor coordinates in point clouds. A recently proposed generalization of Sibson's  $C^1$  interpolant by Hiyoshi builds upon  $C^k$  continuous natural neighbor coordinates whose definition lacks guidelines for a practical implementation. We provide these guidelines for arbitrary dimensions, and discuss the pros and cons of this approach when it is implemented for dimension greater than three in Chapter 5. Then, we describe a concrete implementation for two dimensions. Our guidelines also include simplified instructions for the implementation of Laplace and Sibson coordinates based on the half-space representation of Voronoi polytopes.

Another important aspect in smooth natural neighbor interpolation is the generation of derivatives that best represent the nature of the data set. Derivative generation in general is an active research area, and we propose in Chapter 6 two approaches based on the well-known fitting of Taylor polynomials to scattered values. The contribution of the first method lies in the choice of neighborhood, while the second method reduces the size of the necessary neighborhood for higher order derivatives by an iterative approach.

We rectify some issues that have been left unattended in the past. We raise alertness about the complexity implications of Hiyoshi's  $C^2$  natural neighbor interpolant in structured data sets in Section 6.5, and we correct an assumption made by Brown in his proof of the continuity of his construction in Section 3.2.5.6.

The Laplacian as a fundamental differential operator gives rise to harmonic functions on continuous domains and to discrete harmonic functions on graphs. We show that a Laplacian approximation in a scattered point set that is derived from natural neighbor coordinates leads to discrete harmonic functions that are continuous with respect to the coordinates of the point cloud in Chapter 7.

In many applications, the typical restriction of scattered data interpolation to the convex hull of the data set is an artificial limitation that often even causes unwanted artifacts. The extension of an interpolant past the convex hull of the data sites is called *extrapolation*, and to this day there is no consensus as to how to rate or classify extrapolation. In Section 8.3, we introduce new criteria by which extrapolation schemes can be assessed and compared, and specifically advocate to deviate from standard interpolation near the convex hull to circumvent the artificially induced artifacts mentioned before.

While global scattered data interpolation naturally provides extrapolation to some extent, local methods such as natural neighbor interpolation do not. All the local extrapolation approaches introduced in the past depend on particular triangulations of the data set, which makes them sensitive to perturbations affecting the triangulation. This particular problem is naturally overcome in natural neighbor interpolation, for which we develop in Section 8.7 the framework of dynamic ghost points which allows the transparent and smooth extension of any natural neighbor interpolant past the convex hull of the data set.

In the Finite Element Method, the introduction of nodal integration schemes as a step towards meshless methods has brought with it new requirements on the background tessellation to efficiently support the numerical algorithms. In the case of Voronoi based nodal integration, the Adaptive Delaunay Tessellation has been introduced as the unique tessellation that satisfies a canonical set of desirable properties, but no rigorous proofs were given so far. We deliver the missing proofs in Chapter 9 and thereby lay the theoretical foundation for further advances in this method.



## 2 Foundations

In the following we introduce basic notation and review essential results to lay the foundation for the work presented in this thesis. For the reader's convenience each chapter specifically mentions which of the following sections are relevant to the presentation.

### 2.1 Linear Algebra

We denote a member of the  $n$ -dimensional vector space over  $\mathbb{R}$  by

$$\mathbf{x} := (x_1, \dots, x_n)^T \in \mathbb{R}^n,$$

where  $x_i$  indicates the  $i$ -th component of  $\mathbf{x}$ . For two vectors  $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$ , we use  $\mathbf{x}_1^T \mathbf{x}_2 \in \mathbb{R}$  to denote the *inner product*, and  $\mathbf{x}_1 \mathbf{x}_2^T \in \mathbb{R}^{n \times n}$  to denote the *outer product*.

We use different norms, namely

|  |   |
|--|---|
| $ x $  | absolute value of $x \in \mathbb{R}$ ,            |
| $ \Omega  = \int_{\Omega} dA$  | area or volume of $\Omega \subset \mathbb{R}^n$ , |
| $\ \mathbf{x}\ _2$ or $\ \mathbf{x}\  = (\mathbf{x}^T \mathbf{x})^{\frac{1}{2}}$ | Euclidean norm of $\mathbf{x} \in \mathbb{R}^n$ , |
| $\ \mathbf{x}\ _1 =  x_1  + \dots +  x_n $                                       | Manhattan norm of $\mathbf{x} \in \mathbb{R}^n$ , |
| $\ \mathbf{x}\ _{\infty} = \max_{i=1, \dots, n}  x_i $                           | maximum norm of $\mathbf{x} \in \mathbb{R}^n$ .   |

We always assume the canonical Cartesian basis  $\mathbf{e}_1, \dots, \mathbf{e}_n$ ,  $\mathbf{e}_j = (e_i)_j = \delta_{ij}$  for a vector space  $\mathbb{R}^n$ . Then, a linear map from  $\mathbb{R}^m$  to  $\mathbb{R}^n$  has a unique representation with respect to the Cartesian system  $A\mathbf{x} = \mathbf{y}$ , where  $\mathbf{x} \in \mathbb{R}^m$ ,  $\mathbf{y} \in \mathbb{R}^n$ , and

$$A = \begin{bmatrix} a_{11} & \dots & a_{1m} \\ \vdots & \ddots & \\ a_{n1} & & a_{nm} \end{bmatrix} \in \mathbb{R}^{n \times m}.$$

Consequently,  $A\mathbf{x} \leq \mathbf{y}$  corresponds to the set of  $n$  inequalities

$$\sum_{j=1, \dots, m} a_{ij} x_j \leq y_i, \quad i = 1, \dots, n.$$

### 2.1.1 Affine Geometry

Many of the concepts that apply to polyhedral tessellations and local coordinates are closely related to affine spaces and subspaces of  $\mathbb{R}^n$ .

Corresponding to [PBP02], a finite-dimensional affine space  $\mathcal{A}$  over  $\mathbb{R}$  can be represented by a point space and an underlying vector space  $\mathbf{V}$  where elements of both  $\mathcal{A}$  and  $\mathbf{V}$  can be regarded as elements of  $\mathbb{R}^n$ . Because we will only consider such coordinate representations, we identify  $\mathcal{A}$  and  $\mathbf{V}$  with  $\mathbb{R}^n$ .

This representation of points and vectors depends on a coordinate system, for which we can choose any point  $\mathbf{p} \in \mathcal{A}$  and any set of vectors  $\mathbf{v}_1, \dots, \mathbf{v}_m$  forming a basis of  $\mathbf{V}$ . Then, any point  $\mathbf{q}$  of  $\mathcal{A}$  has a unique representation  $\mathbf{q} = \mathbf{p} + x_1\mathbf{v}_1 + \dots + x_m\mathbf{v}_m$ , i.e., the coordinate column  $\mathbf{x} = (x_1, \dots, x_m)^T \in \mathbb{R}^m$  represents  $\mathbf{q}$  with respect to the *affine system*  $\mathbf{p}; \mathbf{v}_1, \dots, \mathbf{v}_m$ . The point  $\mathbf{p}$  is referred to as the *origin* of the coordinate system and corresponds to the coordinate column  $\mathbf{0} = (0, \dots, 0)^T \in \mathbb{R}^m$ .

The *dimension* of an affine space  $\mathcal{A}$  is defined as the dimension of its underlying vector space  $\mathbf{V}$ . An *affine subspace* of  $\mathcal{A}$  is an affine space  $\mathcal{A}' \subset \mathcal{A}$  whose underlying vector space  $\mathbf{V}'$  is a subspace of  $\mathbf{V}$ .

Any set of points  $\{\mathbf{p}_0, \dots, \mathbf{p}_m\}$  of an affine space  $\mathcal{A}$  is called *affinely independent* if the vectors  $\mathbf{p}_1 - \mathbf{p}_0, \dots, \mathbf{p}_m - \mathbf{p}_0$  are linearly independent.

Let  $m$  be the dimension of  $\mathcal{A}$ . Then, any independent sequence  $\mathbf{p}_0, \dots, \mathbf{p}_m$  forms a *frame* of  $\mathcal{A}$ , and every point  $\mathbf{q} \in \mathcal{A}$  can be uniquely written as

$$\begin{aligned} \mathbf{q} &= \mathbf{p}_0 + x_1(\mathbf{p}_1 - \mathbf{p}_0) + \dots + x_m(\mathbf{p}_m - \mathbf{p}_0) \\ &= x_0\mathbf{p}_0 + x_1\mathbf{p}_1 + \dots + x_m\mathbf{p}_m, \end{aligned}$$

where  $x_0 + \dots + x_m = 1$ . The coefficients  $x_i$  are called the *barycentric coordinates* of  $\mathbf{q}$  with respect to the frame  $\mathbf{p}_0, \dots, \mathbf{p}_m$ .

Let  $\mathbf{a}_1, \dots, \mathbf{a}_l$  be the affine or barycentric coordinate columns of  $l$  points in  $\mathcal{A}$ . Then, the weighted sum

$$\mathbf{a} = \sum_{i=1, \dots, m} \alpha_i \mathbf{a}_i \quad \text{represents a} \quad \begin{cases} \text{point} \\ \text{vector} \end{cases} \quad \text{if} \quad \sum_{i=1, \dots, m} \alpha_i = \begin{cases} 1 \\ 0 \end{cases} .$$

If the weights  $\alpha_i$  sum to one, then  $\mathbf{a} = \alpha_1\mathbf{a}_1 + \dots + \alpha_m\mathbf{a}_m$  is called an *affine combination*. If, in addition, the weights are non-negative, then  $\mathbf{a}$  is called a *convex combination*, and lies in the *convex hull* of  $\mathbf{a}_1, \dots, \mathbf{a}_m$ , which is defined by

$$\mathcal{C}(\mathbf{a}_1, \dots, \mathbf{a}_m) = \{ \mathbf{a} : \mathbf{a} \text{ is a convex combination of } \mathbf{a}_1, \dots, \mathbf{a}_m \} .$$

If the dimension of  $\mathcal{A}$  is  $m$ , then the convex hull of any  $m + 1$  affinely independent points  $\mathbf{a}_0, \dots, \mathbf{a}_m \in \mathcal{A}$  is called a *simplex* in  $\mathcal{A}$ .

If a set of points  $\mathbf{p}_1, \dots, \mathbf{p}_m \in \mathbb{R}^n$  contains  $n + 1$  affinely independent points, we say they

are in *general position*.

If  $\mathcal{A}$  has dimension  $m$ , then the isometric mapping of any compact subset  $\Omega \subset \mathcal{A}$  to  $\mathbb{R}^m$  has a volume which we denote by

$$\text{Vol}(m, \Omega).$$

### 2.1.2 Generalized Barycentric Coordinates

Let  $\mathbf{x}_1, \dots, \mathbf{x}_m \in \mathbb{R}^n$  be a sequence of points in general position. Then, any set of coefficients  $\lambda_1, \dots, \lambda_m \in \mathbb{R}$  in an affine combination of the form

$$\mathbf{x} = \sum_{i=1, \dots, m} \lambda_i \mathbf{x}_i \quad (2.1)$$

is called *generalized barycentric coordinates* of  $\mathbf{x} \in \mathbb{R}^n$  with respect to  $\mathbf{x}_1, \dots, \mathbf{x}_m$ , and will be abbreviated by  $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_m) \in \mathbb{R}^m$ .

Property (2.1) is called *local coordinate property, LCP*, and  $\boldsymbol{\lambda}$  is said to be convex if  $0 \leq \lambda_i, i = 1, \dots, m$ .

The equivalence relation  $\boldsymbol{\lambda}_a \sim \boldsymbol{\lambda}_b \iff \exists \alpha > 0 : \boldsymbol{\lambda}_a = \alpha \boldsymbol{\lambda}_b$  gives rise to *homogeneous generalized barycentric coordinates*, which satisfy

$$\mathbf{0} = \sum_{i=1, \dots, m} \hat{\lambda}_i (\mathbf{x}_i - \mathbf{x}),$$

and we use the hat symbol “ $\hat{\phantom{x}}$ ” to indicate that  $\hat{\boldsymbol{\lambda}} \sim \boldsymbol{\lambda}$ .

**Remark 2.1** *By definition, only points inside the convex hull of  $\mathbf{x}_1, \dots, \mathbf{x}_m$  can be represented by a convex combination of  $\mathbf{x}_1, \dots, \mathbf{x}_m$ .*

### 2.1.3 Singular Value Decomposition

For each matrix  $A \in \mathbb{R}^{m \times n}$  with rank  $r$ , its *singular value decomposition* (SVD) is the decomposition into a set of three matrices

$$A = U \Sigma V^T,$$

with  $U \in \mathbb{R}^{m \times m}$  and  $V \in \mathbb{R}^{n \times n}$  orthonormal, and  $\Sigma \in \mathbb{R}^{m \times n}$  (see Figure 2.1). The only non-zero entries of  $\Sigma$  are its first  $r$  diagonal elements  $\sigma_1, \dots, \sigma_r$ , called *singular values* of  $A$ . Each  $\sigma_i$  is the squareroot of an *eigenvalue* of  $A^T A$ .

The *range* of a linear map  $A \in \mathbb{R}^{m \times n}$  is the linear subspace  $\text{ran } A = \{ A\mathbf{x} : \mathbf{x} \in \mathbb{R}^n \}$ , and is spanned by the first  $r$  column vectors of  $U$ . The *kernel* of  $A$  refers to the linear subspace  $\text{ker } A = \{ \mathbf{x} \in \mathbb{R}^n : A\mathbf{x} = \mathbf{0} \}$ , and is spanned by the last  $n - r$  column vectors of  $V$ .

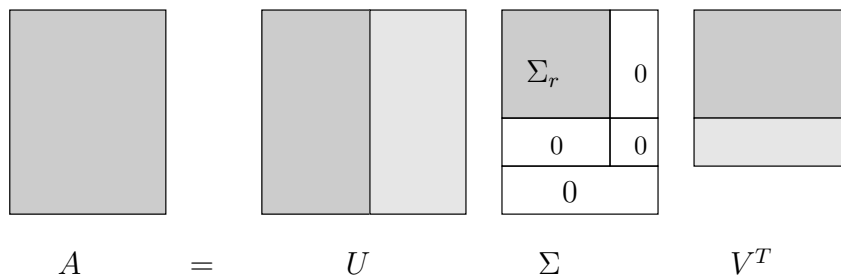


Figure 2.1: The SVD of a matrix  $A$ . If  $r$  is the rank of  $A$ ,  $\Sigma_r \in \mathbb{R}^{r \times r}$ . The dark grey areas are uniquely determined, while the column vectors of  $U$  and the row vectors of  $V^T$  corresponding to the light grey areas are arbitrary orthogonal complements.

### 2.1.4 Linear Least Squares

The general *least squares* problem for vector-valued functions  $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$  arises when the equation  $f(\mathbf{x}) = \mathbf{b}$  is over-constrained. A minimizer  $\mathbf{x}^*$  of the corresponding residual  $\|f(\mathbf{x}) - \mathbf{b}\|^2$  is called *least squares solution* of  $f(\mathbf{x}) = \mathbf{b}$ .

If  $f$  is a linear function given by  $A \in \mathbb{R}^{m \times n}$ , the minimization of

$$\|A\mathbf{x} - \mathbf{b}\|^2$$

is called a *linear least squares (LLS) problem*.

It is a special property of the SVD that it can be used to obtain the minimizer of the squared residual, as can be verified in [GL89]. If  $\mathbf{b} \notin \text{ran } A$ , the unique minimizer  $\mathbf{x}^*$  can be computed using the SVD,  $A = U\Sigma V^T$ ,

$$\mathbf{x}^* = V\Sigma^{-1}U^T\mathbf{b},$$

where  $\Sigma^{-1} \in \mathbb{R}^{n \times n}$  has diagonal entries  $\sigma_1^{-1}, \dots, \sigma_r^{-1}$  corresponding to the singular values of  $A$ .

In case  $\mathbf{b} \in \text{ran } A$ , there is an affine solution space  $\mathcal{A}$  of dimension lower or equal to  $m$ , and the inversion of the SVD provides a unique solution  $\mathbf{x}^* \in \mathcal{A}$  with minimal norm.

If  $A$  has full column rank, the minimizer  $\mathbf{x}^*$  is without loss of generality given by

$$\mathbf{x}^* = A^+\mathbf{b},$$

where  $A^+ := (A^T A)^{-1}A^T$  is the *Moore-Penrose inverse* of  $A$ . If the matrix  $A^T A$  is known to be well-conditioned, the application of  $A^+$  is preferable due to the computational cost involved in computing the SVD.

## 2.2 Vector Analysis

This thesis is mainly concerned with real-valued functions defined over the Cartesian vector space  $\mathbb{R}^n$ , and their smoothness properties.



### 2.2.1 Derivatives

We denote the  $n$ -th derivative with respect to  $x \in \mathbb{R}$  using Leibniz notation

$$\frac{d^n}{dx^n},$$

and the *partial derivative* of a multivariate function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  with respect to the  $i$ -th component  $x_i$  of  $\mathbf{x} \in \mathbb{R}^n$  as

$$\frac{\partial}{\partial x_i}.$$

The *directional derivative* of a function with respect to a direction  $\mathbf{v}$  at a point  $\mathbf{p}$  is given by the univariate derivative of the composition with a linear function  $\mathbf{d}(t) = \mathbf{p} + t\mathbf{v}$ , and will be denoted by

$$\frac{d}{d\mathbf{v}}f|_{\mathbf{p}} := \frac{d}{dt}f(\mathbf{d}(t))|_{t=0}.$$

If  $\Pi_n^d$  denotes the space of  $n$ -variate polynomials of total degree  $d$ , and  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is a function interpolating  $\mathbf{x}_1, \dots, \mathbf{x}_l \in \mathbb{R}^n$  and data  $z_1, \dots, z_l \in \mathbb{R}$ , then we say  $f$  has *polynomial precision* of degree  $d$  if for  $p \in \Pi_n^d$  with  $z_i = p(\mathbf{x}_i)$  for  $i = 1, \dots, l$ ,

$$f \equiv p.$$

### 2.2.2 Multi-Indices

A very useful tool in dealing with multivariate functions is the *multi-index*, which we denote by bold, lowercase letters. An  $n$ -dimensional multi-index of degree  $d \in \mathbb{N}_0$  is an integer-valued vector  $\mathbf{i} = (i_1, \dots, i_n) \in \mathbb{N}_0^n$ , with  $|\mathbf{i}| = i_1 + \dots + i_n = d$ . Most operations on integers can be transferred to  $\mathbf{i}$  by applying them to each component  $i_j$  of  $\mathbf{i}$ :

|  |   |
|--|---|
| $\alpha^{\mathbf{i}} = \alpha_1^{i_1} \cdot \dots \cdot \alpha_n^{i_n}$                                    | $\mathbf{i}$ -th power of $\alpha \in \mathbb{R}^n$ , |
| $\mathbf{i}! = i_1! \cdot \dots \cdot i_n!$  | factorial,  |
| $\binom{j}{\mathbf{i}} = \frac{j!}{\mathbf{i}!}$   | the binomial coefficient<br>with an integer $j$ ,     |
| $\mathbf{e}_j = (\underbrace{0, \dots, 0}_j, 1, \underbrace{0, \dots, 0}_{n-j})$                           | the $j$ -th canonical unit index                      |
| $\mathbf{d}_j = d \cdot \mathbf{e}_j$  | the $j$ -th ..., where $d$ is determined by context   |
| $\frac{d^{ \mathbf{i} }}{d\mathbf{x}^{\mathbf{i}}} = \frac{d^{ \mathbf{i} }}{dx_1^{i_1} \dots dx_n^{i_n}}$ | the $\mathbf{i}$ -th mixed partial derivative,        |
| $\mathbf{i} \leq \mathbf{j} \iff i_k \leq j_k, k = 1, \dots, n$  | partial ordering                                      |
| $\mathbf{i} - \mathbf{j} = (\max\{i_1 - j_1, 0\}, \dots, \max\{i_n - j_n, 0\})$                            | capped difference                                     |
| $\Delta_{i_j}\mathbf{i} = \mathbf{i} - \mathbf{e}_i + \mathbf{e}_j$  | shift along direction $\mathbf{e}_j - \mathbf{e}_i$   |

Using multi-index notation, the *multinomial expansion* of  $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)$  becomes

$$(\alpha_1 + \dots + \alpha_n)^d = \sum_{|\mathbf{i}|=d} \binom{d}{\mathbf{i}} \boldsymbol{\alpha}^{\mathbf{i}}.$$

Multi-index notation allows to hide considerable combinatorial complexity behind an innocent face, making the following result on the size of multi-index sets interesting. After [Mik77, Cam94], the number of  $m$ -dimensional multi-indices  $\mathbf{j}$  with  $|\mathbf{j}| = d$  is

$$|\{\mathbf{j} \in \mathbb{N}^m : |\mathbf{j}| = d\}| = \binom{d+m-1}{d}. \quad (2.2)$$

### 2.2.3 Multivariate Taylor Series

*Taylor's theorem* states that every function  $f$  that is  $n$ -times differentiable at a point  $\mathbf{a}$  can be written as

$$f(\mathbf{a} + \mathbf{x}) = \sum_{|\mathbf{i}| \leq n} \frac{1}{\mathbf{i}!} \frac{d^{\mathbf{i}}}{d\mathbf{x}^{\mathbf{i}}} f|_{\mathbf{a}} \mathbf{x}^{\mathbf{i}} + R(\mathbf{x}),$$

where  $R(\mathbf{x})$  is the remainder term which, for a large group of functions, converges to zero as  $\mathbf{x} \rightarrow \mathbf{0}$ .

For an  $n$ -variate, real-valued function  $f$  that is infinitely differentiable in the neighborhood of a point  $\mathbf{a} \in \mathbb{R}^n$ , its *Taylor series* is a representation as an infinite sum of polynomials whose coefficients are computed from the derivatives of that function at  $\mathbf{a}$ , denoted by

$$f(\mathbf{a} + \mathbf{x}) = \sum_{|\mathbf{i}| \geq 0} \frac{1}{\mathbf{i}!} \frac{d^{\mathbf{i}}}{d\mathbf{x}^{\mathbf{i}}} f|_{\mathbf{a}} \mathbf{x}^{\mathbf{i}}.$$

The function  $f$  is called *analytic* on a neighborhood  $\Omega \ni \mathbf{p}$  if the series converges for every  $\mathbf{x} \in \Omega$ , i.e.,  $f$  is identical to its Taylor series on  $\Omega$ . In this representation, the shape of the function has a good local approximation by the polynomials in the first terms of the Taylor series.

A common notation for the first two terms of the Taylor series expansion uses the gradient  $\nabla f(\mathbf{a})$  and the Hessian  $\mathcal{H}f(\mathbf{a})$  to encode derivatives up to order two in a concise way as

$$f(\mathbf{a} + \mathbf{x}) = f(\mathbf{a}) + (\nabla f(\mathbf{a}))^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \mathcal{H}f(\mathbf{a}) \mathbf{x} + R(\mathbf{x}).$$

## 2.3 Spatial Tessellations

The following definitions have been loosely adopted from [Zie94], and further treatment of spatial tessellations in general can be found in [OBSC00].

Because of their importance in two dimensions, we first introduce planar polygons based

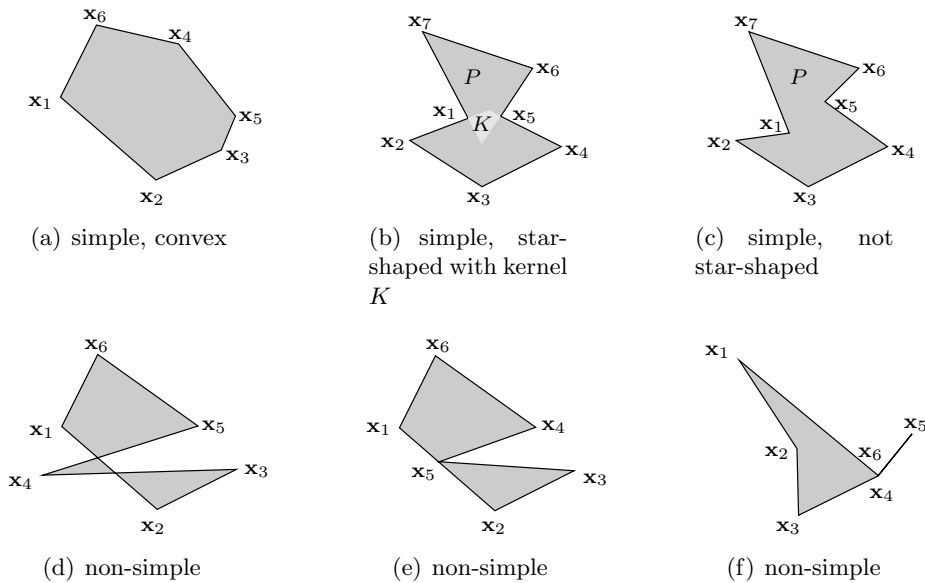


Figure 2.2: (a)-(c) Simple polygons (d)-(f) Non-simple polygons. (b) The kernel of a polygon is the set of internal points which can be connected to all vertices by straight lines without intersecting the polygon.

on their vertex representation, and move to their generalization to polyhedra in  $\mathbb{R}^n$  afterward. The Delaunay triangulation and the Voronoi diagram are then introduced as concrete tessellations.

### 2.3.1 Planar Polygons

Any sequence of points  $\mathbf{p}_1, \dots, \mathbf{p}_m$  in the plane, called *vertices*, defines a *polygon* consisting of a sequence of line segments  $\overline{(\mathbf{p}_1\mathbf{p}_2)}, \dots, \overline{(\mathbf{p}_{m-1}\mathbf{p}_m)}$ , called *edges* of the polygon. If  $\mathbf{p}_m = \mathbf{p}_1$ , the polygon is called *closed*, and its *interior* is defined as the set of points for which there is a ray starting from them that intersects an odd number of edges. A polygon is called *simple* if the intersection of any two edges is either their common vertex or empty. A stronger notion of interior of a simple polygon is given by its *kernel*, which is the set of points visible from all polygon vertices at the same time, i.e., the set of points for which all straight lines connecting them to polygon vertices are completely inside the polygon. If a polygon has a non-empty kernel, it is called *star shaped*. Some examples of simple polygons are given in Figure 2.2(a)-(c), of non-simple polygons in Figure 2.2(d)-(f).

### 2.3.2 Polyhedral Complexes

In higher dimensions, the proper equivalent of a polygon in the plane is a *polyhedral complex*. We will deal with convex polyhedra as the building blocks of spatial tessellations, and with non-convex polygons in the plane.

There is some confusion about the exact meaning of the term polyhedron, as Grünbaum points out in [Grü03]. Often it is understood as a piecewise linear surface embedded in

three-space, sometimes referring to the surface bounding a solid and sometimes to the solid itself. We will restrict the definitions to convex polyhedra representing volumes in affine spaces over  $\mathbb{R}^n$ , and restate some definitions and results as found in textbooks like [ADKS05].

The proper generalization of a convex, closed polygon to  $\mathbb{R}^n$  is a *convex polytope*  $Pt$ , which has two equivalent representations. In the  $\mathcal{V}$ -representation,  $Pt$  is the convex hull

$$Pt = \mathcal{C}(\mathbf{p}_1, \dots, \mathbf{p}_k),$$

of a minimal, finite set of points  $\mathbf{p}_1, \dots, \mathbf{p}_k$ , called *vertices* of the polytope. In the  $\mathcal{H}$ -representation,  $Pt = \bigcap_i \mathcal{H}_i^{\leq}$  is the intersection of a minimal, finite set of half-spaces  $\mathcal{H}_1^{\leq}, \dots, \mathcal{H}_l^{\leq}$ , where  $\mathcal{H}_i^{\leq} = \{ \mathbf{x} \in \mathbb{R}^n : \mathbf{a}_i \mathbf{x} \leq b_i \}$ , and we write

$$Pt = Pt(A, \mathbf{b}) = \{ \mathbf{x} \in \mathbb{R}^n : A\mathbf{x} \leq \mathbf{b} \},$$

where  $A = [\mathbf{a}_1 \dots \mathbf{a}_l]^T \in \mathbb{R}^{l \times n}$  and  $\mathbf{b} = (b_1, \dots, b_l)^T$ .

It is clear from the  $\mathcal{V}$ -representation that a convex polytope is always bounded. The unbounded generalization is called *convex polyhedron*  $Ph$ , which has two equivalent representations as well. For two sets  $A$  and  $B$ , the operation  $A \oplus B = \{ a + b : a \in A, b \in B \}$  represents the *Minkowski sum*. If  $\text{cone}(\mathbf{v}_1, \dots, \mathbf{v}_m) = \{ \sum_i \alpha_i \mathbf{v}_i : \alpha_i \in \mathbb{R}, \alpha_i \geq 0 \}$  is the cone spanned by a set of vectors, and  $\oplus$  is the *Minkowski sum*, then the  $\mathcal{V}$ -representation of a convex polyhedron is

$$Ph = \mathcal{C}(\mathbf{p}_1, \dots, \mathbf{p}_k) \oplus \text{cone}(\mathbf{v}_1, \dots, \mathbf{v}_m),$$

where the sets  $\{\mathbf{p}_1, \dots, \mathbf{p}_k\}$  and  $\{\mathbf{v}_1, \dots, \mathbf{v}_m\}$  are minimal. The  $\mathcal{H}$ -representation of  $Ph$  is equivalent to that of a convex polytope.

A polyhedral complex  $\mathcal{C}$  is defined in [Zie94] as a finite collection of polyhedra in  $\mathbb{R}^n$  such that

1. the empty polyhedron  $P = \emptyset$  is in  $\mathcal{C}$ ,
2. if  $P \in \mathcal{C}$ , then all the faces of  $P$  are also in  $\mathcal{C}$ ,
3. the intersection  $P \cap Q$  of two polyhedra  $P, Q \in \mathcal{C}$  is a face both of  $P$  and of  $Q$ , or empty.

Consequently, a *simplicial complex* is a polyhedral complex containing only simplices.

The generalization of the above to *non-convex polyhedra* will not be given here, but it should be noted that they can be defined as the result of a finite set of set operations and topological operations on convex polyhedra.

### 2.3.3 Tessellations

Most of this thesis will deal with *tessellations* or *tilings* in one way or another. We briefly introduce tessellations, and then turn to two fundamental, mutually dual tessellations: the Delaunay triangulation and the Voronoi diagram.

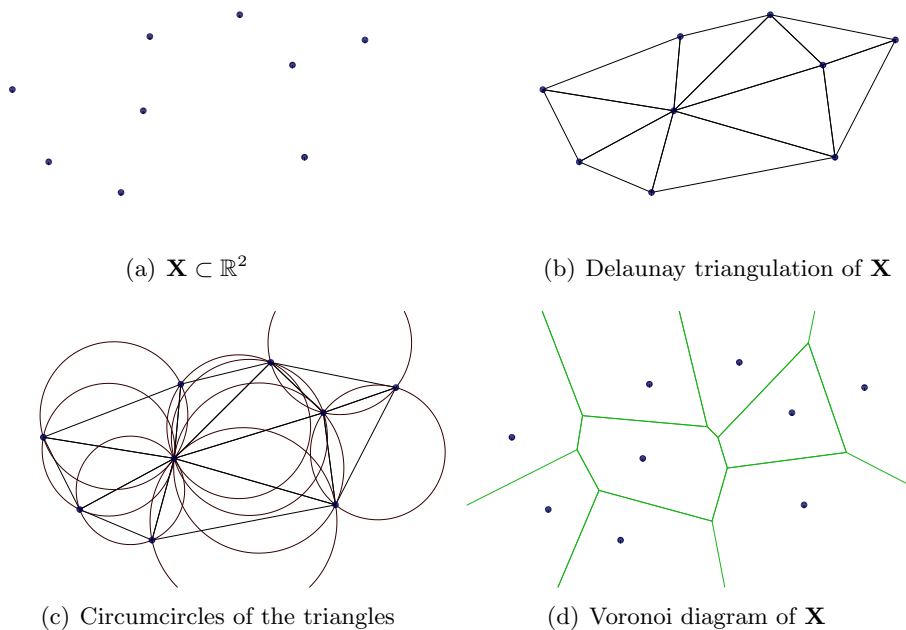


Figure 2.3: Examples of the Delaunay triangulation and the Voronoi diagram in 2D. (a) Set of points. (b) Delaunay triangulation. (c) Corresponding empty circumcircles. (d) Voronoi diagram of  $\mathbf{X}$ .

A general tessellation is composed of tiles of arbitrary shape, but here we restrict our considerations to polyhedral tessellations. If  $\Omega \subseteq \mathbb{R}^n$ , then the decomposition

$$\Omega = \bigcup_i \mathcal{T}_i$$

into  $n$ -dimensional polytopes  $\mathcal{T}_i$ , with  $\mathcal{T}_i \in \mathcal{C}$  for some polyhedral complex  $\mathcal{C}$ , is a *polyhedral tessellation* of  $\Omega$ . If  $\mathbf{X} \subset \mathcal{C}$  is the set of vertices and  $E \subset \mathcal{C}$  is the set of edges of  $\mathcal{C}$ , then the set  $\{\mathcal{T}_1, \dots, \mathcal{T}_m\} \subset \mathcal{C}$  is a *tessellation of  $\mathbf{X}$*  with vertices  $\mathbf{X}$  and edges  $E$ .

A *simplicial tessellation*, or *triangulation* of  $\Omega$  is consequently a decomposition into simplices that are elements of some simplicial complex. For any set of points in general position, one can find a simplicial tessellation. If  $T$  is a simplicial tessellation, in  $\mathbb{R}^2$ , then we use  $T_i$  instead of  $\mathcal{T}_i$  to refer to the simplexes of which  $T$  is composed.

### 2.3.4 The Delaunay Triangulation

If  $T_i$  is an  $n$ -simplex, then the unique circumscribing sphere containing its vertices is called *circumsphere* of  $T_i$ , and the sphere center is called its *circumcenter*, denoted by  $\text{cc}(T_i)$ . If  $T = \{T_1, \dots, T_m\}$  is a simplicial tessellation of a point set  $\mathbf{X} \subset \mathbb{R}^n$ , and no circumsphere contains a point from  $\mathbf{X}$  in its interior,  $T$  is called a *Delaunay triangulation*<sup>1</sup>. For any

<sup>1</sup>Although for  $n > 2$ , it does not consist of triangles but higher dimensional simplices, the term *triangulation* is common.

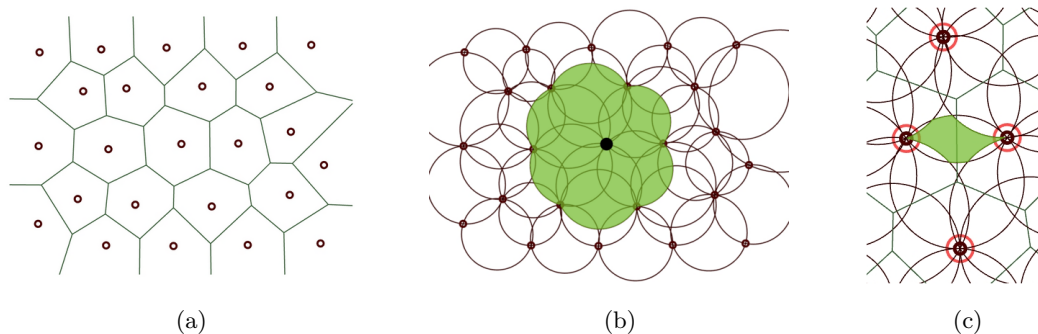


Figure 2.4: (a) Voronoi Diagram of points in  $\mathbb{R}^2$ . (b) Points in the shaded region have the bold site as one natural neighbor. (c) Points in the shaded region have the same four bold sites as natural neighbors.

point set in general position, a Delaunay triangulation exists, see Figure 2.3(a)-(c) for an example in 2D.

The most important result about Delaunay triangulations states that the set of empty circumspheres is unique. Since every set of  $n + 2$  or more points on the same empty circumsphere allows multiple triangulations into simplices without violating the empty circumsphere condition, the Delaunay triangulation of a point set is unique modulo these equivalent triangulations.

A planar Delaunay triangulation can be shown to maximize the minimum angle of all angles of the triangles.

The empty circumsphere condition allows an iterative construction of the Delaunay triangulation. We say that a simplex of a Delaunay triangulation is *in conflict* with a point if that point is contained in the interior of its circumsphere. If  $\{T_1, \dots, T_m\}$  are all simplices of a Delaunay triangulation of  $\mathbf{X}$  that are in conflict with a point  $\mathbf{x}$ , then the Delaunay triangulation of  $\mathbf{X} \cup \{\mathbf{x}\}$  can be constructed by removing  $T_1, \dots, T_m$ , and retriangulating the generated hole by simplices sharing  $\mathbf{x}$  as common vertex.

More on computation and application of the Delaunay triangulation can be found in [HD06].

### 2.3.5 The Voronoi Diagram

Next we introduce the spatial structure upon which most of this thesis is based. For an overview on Voronoi diagrams the reader may refer to [Aur91, OBSC00].

For any finite set of points  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_m\} \subset \mathbb{R}^n$ , the *Voronoi diagram* of  $\mathbf{X}$  is the *unique* polyhedral tessellation  $T = \{\mathcal{T}_{\mathbf{x}_1}, \dots, \mathcal{T}_{\mathbf{x}_m}\}$  of  $\mathbb{R}^n$  such that

$$\mathcal{T}_{\mathbf{x}_i} = \{ \mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{x}_i\| \leq \|\mathbf{x} - \mathbf{x}_j\|, \mathbf{x}_j \in \mathbf{X} \},$$

as shown for 2D in Figure 2.3(d). The members of  $\mathbf{X}$  are called *Voronoi sites* and the vertices *Voronoi vertices*. A polyhedron  $\mathcal{T}_{\mathbf{x}}$  is called *Voronoi tile* of  $\mathbf{x}$ , where we use

the abbreviation  $\mathcal{T}_i := \mathcal{T}_{\mathbf{x}_i}$  to refer to tiles of indexed points. For any two tiles  $\mathcal{T}_i, \mathcal{T}_j$  intersecting in an  $n - 1$ -dimensional polyhedron, we call  $\mathcal{F}_{ij} = \mathcal{T}_i \cap \mathcal{T}_j$  the *Voronoi facet* separating the tiles, and say that  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are *natural neighbors* in  $\mathbf{X}$ . In the context of Voronoi diagrams, we use  $N(\mathbf{x}) \subseteq \mathbf{X}$  to denote the set of natural neighbors of a point  $\mathbf{x} \in \mathbf{X}$ , and  $N_i = \{j : \mathbf{x}_j \in N(\mathbf{x}_i)\}$  to denote the set of indices of the natural neighbors of  $\mathbf{x}_i$ .

The neighborhood relation in Voronoi diagrams induces several related geometric structures. The set of points which have a certain site as natural neighbor is called the *support* of that site, shown in Figure 2.4(b). A set of points sharing the same set of natural neighbors is part of the arrangement of Delaunay circumspheres, as illustrated for 2D in Figure 2.4(c).

**Remark 2.2** *This definition of natural neighbors excludes Voronoi sites sharing a common empty circumsphere, whereas the slightly different definition*

$$\mathbf{x}_i, \mathbf{x}_j \text{ are natural neighbors} \iff \mathcal{T}_i \cap \mathcal{T}_j \neq \emptyset$$

*includes sites on a common empty circumsphere. For the first definition, the natural neighbors of a point in the Voronoi diagram of  $\mathbf{X}$  are a subset of the vertex neighbors of that point in the Delaunay triangulation of  $\mathbf{X}$ , while for the second definition, the subset relation is reversed.*

*For our purposes, the association of natural neighbors of a point  $\mathbf{x} \in \mathbb{R}^n$  with the vertex neighbors of that point in a Delaunay triangulation is sufficient, since the Voronoi tiles of ambiguous neighbors intersect in facets of dimension  $n - 2$  or less and can be ignored in all our considerations.*

If  $T = \{\mathcal{T}_{\mathbf{x}}, \mathcal{T}_{\mathbf{x}_1}, \dots, \mathcal{T}_{\mathbf{x}_m}\}$  is the Voronoi diagram of  $\mathbf{X} = \{\mathbf{x}, \mathbf{x}_1, \dots, \mathbf{x}_m\}$ , and  $T^{(\mathbf{x})} = \{\mathcal{T}_1^{(\mathbf{x})}, \dots, \mathcal{T}_m^{(\mathbf{x})}\}$  is the Voronoi diagram of  $\mathbf{X} \setminus \{\mathbf{x}\}$ , then we call  $T^{(\mathbf{x})}$  the *parent Voronoi diagram* of  $\mathbf{x}$  in  $\mathbf{X}$ , and refer to  $\mathcal{T}_{\mathbf{x}}$  as the *virtual tile* of  $\mathbf{x}$  in  $\mathbf{X} \setminus \{\mathbf{x}\}$ .

The fundamental geometric duality between the Voronoi diagram and the Delaunay triangulation is captured in the following result.

**Corollary 2.3 (Duality of Voronoi Diagram and Delaunay Triangulation)**

*Let  $\mathbf{X}$  be a finite point set in  $\mathbb{R}^n$ , and  $T^D(\mathbf{X}) = \{T_1, \dots, T_k\}$  its Delaunay triangulation, and  $T^V(\mathbf{X}) = \{\mathcal{T}_1, \dots, \mathcal{T}_l\}$  its Voronoi diagram. Then,  $T^D(\mathbf{X})$  is dual to  $T^V(\mathbf{X})$  in the sense that*

1. *each vertex in  $T^D(\mathbf{X})$  corresponds to a Voronoi tile in  $T^V(\mathbf{X})$ ,*
2. *for each vertex  $\mathbf{x}_i \in \mathcal{V}(\mathbf{X})$ ,  $\mathcal{T}_i = \mathcal{C}(\{\text{cc}(T_j) : \mathbf{x}_i \in T_j\})$ , i.e., it is the convex hull of circumcenters of all Delaunay simplices adjacent to the Voronoi site,*
3. *each simplex edge in  $T^D$  is dual to a Voronoi tile facet in  $T^V$ ,*
4. *for each non-boundary edge, the dual tile facet is the convex hull of the circumcenters of all Delaunay simplices adjacent to the Delaunay edge, and is degenerate if the edge in  $T^D(\mathbf{X})$  is ambiguous,*
5. *the set of vertex neighbors in the Delaunay triangulation contains the set of natural neighbors.*

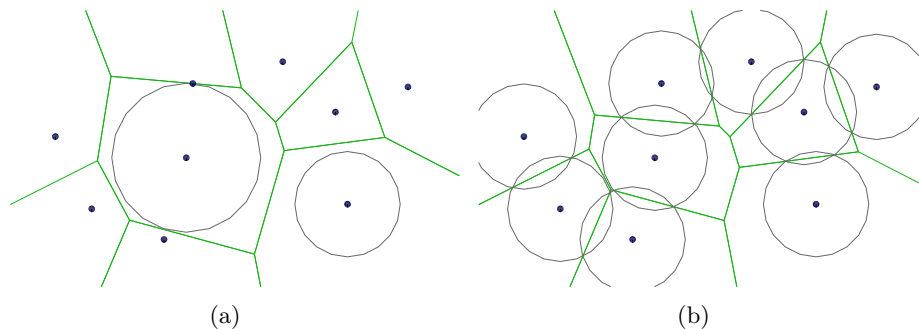


Figure 2.5: Example of the power diagram in 2D, where the circles around each vertex  $\mathbf{x}_i$  have radius  $\sqrt{w_i}$ . (a) Bisectors bounding the Voronoi tiles are displaced by the power weights. (b) Power diagram with uniform power weights: it is identical to the classical Voronoi diagram.

Because of this duality, algorithmic access to the entities of the Voronoi diagram is possible through the Delaunay triangulation, for which efficient and well-studied data structures and algorithms exist. In particular, one of the most often required operations in subsequent algorithms is the identification of natural neighbors of a point in a point set. The vertices of all Delaunay simplices in conflict with a point are its natural neighbors.

Being based on the Euclidean distance, this is also referred to as the *classical Voronoi diagram*. But for a point set in  $\mathbb{R}^n$ , every metric  $\mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$  defines a *generalized Voronoi diagram*, which is a general tessellation of  $\mathbb{R}^n$ . A generalized Voronoi diagram usually lacks most of the properties of the classical Voronoi diagram, i.e., its tiles  $\mathcal{T}_i$  need not be convex, nor connected. However, the generalization introduced in the following produces a polyhedral tessellation like the classical Voronoi diagram.

### 2.3.6 The Power Diagram

The fact that the classical Voronoi diagram is composed of convex polyhedra and is furthermore dual to the Delaunay triangulation makes it very useful in practice. One generalized Voronoi diagram that mostly preserves these beneficial properties is the *power diagram*, also called *Laguerre Voronoi diagram*, which is defined as follows.

A *weighted point* is a pair  $(\mathbf{x}, w) \in \mathbb{R}^n \times \mathbb{R}$ , and  $w$  is its *power weight*. The *power distance* or *Laguerre distance* of two weighted points  $(\mathbf{x}_1, w_1)$ ,  $(\mathbf{x}_2, w_2)$  is defined by

$$d^{\text{pow}}(\mathbf{x}_1, \mathbf{x}_2) = d(\mathbf{x}_1, \mathbf{x}_2)^2 - w_1 - w_2,$$

and of a weighted point  $(\mathbf{x}_1, w_1)$  and a point  $\mathbf{x}$  as  $d^{\text{pow}}(\mathbf{x}_1, \mathbf{x}) = d(\mathbf{x}_1, \mathbf{x})^2 - w_1$ . Every weighted point  $(\mathbf{x}, w)$  defines a sphere with center  $\mathbf{x}$  and a possibly imaginary radius  $\sqrt{w}$ . A weighted point  $(\mathbf{x}_1, w_1)$  is *orthogonal* to another weighted point  $(\mathbf{x}_2, w_2)$  if their power distance is zero.



For the *weighted point set*  $\mathbf{X}^{\text{pow}} = \{(\mathbf{x}_1, w_1), \dots, (\mathbf{x}_m, w_m)\} \subset \mathbb{R}^n \times \mathbb{R}$ , the polyhedral tessellation with

$$\mathcal{T}_i = \{ \mathbf{x} \in \mathbb{R}^n : d^{\text{pow}}(\mathbf{x}_i, \mathbf{x}) \leq d^{\text{pow}}(\mathbf{x}_j, \mathbf{x}), \mathbf{x}_j \in \mathbf{X}^{\text{pow}} \}$$

is the *power diagram* of  $\mathbf{X}^{\text{pow}}$ , which is illustrated in Figure 2.5. In the power diagram, the *power Voronoi tiles*  $\mathcal{T}_i$  can be empty if the corresponding power weight is too small with respect to the power weights of the neighbors. Like the Voronoi diagram, the power diagram is dual to a triangulation – the *regular triangulation* whose definition from [Flo03a] we briefly repeat in the following.

The sphere associated with a weighted point that is orthogonal to  $n + 1$  weighted points is called the *orthosphere* of these  $n + 1$  points. We call a weighted point *in conflict* with an orthosphere if it has a negative power distance to the weighted point associated with the orthosphere, and vice versa.

Every triangle in a regular triangulation of  $\mathbf{X}^{\text{pow}}$  has an associated orthosphere that is not in conflict with any weighted point from  $\mathbf{X}^{\text{pow}}$ , and whose associated point lies in the corresponding power Voronoi tile. This condition is equivalent to the empty circumsphere condition of the Delaunay triangulation.

## 2.4 Graphs

More on graphs can be found in text books such as [Die05, Bol98, GR01].

We denote by  $G = (V, E)$ , the *graph* over the set of *vertices*  $V$ , with  $E \subset V \times V$  being the set of *edges*, denoted by  $e_{ij} = (v_i, v_j) \in E$ . A sequence of vertices  $\rho = (v_1, \dots, v_n)$ ,  $(v_i, v_{i+1}) \in E$ , is a *path* of length  $|\rho| = n$  if none of its edges appears more than once, i.e., edges are mutually distinct. A path is called *closed* if  $(v_n, v_1) \in E$ . We assume a path to be free of loops of length one, i.e.,  $E$  contains no edges  $(v_i, v_i)$ . Furthermore, by assuming  $E$  to be a set, we implicitly disallow parallel edges, i.e., distinct edges with common start and end points.

If  $e_{ij} \in E \iff e_{ji} \in E$  for all  $e_{ij} \in E$ , and two oppositely directed edges are considered equivalent, then  $G$  is called *undirected*, otherwise *directed*.

With weights  $W : E \rightarrow \mathbb{R}$  defined over the edges,  $G = (V, E, W)$  is a *weighted graph* with *edge weights*  $w_{ij} := W(e_{ij})$ .

If the graph vertices are elements of  $\mathbb{R}^2$ ,  $v_i \mapsto \mathbf{x}_i \in \mathbb{R}^2$ , then  $G$  is called a *planar graph* if it is possible to route the edges without intersections. A planar graph is equivalent to a general tessellation of the plane, and regions enclosed by minimal loops of edges are also referred to as the *faces* of the planar graph. One well-known result for planar graphs  $G$  as first shown by Tutte [Tut63] is the existence of an embedding in the plane such that all edges are straight lines, also called *planar straight line graph* (PSLG).

For a planar, connected, undirected graph  $G = (V, E)$  with an embedding such that  $F$  is the set of faces, its *dual graph*  $\bar{G} = (\bar{V}, \bar{E})$  is a graph that has a vertex  $\bar{v}_i \in \bar{V}$  for each face  $f_i \in F$ , and an edge  $\bar{e}_{kl} \in \bar{E}$  for each edge  $e \in E$  joining two faces  $f_k$  and  $f_l$ .

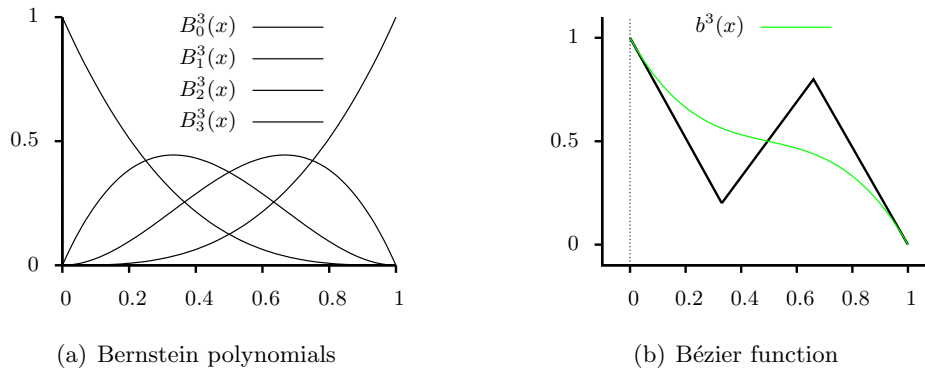


Figure 2.6: (a) Bernstein polynomials  $B_0^3, \dots, B_3^3$ . (b) Corresponding functional Bézier curve  $b^3$ .

## 2.5 Bézier and B-Spline Functions

In this section we introduce the Bernstein-Bézier form of univariate and multivariate polynomials, and B-spline functions, adapted from [HL03, PBP02]. From the functional descriptions, corresponding space curves and surface can directly be obtained by composing multiple real-valued functions into a vector-valued function.

We point out some advantages of Bézier and B-spline representations and present the de Casteljau and de Boor algorithms for their evaluation. More on these concepts can be found in [dB87, HL03, PBP02, Far02].

### 2.5.1 Bernstein Polynomials

Born from the binomial expansion of

$$1 = [\alpha + (1 - \alpha)]^d = \sum_{i=0}^d \binom{d}{i} \alpha^i (1 - \alpha)^{d-i},$$

with  $\alpha \in \mathbb{R}$  and  $d \in \mathbb{N}$ , we define the terms showing up in the sum on the right side as *Bernstein polynomials* of degree  $d$ ,

$$B_i^d(\alpha) = \binom{d}{i} \alpha^i (1 - \alpha)^{d-i}.$$

The Bernstein polynomials of degree 3 are depicted in Figure 2.6(a). By construction, Bernstein polynomials satisfy a set of desirable properties, namely

|                          |  |        |
|--------------------------|--|--------|
| partition of unity:      | $\sum_{i=0}^d B_i^d(\alpha) = 1$                         | (2.3a) |
| positivity:              | $B_i^d(\alpha) \geq 0, \quad \alpha \in [0, 1]$          | (2.3b) |
| symmetry:                | $B_i^d(\alpha) = B_{d-i}^d(1 - \alpha)$                  | (2.3c) |
| roots at $\alpha = 0, 1$ | $B_i^d(0) = B_i^d(1) = 0, \quad 0 < i < d$               | (2.3d) |
| endpoint interpolation   | $B_0^d(0) = B_d^d(1) = 1, \quad B_0^d(1) = B_d^d(0) = 0$ | (2.3e) |
| extrema                  | $\max_{\alpha \in [0,1]} B_i^d(\alpha) = B_i^d(i/d)$     | (2.3f) |

It is well-known that Bernstein polynomials  $B_0^d, \dots, B_d^d$  form a basis for all polynomials of degree  $d$ . Because the evaluation of Bernstein polynomials in  $[0, 1]$  requires exclusively convex combinations, this process is numerically stable, see Section 2.5.5.

## 2.5.2 Bézier Functions

A polynomial

$$b^d(u) = \sum_{i=0}^d c_i B_i^d(u)$$

represented by coefficients  $c_i \in \mathbb{R}$  in the Bernstein basis is called *Bézier function* of degree  $d$  with coefficients  $c_0, \dots, c_d$ . Because of property (2.3f) of Bernstein polynomials, each coefficient has quasi-local influence on the shape of the function. The *control polygon* associated with  $b^d$  has vertices  $(0/d, c_0), \dots, (d/d, c_d)$  and mimics the overall shape of the curve. A cubic Bézier function along with its control polygon is depicted in Figure 2.6(b).

The simple form of derivatives of Bernstein polynomials as discussed in, e.g., [dB87], leads to very simple formulas for the derivatives of  $b^d(u)$  at  $u = 0$  and  $u = 1$ , which is a scaled difference of coefficients,

$$\frac{d}{du} b^d(0) = d(c_1 - c_0), \quad \frac{d}{du} b^d(1) = d(c_d - c_{d-1}),$$

and generalizes to higher derivatives by

$$\begin{aligned} \frac{d^2}{du^2} b^d(0) &= d(d-1)(c_2 - 2c_1 + c_0), \\ \frac{d^m}{du^m} b^d(0) &= \frac{d!}{(d-m)!} \sum_{i=0, \dots, m} (-1)^i \binom{m}{i} c_i. \end{aligned}$$

A Bézier function can also be defined over an arbitrary interval  $[a, b] \subset \mathbb{R}$  by letting  $u(t) = (t - a)/(b - a)$ , in which case the function is given by  $b^d(u(t))$ , and by the chain rule,

$$\frac{d^m}{dt^m} b^d(0) = \frac{1}{(b - a)^m} \frac{d!}{(d - m)!} \sum_{i=0, \dots, m} (-1)^i \binom{m}{i} c_i.$$

### 2.5.3 Multivariate Bernstein Polynomials

While the above is well applicable for univariate Bernstein polynomials, notation for the multivariate case, where the domain of definition is no longer an interval but an  $n$ -dimensional simplex, suggests the use of multi-indices from Section 2.2.2.

With this notation at hand, the definition of multivariate Bernstein polynomials can be derived from  $\boldsymbol{\alpha} \in \mathbb{R}^n$ ,  $\alpha_1 + \dots + \alpha_n = 1$ , and the multinomial expansion of

$$1 = (\alpha_1 + \dots + \alpha_n)^d = \sum_{|\mathbf{i}|=d} \binom{d}{\mathbf{i}} \boldsymbol{\alpha}^{\mathbf{i}}.$$

Following [dB87, PBP02], the  $\mathbf{i}$ -th *multivariate Bernstein polynomial* of degree  $d$  is defined as

$$B_{\mathbf{i}}^d(\boldsymbol{\alpha}) = \binom{d}{\mathbf{i}} \boldsymbol{\alpha}^{\mathbf{i}}.$$

Multivariate Bernstein polynomials inherit all properties (2.3a) - (2.3f) from the univariate case, now reading

$$\text{partition of unity:} \quad \sum_{|\mathbf{i}|=d} B_{\mathbf{i}}^d(\boldsymbol{\alpha}) = 1 \quad (2.4a)$$

$$\text{positivity:} \quad B_{\mathbf{i}}^d(\boldsymbol{\alpha}) \geq 0, \quad \boldsymbol{\alpha} \in [0, 1]^n \quad (2.4b)$$

$$\text{symmetry:} \quad B_{\mathbf{i}}^d(\boldsymbol{\alpha}) = B_{\pi(\mathbf{i})}^d(\pi(\boldsymbol{\alpha})) \quad \text{for any permutation } \pi \quad (2.4c)$$

$$\text{roots} \quad B_{\mathbf{i}}^d(\mathbf{e}_j) = 0, \quad \text{for } j = 1, \dots, n, \mathbf{i} \neq d\mathbf{e}_j \quad (2.4d)$$

$$\text{endpoint interpolation} \quad B_{d\mathbf{e}_j}^d(\mathbf{e}_k) = \delta_{jk}, \quad \text{for } j = 1, \dots, n \quad (2.4e)$$

$$\text{extrema} \quad \max_{\boldsymbol{\alpha} \geq 0, |\boldsymbol{\alpha}|_1=1} B_{\mathbf{i}}^d(\boldsymbol{\alpha}) = B_{\mathbf{i}}^d(\mathbf{i}/d) \quad (2.4f)$$

### 2.5.4 Bézier Simplices

Corresponding to Bézier functions, Bézier simplices are defined using multivariate Bernstein polynomials, where the parameter  $\boldsymbol{\alpha} \in \mathbb{R}^{n+1}$  corresponds to barycentric coordinates in an  $n$ -dimensional simplex, and the multivariate definition does in fact reduce to the classical Bézier functions in the univariate case.

If  $\mathbf{x}_0, \dots, \mathbf{x}_n$  are the vertices of an  $n$ -simplex,  $\boldsymbol{\alpha} \in \mathbb{R}^{n+1}$  are barycentric coordinates in it,

and coefficients  $\{c_{\mathbf{i}}\}_{|\mathbf{i}|=d} \subset \mathbb{R}$ , with  $d \in \mathbb{N}_0$ , then

$$b^d(\boldsymbol{\alpha}) = \sum_{|\mathbf{i}|=d} B_{\mathbf{i}}^d(\boldsymbol{\alpha}) c_{\mathbf{i}}$$

is called  $n$ -variate *Bézier simplex* of degree  $d$ . To simplify the exposition, we introduce an alternative notation to address Bézier coefficients  $c_{\mathbf{i}}$ ,  $|\mathbf{i}| = d$ , by

$$\begin{aligned} c_i &:= c_{d\mathbf{e}_i}, & (\text{vertex } i) \\ c_{ij} &:= c_{\Delta_{ij}\mathbf{e}_i}, & (\text{neighbor of } c_i \text{ towards } c_j) \\ c_{ijk} &:= c_{\Delta_{ij}\Delta_{ik}\mathbf{e}_i}, & (\text{neighbor of } c_{ij} \text{ towards } c_k) \end{aligned}$$

Note that  $c_{ijk} = c_{ikj}$  due to the symmetry of multi-indices.

One important operation on Bézier simplices is the derivation along a direction  $\mathbf{v} \in \mathbb{R}^{n+1}$  in barycentric coordinates,  $v_0 + \dots + v_n = 0$ . For  $\boldsymbol{\alpha}(t) = \mathbf{p} + t\mathbf{v}$ , with  $\mathbf{p} \in \mathbb{R}^{n+1}$  barycentric coordinates of a point, and  $t \in \mathbb{R}$ ,

$$\frac{d}{d\mathbf{v}} f|_{\mathbf{p}} := \frac{d}{dt} f(\boldsymbol{\alpha}(t))|_{t=0}$$

is the directional derivative of a function  $f$  with respect to  $\mathbf{v}$  at  $\mathbf{p}$ . The general form of the directional derivative for Bézier simplices is explicitly given in [Far86, PBP02] as

$$\begin{aligned} \frac{d}{d\mathbf{v}} b^d(\mathbf{p}) &= \frac{d}{dt} b^d(\boldsymbol{\alpha}(t)) \\ &= v_0 \frac{\partial}{\partial \alpha_0} b^d(\mathbf{p}) + \dots + v_n \frac{\partial}{\partial \alpha_n} b^d(\mathbf{p}) \\ &= d \cdot \sum_{|\mathbf{j}|=d-1} g_{\mathbf{j}} B_{\mathbf{j}}^{d-1}(\mathbf{p}), \end{aligned}$$

where  $g_{\mathbf{j}} = v_0 c_{\mathbf{j}+\mathbf{e}_1} + \dots + v_n c_{\mathbf{j}+\mathbf{e}_{n+1}}$ . A direct consequence of the above relation is the formula for edge-aligned derivatives  $D_{ij} := d/d(\mathbf{x}_j - \mathbf{x}_i)$  at the simplex vertices  $\mathbf{x}_0, \dots, \mathbf{x}_n$ . It is given by

$$D_{ij} b^d(\mathbf{e}_i) := D_{\mathbf{e}_j - \mathbf{e}_i} b^d(\mathbf{e}_i) = d(c_{ij} - c_i),$$

which has a direct extension to mixed higher order derivatives by

$$D_{ij} D_{ik} b^d(\mathbf{e}_i) = d(d-1)(c_{ijk} - c_{ij} - c_{ik} + c_i). \quad (2.5)$$

### 2.5.5 De Casteljau's Algorithm

The evaluation of a Bézier simplex by computing and combining the basis functions is not efficient, and the standard method to evaluate Bézier functions is the *de Casteljau algorithm*.

If  $\{c_{\mathbf{i}}^d\}_{|\mathbf{i}|=d}$  are the coefficients of an  $n$ -variate degree  $d$  Bézier simplex  $b^d$ , then the eval-

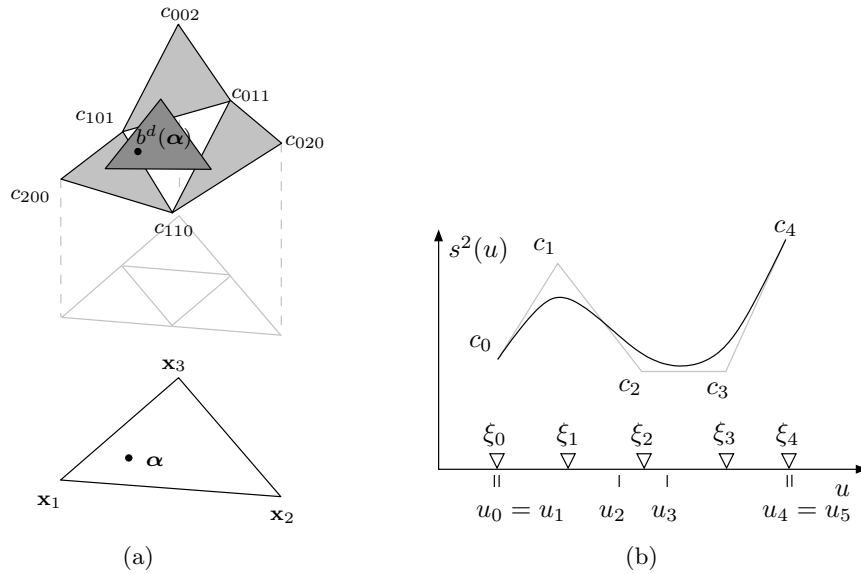


Figure 2.7: (a) De Casteljau's algorithm for a Bézier triangle of degree two. (b) The graph of a quadratic B-spline function  $b^2(u)$ .

uation of  $b^d(\alpha)$  after de Casteljau is the repeated computation of coefficients

$$c_i^{d-1}(\alpha) = \sum_{k=0}^n \alpha_k c_{i+e_k}^d,$$

until  $c_0^0(\alpha) = b^d(\alpha)$  is the desired value. A common geometrical interpretation for the 2-simplex is to imagine the control net of the Bézier simplex as the space triangles over the points  $\mathbf{c}_i = (\sum_{k=0, \dots, n} i_k \mathbf{x}_k) / d$ , as shown in Figure 2.7(a). In every step of the algorithm, a space triangle is replaced by a point which is the affine combination of its vertices with the affine coefficients  $\alpha$ .

## 2.5.6 B-Spline Functions

Spline functions are piecewise polynomial functions with prescribed smoothness between pieces. One way of constructing  $C^{k-1}$ -continuous spline functions is to piece together many degree- $k$  Bézier functions and set up continuity conditions at the joints, which is reasonable considering the relation between the derivatives and the control polygon. For example, in order to obtain a  $C^1$  continuous piecewise polynomial function, it must be ensured that the last and the first segments of the control polygon of two joining quadratic Bézier functions are collinear.

A more elegant representation of  $C^k$ -continuous, piecewise polynomial curves is given by the *B-spline* representation. The loci of the joints of the individual polynomial pieces are implicitly determined by the ordered sequence  $u_0 \leq \dots \leq u_m$ , called *knot vector* of the curve. A B-spline curve implicitly ensures the above mentioned continuity conditions under the assumption of a strictly ordered knot vector.

B-spline functions are affine combinations of *B-splines*, just as Bézier functions are affine combinations of Bernstein polynomials. If  $\mathcal{U} = (u_0, \dots, u_m)$  is a knot vector and  $d \in \mathbb{N}$ , then a *B-spline* of degree  $d$  with knot vector  $\mathcal{U}$  is recursively defined as

$$N_i^0(u) = \begin{cases} 1 & \text{if } u \in [u_i, u_{i+1}), \\ 0 & \text{else.} \end{cases}$$

and

$$N_i^d(u) = \alpha_i^{d-1} N_i^{d-1}(u) + (1 - \alpha_{i+1}^{d-1}) N_{i+1}^{d-1}(u), \quad (2.6)$$

where  $\alpha_i^{d-1} = (u - u_i)/(u_{i+d} - u_i)$ . In case of multiple knots  $u_i = u_{i+r}$ , we declare  $N_i^{r-1} = N_i^{r-1}/(u_{i+r} - u_i) = 0$ .

This recursive definition of  $N_i^d$  is also called *Cox-de Boor recursion*. B-splines exhibit similar properties as Bernstein polynomials, which means that for the set of B-splines  $N_0^d, \dots, N_{m-d-1}^d$  over a knot vector  $u_0 \leq \dots \leq u_m$ , we have

$$N_i^d(u) \begin{cases} > 0 & \text{for } u \in [u_i, u_{i+d}), \\ = 0 & \text{else,} \end{cases} \quad (\text{positivity, locality})$$

$$\sum_{i=0}^{m-d-1} N_i^d(u) = 1 \quad (\text{partition of unity})$$

$$\max_u \{N_i^d(u)\} = \frac{1}{d} \sum_{j=0}^{d-1} u_{i+j} =: \xi_i, \quad \text{where } \xi_i \text{ is called } \textit{Greville abscissa}.$$

A *B-Spline function*  $s^d(u)$  of degree  $d$  is now defined as a linear combination of  $m - d - 1$  basis Splines  $N_i^d(u)$  and coefficients  $c_i$ ,  $i = 0, \dots, m - d - 1$ ,

$$s^d(u) = \sum_{i=0}^{m-d-1} c_i N_i^d(u).$$

Similar to the control polygon of Bézier functions, B-spline functions have an associated control polygon with vertices  $(\xi_i, c_i)$ , where each vertex corresponds to the maximum of its associated basis function. An example is shown in Figure 2.7(b).

### 2.5.7 De Boor's Algorithm

The generalization of de Casteljau's algorithm for the evaluation of B-spline functions is the *de Boor algorithm*, which uses a similar recurrence relation based on repeated linear interpolation. Assume we want to evaluate the B-spline function of degree  $d$  defined over the knot vector  $(u_0, \dots, u_m)$  and coefficients  $c_0^0, \dots, c_{m-d-1}^0$  at parameter  $u \in [u_i, u_{i+1})$ . Then, for  $r = 1, \dots, d$  the de Boor algorithm defines control points  $c_{i+r}^r, \dots, c_{i+d}^r$  as follows,

$$c_j^r(u) = (1 - \alpha) c_{j-1}^{r-1}(u) + \alpha c_j^{r-1}(u) \quad \text{with } \alpha = \alpha_j^{d-r} = \frac{u - u_j}{u_{j+d+1-r} - u_j}. \quad (2.7)$$

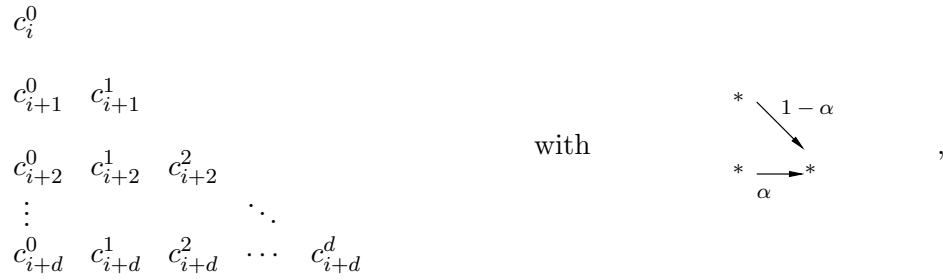


Figure 2.8: Scheme used in the de Boor algorithm to evaluate the B-spline function at a parameter value of  $u$  according to (2.7).

This can be visualized schematically by the diagram in Figure 2.8.

where  $\alpha$  depends on the position where the mask is applied. It evaluates the recursion bottom-up, starting from the control points at the lowest level  $r = 0$ . At each level, it locally generates a new control polygon for level  $r$  by evaluating the piecewise linear control polygon of level  $r - 1$  at a new set of abscissae, which corresponds to repeated insertion of knot  $u$ .

If the control points  $c_0, \dots, c_{m-d}$  are scalars, the *control polygon* of  $s$  has the vertices  $(\xi_i, c_i)$ .



## 3 Related Work

This thesis presents results on natural neighbor interpolation, which is embedded in the larger context of scattered data interpolation. We give a high-level overview on scattered data interpolation in the first part of this chapter, then describe in more detail the concept of natural neighbors and the closely linked field of natural neighbor interpolation.

### 3.1 Scattered Data Interpolation

Scattered data interpolation is the problem of finding an interpolating functional description which is as close as possible to an unknown function for which values are known only at discrete, scattered locations. More formally, given sample data sites  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_m\} \subset \mathbb{R}^n$  and data values  $Z = \{z_1, \dots, z_m\} \subset \mathbb{R}$ , find a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  that satisfies the interpolation constraint  $f(\mathbf{x}_i) = z_i$ .

If not stated otherwise, we will denote by  $\mathbf{x}_0$  the *query position* at which we want to evaluate an interpolant  $f$ . The region in which  $f(\mathbf{x}_0)$  is influenced by a data site is called that data site's *support* and leads to the distinction between schemes with *local* and *global* support. While schemes with global support usually have higher smoothness than local ones, their cost of computation makes them inapplicable for large scale data sets.

The aspects that are addressed by a multitude of scattered data interpolation schemes cover, among others:

**Support:** How is the support determined? If the size of the data set exceeds that of the available RAM, global schemes fail. Local schemes have a small memory footprint and can be computed much more efficiently, but are less smooth in general.

**Smoothness:** How often is  $f$  continuously differentiable?

**Derivative Interpolation:** Can  $f$  interpolate higher order derivatives at the data sites?

**Polynomial Precision:** Up to what order does  $f$  reproduce polynomials?

**Transfinite Interpolation:** Instead of points can one interpolate to curves or higher-dimensional manifolds? Transfinite interpolation leads to a continuous representation of the input data and usually requires considerably more effort in implementation.

**Interpolation on Manifolds:** Can the interpolation scheme still be applied if the embedding space itself is a manifold, and measurements are subject to other metrics?

**Preprocessing and Evaluation:** For a given set of interpolation objectives, what preprocessing is necessary to allow an efficient evaluation of  $f$ ? What steps and data are

then required to evaluate  $f$ ? Is all data required to be known in advance or is a subset sufficient for both preprocessing and evaluation?

**Derivatives and Integrals:** Are there exact formulas for the derivatives or integrals of  $f$ , or are numerical methods necessary?

**Extrapolation:** Local schemes typically define only  $f|_{\mathcal{C}(\mathbf{X})}$ . Does  $f$  have a meaningful definition outside  $\mathcal{C}(\mathbf{X})$ ?

**Approximation Order:** If data is sampled from a known function, how close does  $f$  get to that function with increasing sampling density? To know the approximation order of a method is to know how well it is suited to model phenomena with a certain class of governing functions.

**Non-scalar Values:** Can function values  $Z \not\subset \mathbb{R}$ , e.g.,  $Z \subset \mathbb{R}^d$ , or  $Z \subset \mathbb{C}$ , be interpolated in a meaningful way? For scalar data at the input sites, the space of possible functions can in general be described using linear combinations of neighborhood data. An application of this to the components of vector-valued data is not always in consent with the nature of the data.

**Ease of Implementation:** Methods which are simple to implement are more likely to be put to use.

Texts on scattered data interpolation in general can be found in [LF99, Wen04], and a dated but good survey in [FN91].

### 3.1.1 Inverse Distance Methods

In inverse distance methods, the contribution of a data point to the value of the interpolant is scaled by a power of the inverse distance between query position and the data point. After the introduction of this method by Shepard in [She68], many variations have evolved over time, derived from the basic form

$$f(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathbf{X}} z_i w_i(\mathbf{x}), \quad w_i(\mathbf{x}) = d(\mathbf{x}_i, \mathbf{x})^{-\mu} / \sum_{\mathbf{x}_i \in \mathbf{X}} d(\mathbf{x}_i, \mathbf{x})^{-\mu} \quad (3.1)$$

where  $d(\mathbf{x}_i, \mathbf{x}) = \|\mathbf{x}_i - \mathbf{x}\|$ , and the power  $\mu > 0$  is commonly chosen to be two. The interpolation property becomes clear if we consider  $\mathbf{x} \rightarrow \mathbf{x}_k$  for some  $\mathbf{x}_k \in \mathbf{X}$ , where all coefficients converge to a constant except  $\|\mathbf{x}_k - \mathbf{x}\|^{-\mu}$ , which goes to infinity in both nominator and denominator, leaving  $z_i$  in the limit. To avoid numerical problems arising with the poles at the data sites in (3.1), the alternative evaluation via

$$w_i(\mathbf{x}) = \prod_{j \neq i} d^{-\mu}(\mathbf{x}_i - \mathbf{x}) / \left( \sum_k \prod_{j \neq k} d^{-\mu}(\mathbf{x}_j - \mathbf{x}) \right)$$

can be used.

In its basic form, the method has significant drawbacks. Because of its global support, evaluation in a set of  $m$  points has a complexity in  $\mathcal{O}(m)$ . Furthermore, it produces flat spots at the data sites. These limitations have been overcome with the introduction of blended Taylor polynomials at the data sites to remove the flat spots and the modification

|              | $\varphi(r)$                                   | description   |
|--------------|--|---|
| global       | $r$  | $C^0$ linear  |
|              | $e^{-\alpha r^2}, \alpha > 0$                  | $C^\infty$ Gaussian, width parameterized via $\alpha$         |
|              | $(c^2 + r^2)^\beta, \beta > 0$                 | $C^\infty$ Multiquadrics, shape controlled by $c$ and $\beta$ |
|              | $(c^2 + r^2)^\beta, \beta < 0$                 | inverse Multiquadrics, shape controlled by $c$ and $\beta$    |
|              | $r^\beta, \beta > 0, \beta \notin 2\mathbb{N}$ | powers  |
|              | $r^2 \log r,$                                  | thin-plate splines  |
| com-<br>pact | $(1 - r)_+$                                    | $C^0$ locally supported polynomial                            |
|              | $(1 - r)_+^3(3r - 1)$                          | $C^2$ locally supported polynomial                            |
|              | $(1 - r)_+^4(4r - 1)$                          | $C^2$ locally supported polynomial                            |

Table 3.1: A selection of radial basis functions found in [Wen04], p. 129 and p. 188, where  $(\cdot)_+^n$  is the truncated power function  $(\max(0, \cdot))^n$ , and  $\lceil \cdot \rceil$  is the smallest integer larger than  $\cdot$ .

of the weight functions to have local support by Franke and Nielson in [FN80], and by Renka in [Ren88]. Unfortunately, even with these enhancements, the rational nature of IDW schemes causes the resulting interpolant to have far more oscillations than the input data suggests. Despite this, IDW methods are still used for their easy implementation.

### 3.1.2 Partition of Unity Method

A meta method that allows the decomposition of complicated interpolation problems into a set of local, less complex sub-problems was introduced by Franke and Nielson in [FN80]. Also known as the *partition of unity* method, it is also discussed in [Wen04], Section 15.4. In this method, a set of sufficiently overlapping regions  $\Omega_i \subset \mathbb{R}^n$  is distributed to cover the data points  $\mathbf{X} \subset \mathbb{R}^n$ . With each region  $\Omega_i$ , a blending function  $\psi_i : \mathbb{R}^n \rightarrow \mathbb{R}$  is associated that is positive inside and goes to zero smoothly at its boundary. Now, any scattered data interpolation approach can be used to construct an interpolant  $f_{\Omega_i}$  to the data at  $\mathbf{X} \cap \Omega_i$  in each region.

The global interpolant is then constructed by blending the local results,

$$f(\mathbf{x}) = \frac{1}{\sum_{\Omega_i \ni \mathbf{x}} \psi_i(\mathbf{x})} \sum_{\Omega_i \ni \mathbf{x}} \psi_i(\mathbf{x}) f_{\Omega_i}(\mathbf{x}).$$

The difficulty lies in the determination of the regions and the definition of the blending functions. Common choices for the shape of regions include discs and axis-aligned rectangles, with blending functions being radially symmetric in the first case and some tensor-product spline construction in the latter.

### 3.1.3 Radial Basis Functions

The method of radial basis functions is similar to that of inverse distance methods, yet with a more elegant mathematical foundation, as Wendland develops in [Wen04]. Its first application is due to Hardy, who proposed the multiquadric method in [Har71]. In the radial basis function framework, a set of radially symmetric *basis functions*  $\varphi : \mathbb{R}^+ \rightarrow \mathbb{R}$ ,

## Related Work

are centered at the data points, and the interpolant is defined as a weighted sum over all basis functions, evaluated at the query position,

$$f(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathbf{X}} c_i \varphi(\|\mathbf{x}_i - \mathbf{x}\|) + p(\mathbf{x}),$$

where the  $c_i$  are the coefficients of the basis functions, and  $p \in \Pi^d$  is a polynomial of total degree  $d$ . With  $p$ , the interpolant is able to reproduce degree  $d$  polynomials, and improves the numerical condition of the subsequent fitting process.

The interpolation condition leads to the linear system,

$$f(\mathbf{x}_i) = z_i, \quad \mathbf{x}_i \in \mathbf{X},$$

the solution of which delivers the unique set of coefficients  $c_i$  and the global polynomial  $p$ . Obviously, the system is symmetric and, for proper choices of  $\varphi$ , can be shown to be regular and often even positive definite. These properties make iterative methods applicable for the solution of the linear system and allows for the treatment of large data sets. Some common basis functions are listed in Table 3.1. For each function  $\varphi$  and an associated center  $\mathbf{x}_i$ , the associated support is defined as  $\{\mathbf{x} : \varphi(\|\mathbf{x}_i - \mathbf{x}\|) \neq 0\}$ . Local support refers to functions that are nonzero only in a finite, bounded domain around the center, quasi-local support to functions that converge to zero away from their center, and global support to functions that don't fit into one of the former categories.

For inhomogeneously distributed data, basis functions with a local or quasi-local support can be augmented by a radial scaling factor  $s_i$ , such that basis functions with individual support  $\varphi_i(r) := \varphi(r/s_i)$  are associated with the data sites, and the interpolant becomes

$$f(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathbf{X}} c_i \varphi_i(\|\mathbf{x}_i - \mathbf{x}\|) + p(\mathbf{x}). \quad (3.2)$$

The values of the scaling factors are usually explicitly determined such that the support of each basis function covers a certain amount of neighboring data sites.

One final aspect we mention here is the possibility to use radial basis functions for scattered data approximation. For densely sampled, noisy data, the number of basis functions can be considerably smaller than that of the data sites, as long as the combined support of the basis functions adequately covers the data set. Then, the interpolation condition (3.2) becomes an error term whose minimization delivers an approximating function. If  $\mathbf{y}_1, \dots, \mathbf{y}_k$  are the centers for basis functions  $\varphi_1, \dots, \varphi_k$ , and  $\mathbf{X} \times Z = \{(\mathbf{x}_1, z_1), \dots, (\mathbf{x}_m, z_m)\}$ , then

$$\sum_{i=1, \dots, m} \sum_{j=1, \dots, k} (c_j \varphi_j(\|\mathbf{y}_j - \mathbf{x}_i\|) + p(\mathbf{x}_i) - z_i)^2$$

is the linear least squares objective function whose minimum gives a set of coefficients for the best-fitting radial basis function approximation.

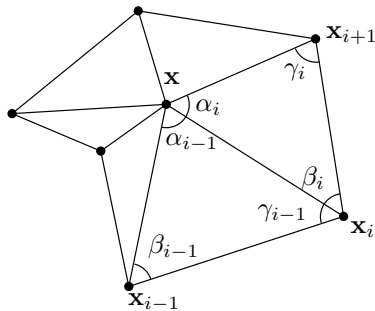


Figure 3.1: Angle notation used in the definition of barycentric coordinates.

### 3.1.4 Finite Element Methods

These interpolation methods borrow their name from the Finite Element Method for the numerical solution of partial differential equations, where the underlying domain is typically a tessellation composed of *finite elements*. Starting with a tessellation of the convex hull of the data sites,  $\bigcup_l \Omega_l = \mathcal{C}(\mathbf{X})$ , a piecewise interpolant is composed of analytic functions  $\varphi_l(\mathbf{x})|_{\Omega_l}$  over the tiles  $\Omega_l$ , where global smoothness is achieved by matching derivatives along the joints.

Most widely adopted due to their simplicity are triangulation schemes, where  $\Omega_l$  are simplices, and  $\varphi_l(\mathbf{x})$  are polynomials or rational functions in the barycentric coordinates over the simplex that interpolate function values and derivatives at the data sites and sometimes at the simplex facets. If the function derivatives are known at the vertices, several methods are available to construct smooth interpolants over a triangulation. Well-known  $C^1$  piecewise polynomial interpolants are the nine-parameter interpolant by Ženíšek [Ž73], the cubic split by Clough and Tocher [CT65], or the split introduced by Powell and Sabin [PS77].

If derivatives are not available, they need to be generated in a preprocessing step. This might be done in a global preprocessing step, such as in the minimum norm network approach of Nielson [Nie83], or the global energy minimization approach by Alfeld [Alf85]. A survey of interpolation with triangulations and quadrangulations can be found in the chapter by Zeilfelder and Seidel in [ZS02].

### 3.1.5 Generalized Polygonal Barycentric Coordinates

The introduction of generalized barycentric coordinates in Section 2.1.2 stated their common properties, yet left open the concrete choice of coordinates. The most common form are polygonal barycentric coordinates, which are defined with respect to the vertices of planar polygons and find application in surface mesh parameterization [HLS07], image warping [HF06, WSHD07], or finite element analysis [MP07, TS08].

Next, we review Wachspress coordinates, cotangent coordinates and mean value coordinates, which can be generalized to convex polytopes in  $\mathbb{R}^n$  without loss of generality. Note furthermore that inside convex polygons, all polygonal barycentric coordinates can be related to the unifying construction found by Ju et al. [JLW07].

### 3.1.5.1 Wachspress Coordinates

Probably the first appearance of generalized barycentric coordinates is due to Wachspress [Wac75], who proposed to use

$$\lambda_i^{\text{WAC}} = \hat{\lambda}_i^{\text{WAC}} / \sum_j \hat{\lambda}_j^{\text{WAC}}, \quad \hat{\lambda}_i^{\text{WAC}} = \frac{\cot \beta_i + \cot \gamma_{i-1}}{\|\mathbf{x} - \mathbf{x}_i\|^2}$$

in the context of finite element discretizations, where  $\beta_i$  and  $\gamma_{i-1}$  are as shown in Figure 3.1. In non-convex polygons, the coordinates for  $\mathbf{x}$  have undesired poles. A generalization of Wachspress coordinates to higher dimensions has been presented by Warren et al. in [WSHD07].

### 3.1.5.2 Cotangent Coordinates

The standard piecewise linear approximation of the Laplace equation in finite elements has produced local coordinates which are referred to as *discrete harmonic coordinates*, or *cotangent coordinates* because of their definition as

$$\lambda_i^{\text{COT}} = \hat{\lambda}_i^{\text{COT}} / \sum_j \hat{\lambda}_j^{\text{COT}}, \quad \hat{\lambda}_i^{\text{COT}} = \cot \beta_{i-1} + \cot \gamma_i,$$

where  $\beta_{i-1}$  and  $\gamma_i$  are as shown in Figure 3.1. In Section 3.2.5.2 another set of coordinates, Laplace coordinates  $\lambda^{\text{LAP}}$ , is introduced which have an interesting relation to Cotangent coordinates: If computed for a vertex of a Delaunay Triangulation with respect to the one-ring neighborhood, Cotangent coordinates coincide with Laplace coordinates.

### 3.1.5.3 Mean Value Coordinates

The discretization of the mean value theorem led to the set of barycentric coordinates by Floater [Flo03a], which are consequently called *mean value coordinates* and defined as

$$\lambda_i^{\text{MVC}} = \hat{\lambda}_i^{\text{MVC}} / \sum_j \hat{\lambda}_j^{\text{MVC}}, \quad \hat{\lambda}_i^{\text{MVC}} = \frac{\tan(\alpha_{i-1}/2) + \tan(\alpha_i/2)}{\|\mathbf{x} - \mathbf{x}_i\|},$$

where  $\alpha_i$  and  $\alpha_{i-1}$  are as shown in Figure 3.1. Generalizations to 3D as well as geometric constructions have been given in [FKR05, JSWD05]. Mean value coordinates form the most stable set of polygonal coordinates: they are well-defined inside and outside general polygons and even allow a generalization to sets of polygons.

However, they are guaranteed to be non-negative only in the kernel of star-shaped polygons.

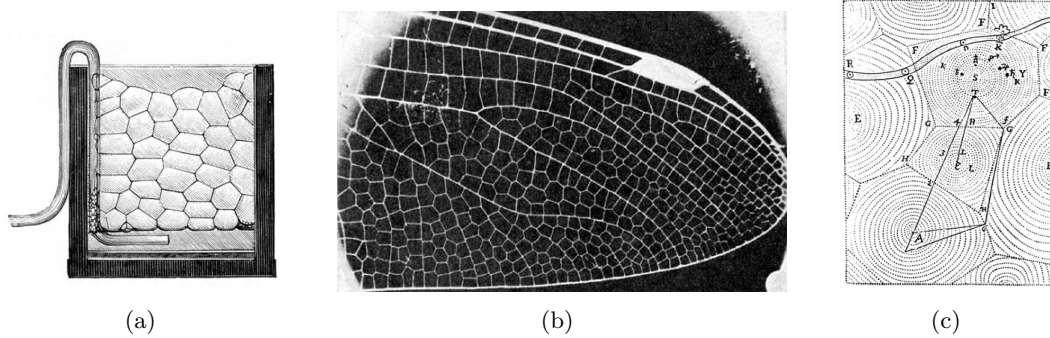


Figure 3.2: (a) Soap Bubbles in Frame. *Fig. 52 from Soap Bubbles, Their Colors and Forces which Mold Them. C.V. Boys.* (b) Part of a dragonfly's wing. *Fig. 162. From On Growth and Form. D'Arcy Thompson.* (c) Gravitational influence of stars. *Descartes. 1644.* Images taken from [Unk08].

### 3.1.5.4 A General Geometric Construction of Coordinates in Convex Simplicial Polytopes

Ju et al. showed in [JLW07] that all barycentric coordinates inside convex polygons can be generated by a unifying construction based on Stokes' theorem and a closed curve around a point inside the polygon. The shape of the curve characterizes the type of barycentric coordinate. They presented the corresponding constructions for mean value coordinates, Wachspress coordinates, and cotangent coordinates. An important result of their work with respect to natural neighbor coordinates is the proof that cotangent coordinates as they are used in the Finite Element Method and Laplace natural neighbor coordinates are identical on Delaunay triangulations.

## 3.2 Natural Neighbor Interpolation

Natural neighbor interpolation refers to local scattered data interpolation in a spatial neighborhood defined by the Voronoi diagram of the data sites. Before presenting former results on natural neighbor interpolation, we start by describing the general concept of natural neighbors and point out important aspects. We then discuss previous work on natural neighbor interpolation, which has in large parts been published in [BU06].

### 3.2.1 Natural Neighbor Concepts

At the core of the concept of natural neighbors is the notion of directional proximity that is linked to the spatial constellation of geometric objects: two geometric objects are natural neighbors if they are close to each other, and no other object is in their way.

The simplest and most widely adopted implementation of this concept is the classical Voronoi diagram of a set of points. Every point claims from its surrounding space what is closer to itself than to any other point. If the claimed spaces of two points meet, they are neighbors – natural neighbors. The more unstructured the set of points, the less regular

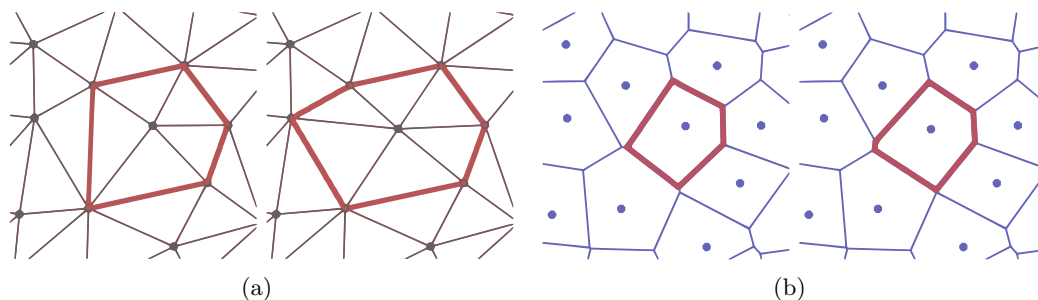


Figure 3.3: (a) A slight perturbation in a triangulation can lead to topological changes. (b) The Voronoi diagram of a point set continuously depends on the coordinates of the points.

the areas they claim, but the neighborhood structure always stays intact. This basic principle has many correspondences in nature, as Figure 3.2 suggests.

Although the classical Voronoi diagram is named after Voronoi’s work in [Vor08], it is also known under different names. Thiessen used the same spatial structure in [Thi11] for weather forecast modelling, giving it the name *Thiessen polygons*. From Dirichlet’s findings in [Dir50], the term *Dirichlet tessellation* was derived. But already Descartes came up with a generalized Voronoi diagram in [Des44], shown in Figure 3.2(c). Very likely others knew it even earlier, making it hard to do real justice in attributing that spatial structure to any specific name. The variety of fields in which the Voronoi diagram is used to model and describe spatial situations and processes is documented in an impressive survey by Aurenhammer in [Aur91]. Aspects of the Voronoi diagrams specifically interesting for computer aided geometric design are looked at by Sugihara in [Sug02].

Many geometric concepts, such as the medial axis, have a close connection to the Voronoi diagram. This has been used by Amenta et al. to develop a surface reconstruction method with provable properties [ACK01]. Alliez et al. used the directionally sensitive proximity notion of the Voronoi diagram to robustly estimate normals from unorganized point clouds in [ACSTD07]. Schussmann et al. exploited the spatial proximity expressed in natural neighbors for multi-resolution representation of scattered data in [SBHJ00]. A generalization of the Finite Element Method (FEM) that drops the requirement of an explicit tessellation of the simulation domain in the spirit of meshfree methods is the Natural Element Method (NEM) [BS95, Tra94, SMB98]. Cueto et al. presented an important generalization of the NEM to non-convex domains in [CDG00, CCD02], while Yvonnet et al. investigated certain modifications of the Voronoi diagram to model discontinuous cracks in the NEM in [YRLC04].

### 3.2.2 Properties of Natural Neighbor Interpolation

There is a close relation between the concept of natural neighbors in scattered data interpolation and local coordinates in point clouds, where terms such as *natural neighbor coordinates*, *natural neighbor interpolation* have been used to refer to Sibson’s original work as well as to the whole class of algorithms based on the idea of natural neighbors. In this thesis, we adopt the latter and refer to the class of algorithms if not stated otherwise.



The most intriguing property of natural neighbor interpolation that sets it apart from other tessellation techniques for point sets is the continuous dependence of the interpolant on the coordinates of the input data, which results from the corresponding property of the Voronoi diagram, as illustrated in Figure 3.3. Further, major advantages of natural neighbor interpolation are

- (+1) The definition of neighborhood is local, completely automatic, and copes extremely well with inhomogeneous point distributions.
- (+2) Most interpolants based on  $C^k$ -continuous natural neighbor coordinates depend  $C^k$ -continuously on the coordinates of the point cloud.
- (+3) The Voronoi diagram needs not be constructed at any time since all operations can be carried out on the Delaunay triangulation of the data sites, which is very well understood and supported by efficient data structures.
- (+4) By definition, the interpolants generalize to any dimension.

However, natural neighbor interpolation has disadvantages as well, namely

- (-1) Interpolation is defined only inside the convex hull of the data sites, with undesirable artifacts near the boundary of the convex hull.
- (-2) Globally smooth interpolants can be relatively expensive to evaluate.
- (-3) Evaluating natural neighbor interpolants in higher dimensions is computationally expensive.

A solution to (-1) this is presented in Section 8.7.

### 3.2.3 Steps of Natural Neighbor Interpolation

We consider the task of interpolating data  $(\mathbf{x}_i, z_i)$ ,  $i = 1, \dots, m$ , with  $\mathbf{x}_i \in \mathbf{X} \subset \mathbb{R}^n$ , and  $z_i \in Z \subset \mathbb{R}$  at a point  $\mathbf{x}_0 \in \mathcal{C}(\mathbf{X})$ , where we deliberately chose the index to indicate that  $\mathbf{x}_0 \in \mathbf{X}$  to facilitate the exposition. Natural neighbor interpolation at  $\mathbf{x}_0$  is composed of two basic steps:

1. Compute local coordinates  $\boldsymbol{\lambda}(\mathbf{x}_0)$  with respect to the natural neighbors  $N(\mathbf{x}_0)$ .
2. Combine the values  $z_i$  associated with  $\mathbf{x}_i \in N(\mathbf{x})$  using some blending function  $\varphi(\boldsymbol{\lambda}, Z)$ .

The straightforward choice for  $\varphi$  in the second step is  $\varphi(\boldsymbol{\lambda}, Z) = \sum_{i \in N_0} \lambda_i z_i$ , leading to the scattered data interpolant

$$f(\mathbf{x}_0) = \sum_{i \in N_0} \lambda_i z_i. \quad (3.3)$$

In Section 3.2.5 we concentrate on the computation of local coordinates, yet examples given there utilize the above choice for  $\varphi$ . Other choices for  $\varphi$  are then discussed in Section 3.2.7.

### 3.2.4 Smoothness of Natural Neighbor Interpolation

Natural neighbor interpolants come with varying smoothness, which has basically two different reasons. For most natural neighbor methods the regions of constant support, i.e., where  $N(\mathbf{x}_0)$  does not change, are the cells in the arrangement of Delaunay circum-spheres, as depicted in Figure 3.4. These regions are considerably more complex than those appearing in, e.g., the finite element method, where they are polygonal domains. Natural neighbor interpolation can be interpreted as sophisticated finite element interpolation over the arrangement of circumcircles, where  $\mathbf{x}$  is translated into  $\lambda$  such that compatibility conditions are automatically met at the joints. However, these complex regions need not be considered explicitly as they naturally arise from the structure of the Delaunay triangulation.

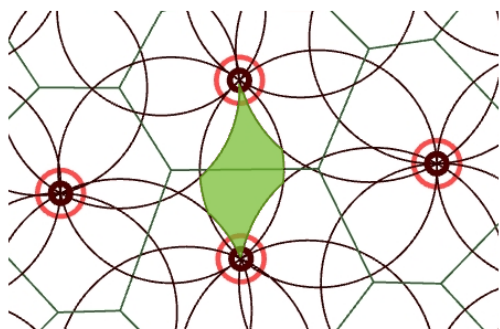


Figure 3.4: Points in the shaded region have the same set of natural neighbors.

Where  $N(\mathbf{x}_0)$  is unique, i.e., inside the region shown in Figure 3.4,  $f(\mathbf{x}_0)$  is a  $C^\infty$ -continuous function. Continuity issues arise whenever  $N(\mathbf{x}_0)$  changes, i.e.,

1.  $\mathbf{x}_0$  crosses a Delaunay circle, or
2.  $\mathbf{x}_0$  passes a data site  $\mathbf{x}_i$ .

The first problem is solved by construction of  $\lambda(\mathbf{x}_0)$ , while the second is solved by construction of some local blending function  $\varphi(\lambda, Z)$ . The next section discusses previous work addressing the first issue.

### 3.2.5 Natural Neighbor Coordinates in Point Clouds

Natural neighbor coordinates for a point  $\mathbf{x}_0$  of a point cloud  $\mathbf{X} \subset \mathbb{R}^n$  are generalized barycentric coordinates with respect to the natural neighborhood  $N(\mathbf{x}_0)$  in  $\mathbf{X}$ . In the following, we will refer to the spatial dimension by  $n$ . Recall the equivalence relation introduced in Section 2.1.2 by which a set of coefficients carrying the hat symbol “ $\hat{\cdot}$ ” is equivalent to the set of normalized coefficients without it.

Natural neighbor coordinates for  $\mathbf{x}_0 \in \mathbb{R}^n$  are based on sizes of geometric entities in its Voronoi tile  $\mathcal{T}_0$ . The rate at which these entities change with  $\mathbf{x}_0$  basically determines the smoothness of the coordinates. Whenever the query position coincides with a data site,  $\mathbf{x}_0 = \mathbf{x}_i$ , these entities are not defined, but generally the coordinates can be continuously extended for  $\mathbf{x}_0 \rightarrow \mathbf{x}_i$ , yielding  $C^0$  continuity at the data sites.

The defining property of natural neighbor coordinates is their region of influence which,

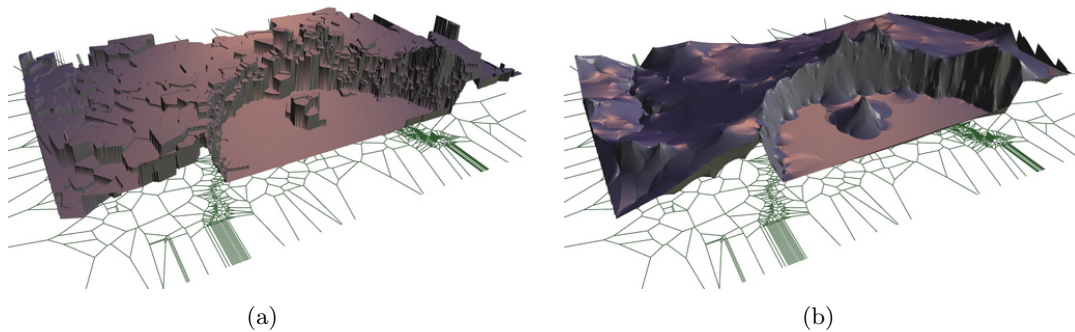


Figure 3.5: Interpolation of 1493 scattered points sampled from the crater lake data set. The original data set is due to US geological survey with  $344 \cdot 463$  points. (a) The nearest neighbor interpolant is piecewise constant and discontinuous along the edges of the Voronoi diagram. (b) The Laplace interpolant is continuous with derivative discontinuities along the Delaunay circles.

for the coordinate associated with  $\mathbf{x}_i$  is the interior of the union of all Delaunay circum-spheres passing through  $\mathbf{x}_i$ , depicted in Figure 2.4(b). The following definition captures this result.

**Definition 3.1 (Natural Neighbor Coordinates)** *Let  $\mathbf{X} \subset \mathbb{R}^n$  be a set of points, and  $N(\mathbf{x})$  the set of natural neighbors of  $\mathbf{x}$ . Any set of convex coordinates  $\lambda$  of  $\mathbf{x}$  with respect to  $N(\mathbf{x})$  that satisfies*

1.  $\lambda_i > 0 \iff \mathbf{x}_i \in N(\mathbf{x})$ ,
2.  $\lambda$  is continuous with respect to  $\mathbf{x}$ ,

*is called a set of natural neighbor coordinates of  $\mathbf{x}$  in  $\mathbf{X}$ , or just natural neighbor coordinates of  $\mathbf{x}$ .*

In the following we present a comprehensive selection of local coordinate definitions that fit into the framework of natural neighbor coordinates.

### 3.2.5.1 Nearest Neighbors

Although not local coordinates by definition, the *nearest neighbors* of a point  $\mathbf{x}_0$  lead to a set of coefficients

$$\lambda_i^{\text{NEAR}} := \begin{cases} 1, & \mathbf{x}_0 \in \mathcal{T}_i, \\ 0, & \text{otherwise.} \end{cases}$$

The resulting scattered data interpolant  $f^{\text{NEAR}}$  was first used by Thiessen in [Thi11] and is the most basic application of natural neighbor concepts. It is discontinuous across the boundaries of the Voronoi tiles, as can clearly be seen in Figure 3.5(a), which limits its importance in many applications. However, this approach is still useful in case of discrete data that does not lend to the computation of intermediate values and thus is discontinuous by nature.

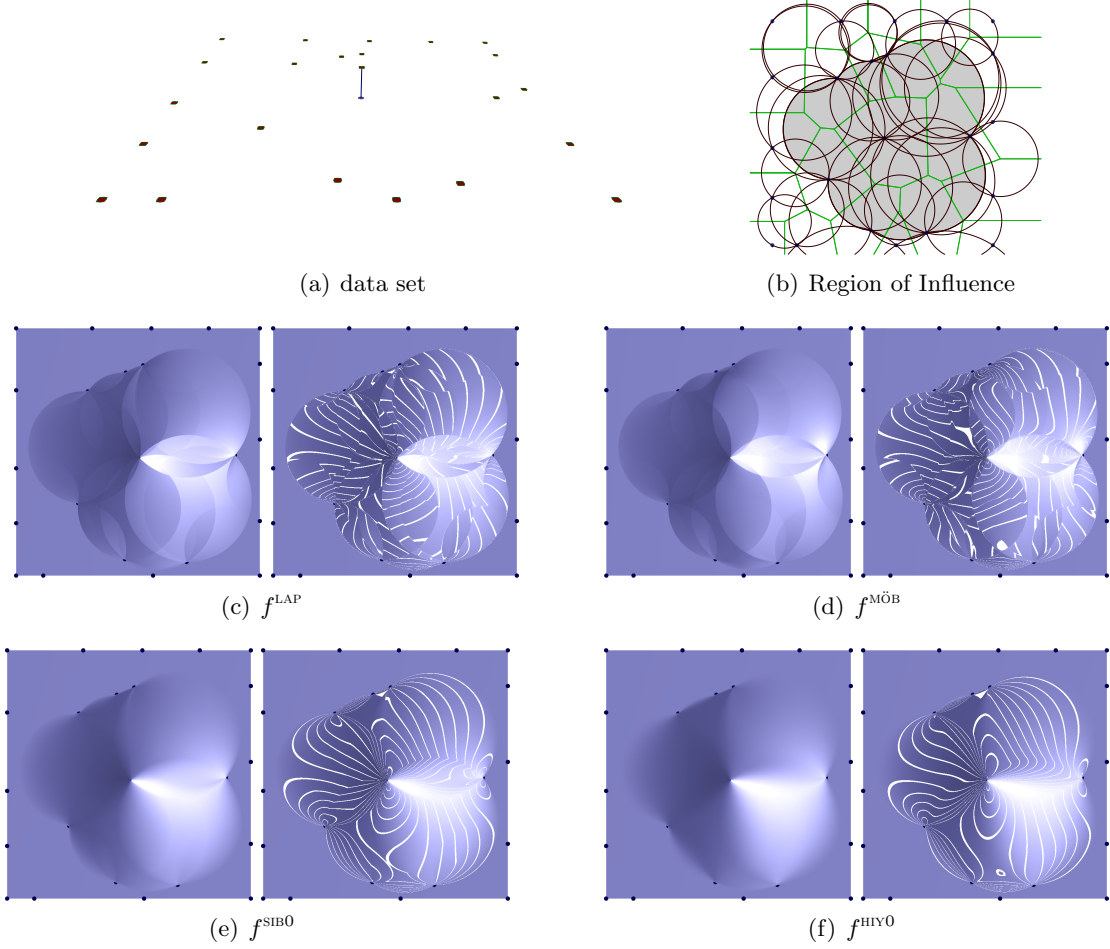


Figure 3.6: (a) Point set  $\mathbf{X}$ , the center point receiving a value of one, the remaining a value of zero. (b) Top view with Delaunay circumcircles and the region of influence of the middle point in grey. (c)-(f) Left show basis functions of  $f^{\text{LAP}}$ ,  $f^{\text{MöB}}$ ,  $f^{\text{SIB}0}$ ,  $f^{\text{HY}0}$ , as a height-field from above with specular lighting, which is sensitive to second derivatives in the function. The right images show reflection lines which are particularly suited to expose  $C^2$  discontinuities.

### 3.2.5.2 Laplace Coordinates

A set of  $C^{n-2}$ -smooth local coordinates has been proposed by different authors as *Laplace-* or *Non-Sibsonian coordinates* in [CFL82, BIK<sup>+</sup>97, Sug99].

From the volumes of the facets  $\mathcal{F}_1, \dots, \mathcal{F}_m$  of  $\mathcal{T}_0$ , and the distance  $r_i = \|\mathbf{x}_0 - \mathbf{x}_i\|$ ,  $i \in N_0$ , they are defined as

$$\lambda_i^{\text{LAP}} = \hat{\lambda}_i^{\text{LAP}} / \sum_j \hat{\lambda}_j^{\text{LAP}}, \quad \hat{\lambda}_i^{\text{LAP}} := \frac{\text{Vol}(n-1, \mathcal{F}_i)}{r_i},$$

which is shown for 2D in Figure 3.7(a). These coordinates and the resulting interpolant  $f^{\text{LAP}}$  are continuous in  $\mathcal{C}(\mathbf{X})$  and have derivative discontinuities at the data sites. For

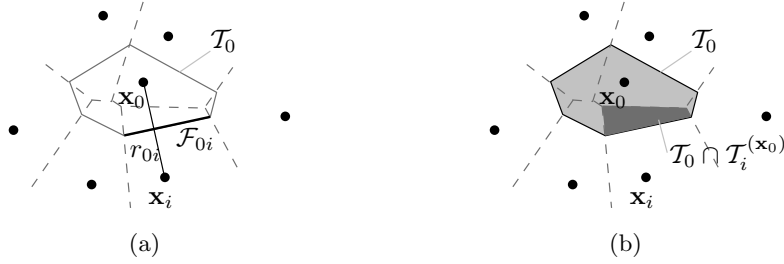


Figure 3.7: Geometric entities involved in the computation of (a) Laplace coordinates, (b) Sibson coordinates.

$\mathbf{X} \subset \mathbb{R}^n$ , we find that  $\lambda_i^{\text{LAP}}$  is  $C^{n-2}$  on the Delaunay circumspheres. The basis function for  $f^{\text{LAP}}$  is illustrated in Figure 3.6(c), and its application to a real-world data set is shown in Figure 3.5(b). The derivative discontinuities are clearly visible in Figure 3.6(c) as jumps in the reflection lines. Different proofs for  $\lambda_i^{\text{LAP}}$  satisfying the local coordinate property (2.1) have been given in [HS00b, BIK<sup>+</sup>97].

Laplace coordinates  $\lambda^{\text{LAP}}$  coincide with cotangent coordinates  $\lambda^{\text{COT}}$  if computed in the one-ring neighborhood of the Delaunay triangulation. We discovered this correspondence just about when an alternative proof has been given in [JLW07]. However, our proof takes a different perspective and is presented next.

**Theorem 3.2** *Cotangent coordinates  $\hat{\lambda}^{\text{COT}}$  and Laplace coordinates  $\hat{\lambda}^{\text{LAP}}$  differ by a factor of two.*

**Proof 3.3** *Consider the setting in Figure 3.8(a), showing two Delaunay triangles sharing the edge  $(\mathbf{x}, \mathbf{x}_i)$ . We start by noting that the Laplace coordinate of  $\mathbf{x}$  with respect to  $\mathbf{x}_i$  is given by  $\hat{\lambda}_i^{\text{LAP}} = (h_i^- + h_i^+)/r_i$ . In the following we only consider  $h_i^+/r_i$ , corresponding to the right triangle, the rest then follows by symmetry.*

*Figure 3.8(b) shows how the circumcenter of the triangle partitions it into three pairs of symmetric triangles, with  $\gamma_i = \delta' + \delta''$ . Substituting this into  $2\delta + 2\delta' + 2\delta'' = \pi$  and rearranging terms gives  $\delta = \pi/2 - \gamma_i$ .*

*Now, as indicated in Figure 3.8(c), with  $h_i^+ = r_i/2 \cdot \tan \delta = r_i/2 \cdot \cot \gamma_i$  we see that*

$$\frac{h_i^+}{r_i} = \frac{r_i \cot \gamma_i}{2r_i} = \frac{\cot \gamma_i}{2}, \quad \text{and} \quad \frac{h_i^-}{r_i} = \frac{\cot \beta_{i-1}}{2},$$

*by symmetry. Consequently,*

$$\hat{\lambda}_i^{\text{LAP}} = \frac{h_i^- + h_i^+}{r_i} = \frac{1}{2}(\cot \gamma_i + \cot \beta_{i-1}) = \frac{1}{2}\hat{\lambda}_i^{\text{COT}}$$

*For obtuse triangles, the cotangent becomes negative and the signed sum  $h_i^+ + h_i^-$  still gives the correct distance between the circumcenters.*

□

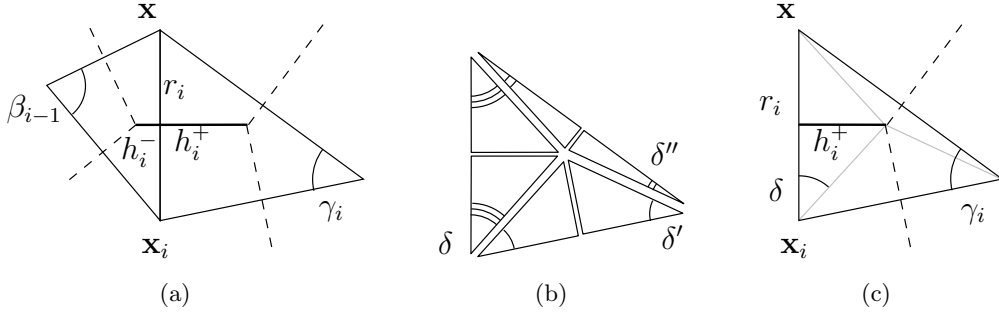


Figure 3.8: Notation used in the proof that  $\hat{\lambda}^{\text{COT}} = 2\hat{\lambda}^{\text{LAP}}$ .

### 3.2.5.3 Sibson's Coordinates

The first appearance of natural neighbor coordinates is due to Sibson [Sib80], who discovered a vector identity that involved the Voronoi diagram and leads to a set of natural neighbor coordinates, *Sibson coordinates*. It is most illustrative to describe Sibson coordinates for a point  $\mathbf{x}_0$  as the ratio of volumes that are “stolen” by the Voronoi tile of  $\mathbf{x}_0$  from the Voronoi tiles of its natural neighbors before  $\mathbf{x}_0$  was added to the Voronoi diagram.

With  $\mathcal{T}^{(\mathbf{x}_0)}$  denoting the parent Voronoi tile from Section 2.3.5, illustrated for 2D in Figure 3.7(b), this identity is formally given by

$$\text{Vol}(n, \mathcal{T}_0)\mathbf{x} = \sum_{i \in N_0} \text{Vol}(n, \mathcal{T}_0 \cap \mathcal{T}_i^{(\mathbf{x}_0)})\mathbf{x}_i,$$

where Sibson's homogeneous coordinates follow directly as

$$\lambda_i^{\text{SIB}} = \hat{\lambda}_i^{\text{SIB}} / \sum_j \hat{\lambda}_j^{\text{SIB}}, \quad \hat{\lambda}_i^{\text{SIB}} := \text{Vol}(n, \mathcal{T}_0 \cap \mathcal{T}_i^{(\mathbf{x}_0)}).$$

It can be argued that every intersection volume changes proportional to the  $n$ -th power of the distance of  $\mathbf{x}_0$  to a specific natural neighbor. The fact that the volumes have dimension  $n$  results in  $\lambda^{\text{SIB}}$  being  $C^{n-1}$  continuous in  $\mathcal{C}(\mathbf{X}) \setminus \mathbf{X}$ .

Figure 3.6(f) shows the basis function of  $f^{\text{SIB}0}$ , at the same time giving a height-field visualization of the coordinate associated with the center point, and showing where the  $C^1$  discontinuities arise.

Properties of Sibson coordinates received close attention by Farin [Far90] and Piper [Pip92]. With  $\mathbf{m}_i$  denoting the centroid of facet  $\mathcal{F}_i$  of  $\mathcal{T}_0$ , the explicit formula for the gradient of  $\lambda_i$  is

$$\nabla \lambda_i^{\text{SIB}} = \text{Vol}(n-1, \mathcal{F}_i) \cdot (\mathbf{m}_i - \mathbf{x}_0) / r_i. \quad (3.4)$$

Different proofs for  $\lambda_i^{\text{SIB}}$  satisfying the local coordinate property (2.1) have been given by [Sib80, Pip92, HS00b].

Piper showed in [Pip92] that Sibson coordinates smoothly depend on the point coordinates of the point set.

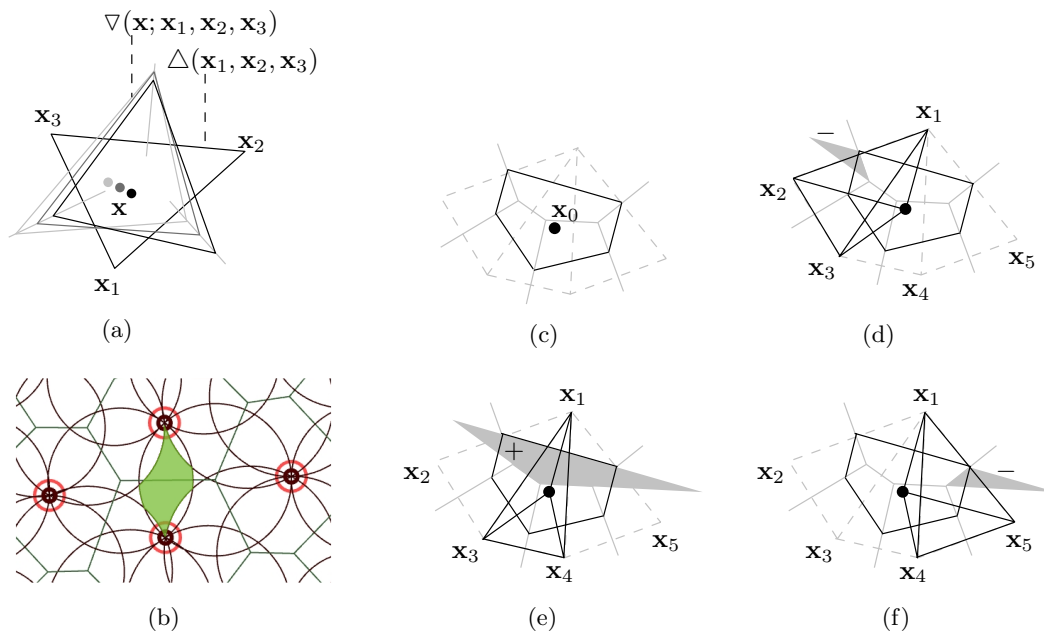


Figure 3.9: Illustration of the signed decomposition of the subtile  $\mathcal{T}_0 \cap \mathcal{T}_1^{(x_0)}$ . (a) dual triangle  $\nabla(x; \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$  of  $\Delta(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$  (b) area of constant neighborhood: every  $\mathbf{x}$  in the shaded region has the same set  $N(\mathbf{x})$ . (c) Voronoi tile of  $\mathbf{x}_0$  (d) area of the triangle  $\Delta(\mathbf{c}_{123}, \mathbf{c}_{130}, \mathbf{c}_{102})$ , (e) area of the triangle  $\Delta(\mathbf{c}_{134}, \mathbf{c}_{140}, \mathbf{c}_{103})$ , (f) area of the triangle  $\Delta(\mathbf{c}_{145}, \mathbf{c}_{150}, \mathbf{c}_{104})$ , where  $\mathbf{c}_{ijk}$  denotes the circumcenter of  $\Delta(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ . While iterating in counterclockwise order along vertices  $\mathbf{x}_i$  adjacent to  $\mathbf{x}_1$ , the scheme is  $\Delta(\mathbf{c}_{1i(i+1)}, \mathbf{c}_{1(i+1)0}, \mathbf{c}_{10i})$ .

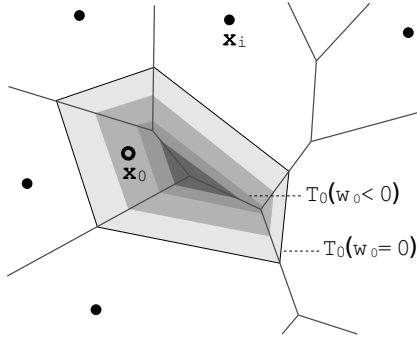
### 3.2.5.4 Watson's Construction for Sibson's Coordinates

Watson proposed to compute the areas involved in the definition of Sibson coordinates by a signed triangle decomposition in [Wat92], p. 81. Besides its straightforward application in higher dimensions, this method gives an explicit construction of the rational function describing Sibson coordinates inside the regions of constant neighborhood.

The algorithm computes intersected Voronoi tile areas that define Sibson coordinates by accumulating the signed areas of *dual triangles*. The dual triangle of  $\Delta(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$  with respect to a point  $\mathbf{x}_i$  is thereby defined as  $\nabla(\mathbf{x}_i; \mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c) := \Delta(\mathbf{c}_{iab}, \mathbf{c}_{ibc}, \mathbf{c}_{ica})$ , where  $\mathbf{c}_{abc}$  is the circumcenter of  $\Delta(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$ .

To compute the area covered by the Voronoi subtile associated with a certain neighbor, say  $\mathcal{T}_0 \cap \mathcal{T}_1^{(x_0)}$  associated with  $\mathbf{x}_1$  in Figure 3.9, the construction traverses the counterclockwise ordered set of vertices adjacent to  $\mathbf{x}_1$  and accumulates the signed areas of  $\nabla(\mathbf{x}_1; \mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_0)$ .

This method has serious numerical issues if the query position  $\mathbf{x}_0$  happens to lie on or close to a Delaunay edge: the circumcenters are computed from degenerate triangles, and signed areas become infinite. A workaround has been proposed by Hiyoshi in his work on stable computation of natural neighbor coordinates in [Hiy05].



• Figure 3.10: The power diagram of the weighted point set  $\mathbf{X}^{\text{pow}}$ , the black dots showing  $\mathbf{X}$ , for uniform power weights  $w_i = 0$ ,  $i \geq 1$ , and the power tile  $\mathcal{T}_0(w_0)$  for different values of  $w_0 \leq 0$ .

### 3.2.5.5 Hiyoshi's Coordinates

Hiyoshi and Sugihara [HS00b] proposed a generalization of Laplace and Sibson coordinates based on an integral expression of  $\lambda_i^{\text{LAP}}$  in the power diagram. As illustrated in Figure 3.10, the area covered by  $\mathcal{T}_0(w_0)$  in the power diagram shrinks with decreasing  $w_0$  until it vanishes for  $w_0 = \sup \{ w_0 : \mathcal{T}_0(w_0) = \emptyset \}$ , see Figure 3.10. Hiyoshi observed that for any  $\mathcal{T}_0(w_0) \neq \emptyset$ , the construction for Laplace coordinates applied to  $\mathcal{T}_0(w_0)$  renders valid local coordinates in the power diagram, and that the areas swept by the line segments are those of the sub-tiles used in the computation of Sibson coordinates. Following this observation, he defines a family of local coordinates, coined *order  $k$  standard coordinates*, which reflects the fact that in 2D, the coordinates are  $C^k$  everywhere except at the Voronoi sites.

If  $\hat{\lambda}_i^{\text{LAP}}(w) := \hat{\lambda}_i^{\text{LAP}}|_{w_0=w}$  are the Laplace coordinates of  $\mathbf{x}_0$  for a power weight  $w_0 = w$ , then homogeneous order  $k$  standard coordinates are defined as

$$\hat{\lambda}_i^k = \hat{\lambda}_i^k(0), \quad \hat{\lambda}_i^k(w) = \int_{w_0}^w \hat{\lambda}_i^{k-1}(t) dt, \quad \hat{\lambda}_i^0(w) = \hat{\lambda}_i^{\text{LAP}}(w). \quad (3.5)$$

We refer to standard coordinates of order two explicitly as *Hiyoshi coordinates*

$$\lambda_i^{\text{HIY}} = \hat{\lambda}_i^{\text{HIY}} / \sum_j \hat{\lambda}_j^{\text{HIY}}, \quad \lambda_i^{\text{HIY}} := \lambda_i^2.$$

In [Hiy05] Hiyoshi restated the above as

$$\hat{\lambda}_i^k := \frac{1}{(k-1)!} \int_{\mathbf{x} \in \mathcal{T}_i(\mathbf{x}_0) \cap \mathcal{T}_0} ((\mathbf{x}_0 - \mathbf{x}_i) \cdot (\mathbf{x} - \mathbf{m}_i))^{k-1} |d\mathbf{x}|,$$

where  $\mathbf{m}_i$  is the centroid of the corresponding Voronoi facet, and  $|d\mathbf{x}|$  denotes the area integral. For  $k = 0, 1$ , Hiyoshi coordinates coincide with Laplace and Sibson's coordinates. For  $k > 1$ , these coordinates are  $C^{k+n-2}$  in  $\mathcal{C}(\mathbf{X}) \setminus \mathbf{X}$ . As Hiyoshi pointed out in [Hiy05], the limit  $k \rightarrow \infty$  does not lead to  $C^\infty$  coordinates but to the piecewise linear interpolant on the Delaunay tessellation.

The basis function of  $f^{\text{HIY}0}$  can be seen in Figure 3.6(f), where the subtle fact that the reflection lines do not show any cusps is due to the  $C^2$  continuity of  $\lambda^{\text{HIY}0}$  in  $\mathcal{C}(\mathbf{X}) \setminus \mathbf{X}$ .



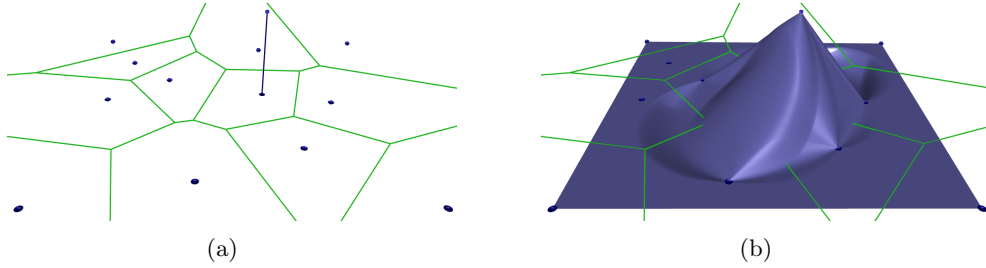


Figure 3.11: (a) Data set. (b) Basis function for Brown coordinates  $\lambda^{\text{BRO}}$ . Notice the dents indicating negative values.

### 3.2.5.6 Brown Coordinates in the Delaunay Triangulation

In [Bro97], Brown introduces a general framework to construct continuous coordinates in arbitrary point clouds in the plane based on barycentric coordinates with respect to Delaunay triangles, which are combined using the partition of unity approach with blending functions over the Delaunay circumcircles. This approach extends to higher dimensions without restrictions.

Given a Delaunay triangulation  $T = \{T_1, \dots, T_m\}$  of  $\mathbf{X}$ , a point  $\mathbf{x}$  has possibly non-convex barycentric coordinates  $\lambda_1(\mathbf{x}), \dots, \lambda_m(\mathbf{x})$  with respect to the vertices of the individual triangles. To facilitate notation, we assume without loss of generality that  $\lambda_i = (\lambda_{i1}, \dots, \lambda_{im}) \in \mathbb{R}^m$ , where  $\lambda_{ij} = 0$  if  $\mathbf{x}_j \notin T_i$ . This allows to compute a new set of generalized barycentric coordinates as an affine combination of a set of triangular barycentric coordinates.

Assume that  $T_1, \dots, T_k$  are the triangles containing  $\mathbf{x}$  in their circumcircles. In Brown's method,  $\lambda_1, \dots, \lambda_k$  are mixed using blending functions  $\psi_1, \dots, \psi_k : \mathbb{R}^2 \rightarrow \mathbb{R}$ , over the circumcircles with radius  $r_i$ , centered at the circumcenters  $\mathbf{c}_i$ . If every  $\psi_i$  is  $C^k$ -continuous and strictly positive exactly when inside the circumcircle, then

$$\lambda^{\text{BRO}}(\mathbf{x}) = \frac{1}{\sum_{i=1}^k \psi_i(\mathbf{x})} \sum_{i=1}^k \psi_i(\mathbf{x}) \lambda_i(\mathbf{x})$$

yields non-convex natural neighbor coordinates for  $\mathbf{x}$  in  $\mathbf{X}$  that are  $C^k$  continuous in  $\mathcal{L}(\mathbf{X}) \setminus \mathbf{X}$ .

An example for such a weight function was given by Brown as

$$\psi_i(\mathbf{x}) = \begin{cases} (\|\mathbf{x} - \mathbf{c}_i\|^2 - r_i^2)^4 & \text{if } \|\mathbf{x} - \mathbf{c}_i\| < r_i, \\ 0 & \text{else.} \end{cases} \quad (3.6)$$

**Remark 3.4** In his paper [Bro97], Brown claimed in Lemma 2.5 that inside the union of the interior of all circumcircles,  $\lim_{\mathbf{x} \rightarrow \mathbf{x}_i} \lambda_j(\mathbf{x}) = \delta_{ij}$ . Figure Figure 3.12 indicates that in case of co-circular vertices, this Lemma does not hold. The proof of this lemma considered a simple setting with four points,  $\mathbf{x}_1, \dots, \mathbf{x}_4$ , and two triangles  $T_1, T_2$ , where  $\mathbf{x}_1$  and  $\mathbf{x}_4$  are not connected by a Delaunay edge. It started with the assumption that there exists a

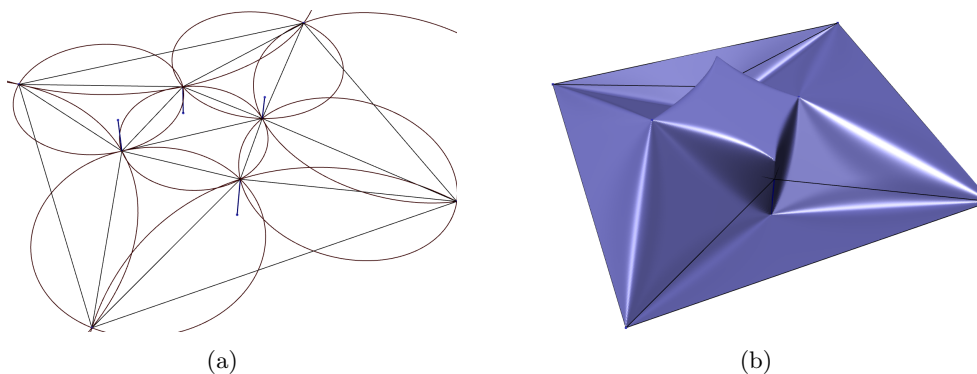


Figure 3.12: (a) Degenerate setting with four collinear points in the middle. (b) Interpolant based on Brown coordinates; the discontinuities are clearly visible as steps in the heightfield.

neighborhood of  $\mathbf{x}_1$  in which  $\psi \equiv 0$ . This assumption is true only if the  $\mathbf{x}_1, \dots, \mathbf{x}_4$  are not co-circular. Otherwise,  $\psi_1 \equiv \psi_2$  and the proof fails, supporting our observation.

These artifacts can already be observed for nearly-identical circumcircles, which is a drawback that Brown failed to mention in his paper.

Brown also showed that, dropping the constraint of strict positivity inside the circumcircles, his method produces Sibson coordinates if the blending functions  $\psi_i$  are computed based on Watson's construction. If  $\mathbf{x}_{i1}, \mathbf{x}_{i2}, \mathbf{x}_{i3}$  are the counterclockwise oriented vertices of  $T_i$ , then  $\nabla(\mathbf{x}; T_i)$  denotes the dual triangle of  $\Delta(\mathbf{x}_{i1}, \mathbf{x}_{i2}, \mathbf{x}_{i3})$  with respect to  $\mathbf{x}$  as defined in Section 3.2.5.4. Now, the choice

$$\psi_i^{\text{SIB}}(\mathbf{x}) := \text{signed area of } \nabla(\mathbf{x}; \mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c) \quad (3.7)$$

yields Sibson coordinates  $\lambda^{\text{BRO}} = \lambda^{\text{SIB}}$ . As Brown pointed out, the signed area of the dual triangle is infinite if  $\mathbf{x}$  lies on any edge of  $T_i$ . Although the coordinates can be shown to be continuous, this brings up numerical issues, especially in the vicinity of the vertices.

Because of equivalence of Watson's construction (cf. Section 3.2.5.4) with Sibson coordinates for  $\psi^{\text{SIB}}$ , Brown's method is able to produce convex natural neighbor coordinates, which suggests that other constructions for convex local coordinates are possible if the constraint of strict positivity inside the circumcircles is dropped.

Brown's method provides local coordinates with respect to the set of natural neighbors inside the convex hull, and reproduces Sibson coordinates for a certain choice of blending functions. It can be seen as a generalized construction for possibly non-convex natural neighbor coordinates. However, the coordinates do in general not change continuously with the coordinates of the points in  $\mathbf{X}$ , which becomes apparent if a flip operation in the Delaunay triangulation is considered.

A big advantage of Brown's approach is the straightforward extension past the convex hull of the data set, which is discussed further in Section 8.6.1.

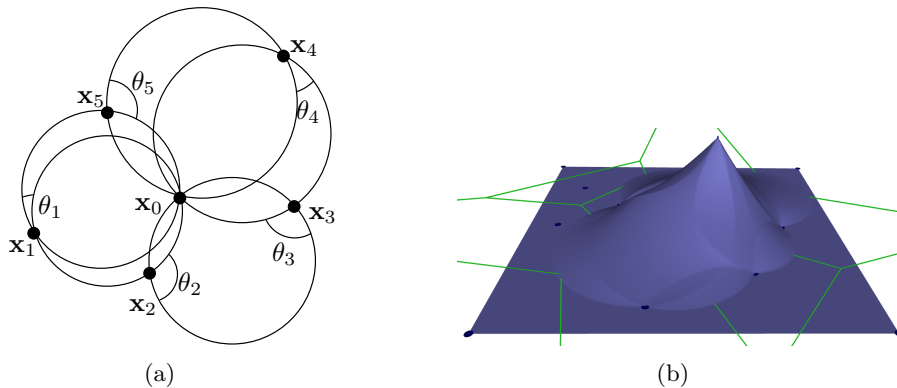


Figure 3.13: Delaunay circumcircles passing through  $\mathbf{x}_0$  and its natural neighbors. The angles at the lunes are indicated.

### 3.2.5.7 Möbius Invariant Natural Neighbor Coefficients in 2D

A Möbius transformation is a one of the form  $z \mapsto (az + b)/(cz + d)$ , with  $a, b, c, d, z \in \mathbb{C}$ , with the major property that it maps circles to circles, and preserves angles. In [BE03], Bern and Eppstein proposed a construction of coefficients  $\lambda$  that do not comprise barycentric coordinates but are invariant under Möbius transformations. They replace area-derived coefficients by ones proportional to angles. Since angles are invariant under Möbius transformations, so are the derived coefficients.

Let  $\theta_i$  denote the exterior angle formed at the *lune* of the two Delaunay circumcircles passing through  $\mathbf{x}_0$  and  $\mathbf{x}_i$ , see Figure 3.13(a). The Möbius invariant natural neighbor coefficients are defined as

$$\lambda_i^{\text{MÖB}} = \hat{\lambda}_i^{\text{MÖB}} / \sum_j \hat{\lambda}_j^{\text{MÖB}}, \quad \hat{\lambda}_i^{\text{MÖB}} = \tan\left(\frac{\theta_i}{2}\right).$$

Considering the complex plane  $\mathbb{C}$  as the domain for scattered data interpolation yields the visualization in Figure 3.13(b). There is a strong visual similarity of  $f^{\text{MÖB}}$  and  $f^{\text{LAP}}$ .

## 3.2.6 Transfinite Natural Neighbor Coordinates

In this section we discuss methods to interpolate line segments, polygons and circular arcs instead of points, commonly known as *transfinite interpolation*. The problem of transfinite interpolation based on natural neighbor coordinates has been addressed by Anton et al. for Sibson coordinates with respect to points and line segments in [AMG98], by Gross et al. with respect to circles and polygons [GF99], and by Hiyoshi et al. for Laplace coordinates with respect to points, line segments, and circles [HS00a]. An interesting opportunity in transfinite interpolation is to deliberately impose discontinuities along the manifold data sites by using different values for each side.

First, we explain the main differences between ordinary Voronoi diagrams and such with lines and curves as data sites, before taking a closer look at how the identities expressed by local coordinates in Section 3.2.5 extend to the transfinite case.

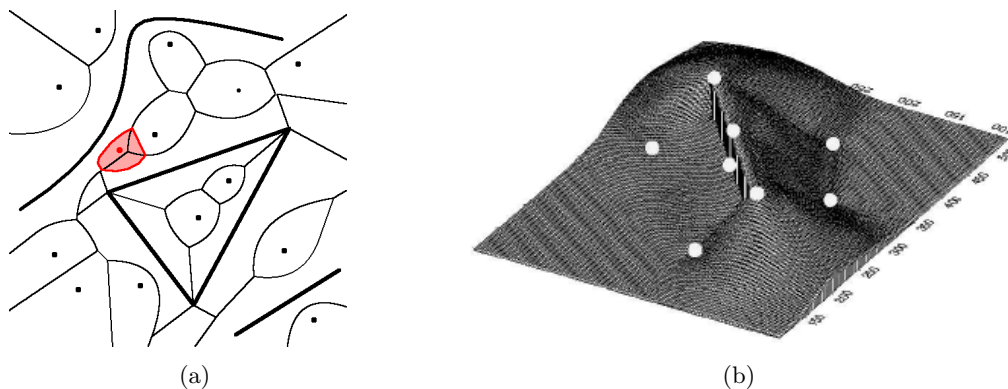


Figure 3.14: (a) Voronoi diagram of a set of points, line segments and general curves (drawn bold). The virtual tile of a new point is shaded. Picture courtesy of [Hof99]. (b) Transfinite interpolation of a directed line segments and several points. Picture courtesy of [AMG04].

### 3.2.6.1 Generalized Voronoi Diagrams in 2D

Non-point data sites lead to *generalized Voronoi diagrams*, and the geometric primitives that constitute the local coordinates from Section 3.2.5 are no longer convex polygons. The main consequence of this generalization is an increased complexity in both data handling and the computation of the interpolant, which also seems to be the reason that research in this direction has been restricted to two dimensions so far.

Consider a non-intersecting curve  $C \subset \mathbb{R}^2$  as an additional data site besides the point-shaped data sites  $\mathbf{x}_i \in \mathbf{X} \subset \mathbb{R}^2$ . The definition of generalized Voronoi diagrams in Section 2.3.5 still holds with a modified distance function,

$$d(\mathbf{x}, C) = \min_{\mathbf{y} \in C} \|\mathbf{y} - \mathbf{x}\|.$$

Tiles induced by points are still convex, while for curves this is in general not true. As in the ordinary Voronoi diagram, the query point  $\mathbf{x}_0$  has a convex tile  $\mathcal{T}_0$ . An example of a generalized Voronoi diagram and the virtual tile can be seen in Figure 3.14(a). The shape of the bisectors between the various elements of the Voronoi diagram is at least as complicated as that of the elements itself. Thus, an exact computation of areas and lengths seems feasible only for simple shapes of the data sites. For arbitrary shapes, the Voronoi diagram can be approximated using graphics hardware, see Section 3.2.9.3.

### 3.2.6.2 Interpolating Data on Line Segments

If the data sites are line segments, there are bisectors between lines, between points, and between points and lines, where endpoints of line segments also count as points. The bisectors are parabolic arcs between the interior of a line segment and a point, while all other bisectors remain linear. In practice, the endpoints of a line segment are treated as separate data sites, which leads to a partition of its Voronoi tile into tiles for its directed

half edges and its end points, as shown in Figure 3.14(a).

Local coordinates in the transfinite setting can be derived from the discrete setting by considering natural neighbor coordinates for a set of point data sites, say  $N(\mathbf{x}_0)$ , and a line segment data site  $C(t)$ ,  $t \in [0, 1]$ , approximated at a sequence of  $m + 1$  knots  $t_j = j/m$ ,  $j = 0, \dots, m$  by points  $\mathbf{c}_0 = C(t_0), \dots, \mathbf{c}_m = C(t_m)$ . If  $\lambda_i$  is the natural neighbor coordinate of  $\mathbf{x}_0$  with respect to  $\mathbf{x}_i$  and  $\mu_i$  is the natural neighbor coordinate of  $\mathbf{x}_0$  with respect to  $\mathbf{c}_i$ , then

$$\mathbf{x}_0 = \sum_{i \in N_0} \lambda_i \mathbf{x}_i + \sum_{j=1, \dots, m} \mu_j \mathbf{c}_j. \quad (3.8)$$

When the sequence is refined such that  $m \rightarrow \infty$ , it follows that  $\mu_i \rightarrow \mu(t_i)$  for any  $t_i \in [0, 1]$ , and

$$\mathbf{x}_0 = \sum_{i \in N_0} \lambda_i \mathbf{x}_i + \int_{t \in [0, 1]} \mu(t) C(t) dt, \quad (3.9)$$

If  $z(t)$  describes the Data distributed over  $C(t)$  for  $t \in [0, 1]$ , an interpolant is given by

$$f(\mathbf{x}_0) = \sum_{i \in N_0} \lambda_i z_i + \int_{t \in [0, 1]} \mu(t) z(t) dt. \quad (3.10)$$

In [GF99], interpolation of arbitrary functions over polygons is solved. If  $\mathbf{c}_i(t)$  is a line segment of the polygon, parameterized by  $t \in [0, 1]$ , each subtile  $\mathcal{T}_0 \cap \mathcal{T}_i^{(\mathbf{x}_0)}$  can be interpreted to have a certain thickness above  $\mathbf{c}_i$ , which is nonzero only where the subtile projects to  $\mathbf{c}_i$ . The  $\lambda_i(\mathbf{x}_0; t)$  are taken to be this thickness, normalized by the overall area, and define a meaningful density for the accumulation of data values. The application of this interpolant to the data distributed along the non-convex polygon in Figure 3.15(a) is shown in Figure 3.15(b).

In [AMG98, AMG04], the same approach has been implemented for arbitrary arrangements of non-intersecting line segments and points. By allowing different values on both sides of the line segments, they are able to faithfully model discontinuities as they arise in, e.g., geology. See Figure 3.14(b) for an example. Although they restrict their approach to linear data distributions along the lines, the approach of [GF99] can also be applied to interpolate to arbitrary scalar functions over the sites. While the last two approaches focus on a generalization of Sibson's coordinates, [HS00a] generalizes Laplace interpolation to arrangements of multiple classes of curves. The main difference lies in defining the coordinates of  $\mathbf{x}_0$  by a density function over the bisector bounding  $\mathcal{T}_0$ . The result of this interpolant applied to an arrangement of points, line segments, and circles is shown in Figure 3.15(c).

### 3.2.6.3 Interpolating Data on Circles, Lines and Points

In case the input consists of data distributed over a circle,  $\mathcal{T}_0$  is an ellipse. As a result, Sibson's transfinite interpolant takes on a simple form. Let  $c$  be the unit circle centered at  $\mathbf{0}$ ,  $z_1(\Theta)$  the data, parameterized over  $\Theta \in [0, 2\pi)$ , and  $\mathbf{x}_0 = (\rho, \theta)$  be expressed in

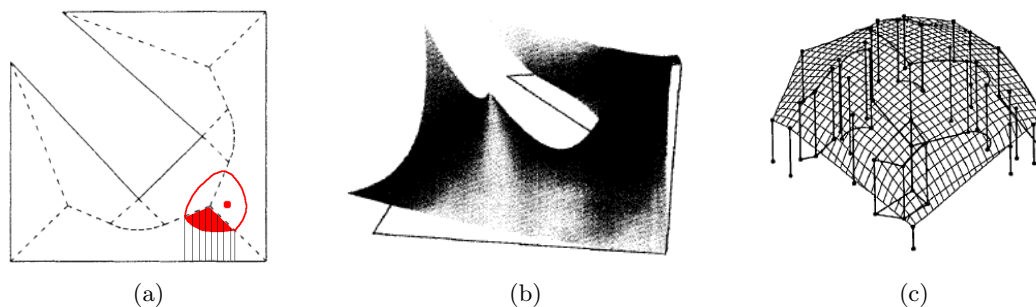


Figure 3.15: Transfinite interpolation of curves. (a) The Voronoi diagram of a polygon, the contribution of the lower subtile depicted by the thin lines. (b) Transfinite interpolation of the boundary values. (c) Transfinite interpolation of a collection of points, line segments and circular arcs. Pictures (a), (b) courtesy of [GF99], (c) courtesy of [HS00a].

polar coordinates. Then in [GF99] a Sibson’s transfinite interpolant on circles is defined as

$$f(\mathbf{x}_0) = \frac{(1 - \rho^2)^{3/2}}{2\pi} \int_0^{2\pi} \frac{z_1(\Theta)}{(\rho \cos(\theta - \Theta) - 1)^2} d\Theta \quad \begin{cases} 0 \leq \rho < 1 \\ 0 \leq \theta \leq 2\pi. \end{cases}$$

Based on a similar idea, [HS00a] formulated an identity and the resulting interpolant for Laplace coordinates.

### 3.2.7 Smooth Natural Neighbor Interpolation

The previous section considered interpolants building on a linear combination of data values by local coordinates as in (3.3), resulting in derivative discontinuities at  $\mathbf{x}_i$  which are inherited from the local coordinates.

The interpolation of smooth functions requires additional efforts, and there are two distinct approaches to this: One is to construct some polynomial in the local coordinates that interpolates derivatives at the data sites [Sib81, Far90, HS04]. The other constructs non-convex coordinates from a larger natural neighborhood as explained in [Cla96, Flö03b], although our observations indicate that this approach still is only  $C^0$  at the data sites.

To apply the first approach, the derivatives at the data sites need to be known. This is the subject of Chapter 6, where we show how derivatives can be estimated if they are not provided.

#### 3.2.7.1 Sibson’s $C^1$ Interpolant

In [Sib81] Sibson described the construction of a  $C^1$  interpolant. He estimates a gradient  $\nabla_i$  for each Voronoi site  $\mathbf{x}_i$  from the weighted least squares plane through the neighboring data  $\{(\mathbf{x}_i, z_i)\}_{i \in N_0}$ , which are then interpolated by blending first order functions with the help of coordinates  $\lambda_i^{\text{SIB}}$ .

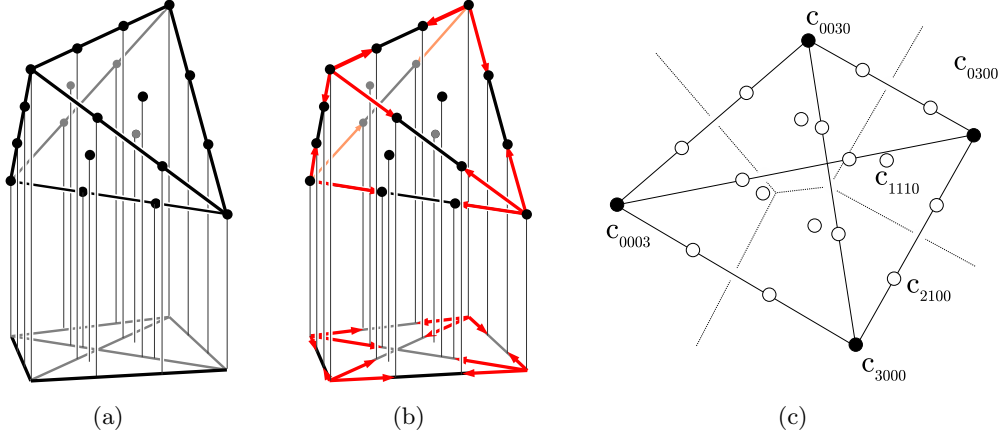


Figure 3.16: (a) Relation between a four-variate, cubic Bézier simplex in  $\mathbb{R}^3$  and the projection of its domain to  $\mathbb{R}^2$ . (b) The red arrows indicate how the derivatives are used to define the control points. (c) Planar projection of the control net of a cubic Bézier simplex in  $\mathbb{R}^3$ .

With  $r_i := d(\mathbf{x}_0, \mathbf{x}_i)$ ,  $\gamma_i := \lambda_i^{\text{SIB}}/r_i$ , define

$$\begin{aligned} \zeta_i &:= z_i + (\mathbf{x}_0 - \mathbf{x}_i)^T \nabla_i & \text{and} & \quad \zeta := \left( \sum_{i \in N_0} \gamma_i \zeta_i \right) / \left( \sum_{i \in N_0} \gamma_i \right), \\ \alpha &:= \left( \sum_{i \in N_0} \lambda_i^{\text{SIB}} r_i \right) / \left( \sum_{i \in N_0} \gamma_i \right) & \text{and} & \quad \beta := \sum_{i \in N_0} \lambda_i^{\text{SIB}} r_i^2. \end{aligned}$$

Blending  $\zeta$  and Sibson's  $C^0$  interpolant  $f^{\text{SIB}0}(\mathbf{x}_0)$  yields Sibson's  $C^1$  interpolant

$$f^{\text{SIB}1}(\mathbf{x}_0) = \frac{\alpha f^{\text{SIB}0}(\mathbf{x}_0) + \beta \zeta}{\alpha + \beta}.$$

The interpolant  $f^{\text{SIB}1}$  exactly reproduces functions of the form  $\mathbf{x} \mapsto \mu(\mathbf{x} - \mathbf{a})^T(\mathbf{x} - \mathbf{a})$  for  $\mu \in \mathbb{R}$  and  $\mathbf{a} \in \mathbb{R}^n$ , i.e., spherical quadratics. There is no obvious generalization of this approach to higher orders of continuity.

### 3.2.7.2 Farin's $C^1$ Interpolant

A much more general approach which is not restricted to natural neighbor coordinates but can be applied to all convex, local coordinates was proposed by Farin in [Far90]. Any set of generalized barycentric coordinates  $\boldsymbol{\lambda} \in \mathbb{R}^l$  can be seen as barycentric coordinates in an  $l$ -variate Bézier simplex. We can without loss of generality assume that the domain of this simplex projects to  $\mathbb{R}^n$  as the convex hull of  $N(\mathbf{x}_0)$ , where  $l = |N_0|$ . This is illustrated for  $n = 2$  in Figure 3.16. In Bézier simplices it is easy to model directional derivatives at the vertices  $\mathbf{x}_i$  by appropriately choosing the Bézier control net according to the relation between control points and derivatives given in (2.5). Prescribed derivatives at  $\mathbf{x}_i$  fix a certain number of control points, which is illustrated in Figure 3.16(b) for the cubic case. The remaining control points can be chosen arbitrarily without interfering with the interpolation property, and the concept of degree elevation allows to achieve polynomial precision.

## Related Work

Farin presented the implementation of the above idea for cubic Bézier simplices over  $\lambda^{\text{SIB}}$  to interpolate gradients  $\nabla_i$ , yielding a globally  $C^1$ -continuous interpolant, which we revisit again below.

Without loss of generality we assume  $N(\mathbf{x}_0) = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ . Following the notation introduced in Section 2.5.4, we denote by  $D_{ij}$  the directional derivative along  $\mathbf{d}_{ij} := \mathbf{x}_j - \mathbf{x}_i$ , by  $\mathbf{i} \in \mathbb{N}^m$ ,  $i_1 + \dots + i_m = d$  the  $n$ -dimensional multi-index that enumerates the  $d$ -th degree Bézier control points  $(\xi_{\mathbf{i}}, c_{\mathbf{i}})$ , by  $\mathbf{e}_j$  the  $j$ -th canonical unit-index, and by  $d\mathbf{e}_i$  the index of vertex  $\mathbf{x}_i$  of the Bézier simplex. Since we consider functional Bézier simplices, the control point abscissae  $\xi_{\mathbf{i}} = (i_1 \mathbf{x}_1 + \dots + i_m \mathbf{x}_m)/m$  are defined by the data site constellation, and focus is on control point ordinates  $c_{\mathbf{i}}$ . See Figure 3.16(c) for an example.

Control points at the vertices are fixed by the interpolation constraint  $c_i = z_i$ . The directional derivative at  $\mathbf{x}_i$  towards  $\mathbf{x}_j$ , given by

$$D_{ij} = 3(c_{ij} - c_i) = \nabla_i^T \mathbf{d}_{ij}, \quad j \in N_0 \setminus \{i\},$$

constrains all inner control points  $c_{ij}$  to be coplanar,

$$c_{ij} = z_i + \frac{1}{3} \nabla_i^T \mathbf{d}_{ij} \quad \text{for } i \neq j.$$

This fixes all control points except one on each simplex face.

By degree elevation for Bézier simplices, these are chosen to ensure quadratic precision of the resulting interpolant, see [Far90, Flö03b]. In particular, the construction of under-determined control points is found by examining Bézier simplices of degree two, one for each vertex  $\mathbf{x}_i$ . All inner control points of such a simplex are determined solely by the value and edge derivatives at  $\mathbf{x}_i$ . If the values and derivatives come from a quadratic function, all these quadratic simplices agree. The degree-elevation formula expresses the cubic control points as linear combinations of the quadratic control points, where by rearranging equations every under-determined cubic control point is expressed as a linear combination of fixed cubic control points. By averaging all these equations we get a symmetric construction of the under-determined control point that furthermore guarantees quadratic precision. Let  $\mathbf{k} = \mathbf{e}_i + \mathbf{e}_j + \mathbf{e}_k$  for  $i < j < k$ , then  $c_{\mathbf{k}}$  is an inner control point. To set it such that  $b^3(\boldsymbol{\lambda})$  has quadratic precision, set

$$u_{\mathbf{k}} = \frac{1}{3}(c_i + c_j + c_k) \quad \text{and} \quad v_{\mathbf{k}} = \frac{1}{6}(c_{ij} + c_{ik} + c_{ji} + c_{jk} + c_{ki} + c_{kj}),$$

i.e., the average of the remaining fixed control points, then  $c_{\mathbf{k}} = \frac{3}{2}v_{\mathbf{k}} - \frac{1}{2}u_{\mathbf{k}}$  yields quadratic precision for the interpolant. The resulting interpolant inherits  $C^1$  continuity on  $\mathcal{C} \setminus \mathbf{X}$  from  $\lambda^{\text{SIB}}$  and is given by

$$f^{\text{FAR}}(\mathbf{x}_0) := b^3(\boldsymbol{\lambda}^{\text{SIB}}(\mathbf{x}_0)).$$

Figure 3.17(c) shows the interpolant in a setting exposing the polynomial part.



### 3.2.7.3 Hiyoshi's $C^2$ Interpolant

Applying the above approach to quintic Bézier simplices over  $\lambda^{\text{HY}2}$ , [HS04] present a construction of control points that matches derivatives up to order two given by the gradient  $\nabla_i$  and the Hessian  $\mathcal{H}_i$  at data site  $\mathbf{x}_i$ . The notation used in the following was introduced in Section 2.5.4.

Let  $i, j, k, l, m$  be mutually distinct,  $\mathbf{d}_{ij} = \mathbf{x}_j - \mathbf{x}_i$ ,  $z_{ij} := \nabla_i^T \mathbf{d}_{ij}$ , and  $z_{ijk} := \mathbf{d}_{ij}^T \mathcal{H}_i \mathbf{d}_{ik}$ , then the control points as explicitly described by Hiyoshi are

$$\begin{aligned}
c_i &= z_i, \\
c_{ij} &= z_i + \frac{1}{5}z_{ij}, \\
c_{ijk} &= z_i + \frac{1}{5}(z_{ij} + z_{ik}) + \frac{1}{20}z_{ijk}, \\
c_{ijjk} &= \frac{1}{2}(z_i + z_j) + \frac{3}{20}(z_{ij} + z_{ji}) + \frac{1}{10}(z_{ik} + z_{jk}) + \frac{1}{30}(z_{ijk} + z_{jik}) + \frac{1}{120}(z_{ijj} + z_{jii}), \\
c_{ijkl} &= \frac{7}{10}(z_i + z_j + z_k + z_l) + \frac{11}{90}(z_{ij} + z_{ik} + z_{il}) + \frac{1}{45}(z_{ijk} + z_{ijl} + z_{ikl}) + \\
&\quad \frac{1}{45}(z_{jil} + z_{jki} + z_{jkl} + z_{kij} + z_{kil} + z_{kjl} + z_{lij} + z_{lik} + z_{ljk}) + \\
&\quad \frac{1}{180}(z_{jik} + z_{jil} + z_{jkl} + z_{kij} + z_{kil} + z_{kjl} + z_{lij} + z_{lik} + z_{ljk}), \\
c_{ijklm} &= \frac{1}{5}(z_i + z_j + z_k + z_l + z_m) + \\
&\quad \frac{1}{30}(z_{ij} + z_{ik} + z_{il} + z_{im} + z_{ji} + z_{jk} + z_{jl} + z_{jm} + z_{ki} + z_{kj} + \\
&\quad \quad z_{kl} + z_{km} + z_{li} + z_{lj} + z_{lk} + z_{lm} + z_{mi} + z_{mj} + z_{mk} + z_{ml}) + \\
&\quad \frac{1}{180}(z_{ijk} + z_{ijl} + z_{ijm} + z_{ikl} + z_{ikm} + z_{ilm} + \\
&\quad \quad z_{jil} + z_{jik} + z_{iim} + z_{jkl} + z_{jkm} + z_{jlm} + \\
&\quad \quad z_{kij} + z_{kil} + z_{kim} + z_{kjl} + z_{kjm} + z_{klm} + \\
&\quad \quad z_{lij} + z_{lik} + z_{lim} + z_{ljk} + z_{ljm} + z_{lkm} + \\
&\quad \quad z_{mij} + z_{mik} + z_{mil} + z_{mjk} + z_{mjl} + z_{mkl}).
\end{aligned} \tag{3.11}$$

A visualization of  $f^{\text{HY}2}$  in a setting exposing the polynomial part is given in Figure 3.17(d).

**Remark 3.5** *As mentioned in the original paper, only  $c_i$ ,  $c_{ij}$ , and  $c_{ijk}$  are fixed by gradients and Hessians. The remaining points are obviously chosen based on the degree elevation approach to achieve cubic precision. However, the choice of coefficients is not unique. Motivated by the opaqueness of the explicitly stated formulas we tried to reproduce Hiyoshi's results in a structured manner. Although we were able to construct cubic precision control points leading to smooth surfaces, none of our constructions produced as fair surfaces as Hiyoshi's choice of control points.*

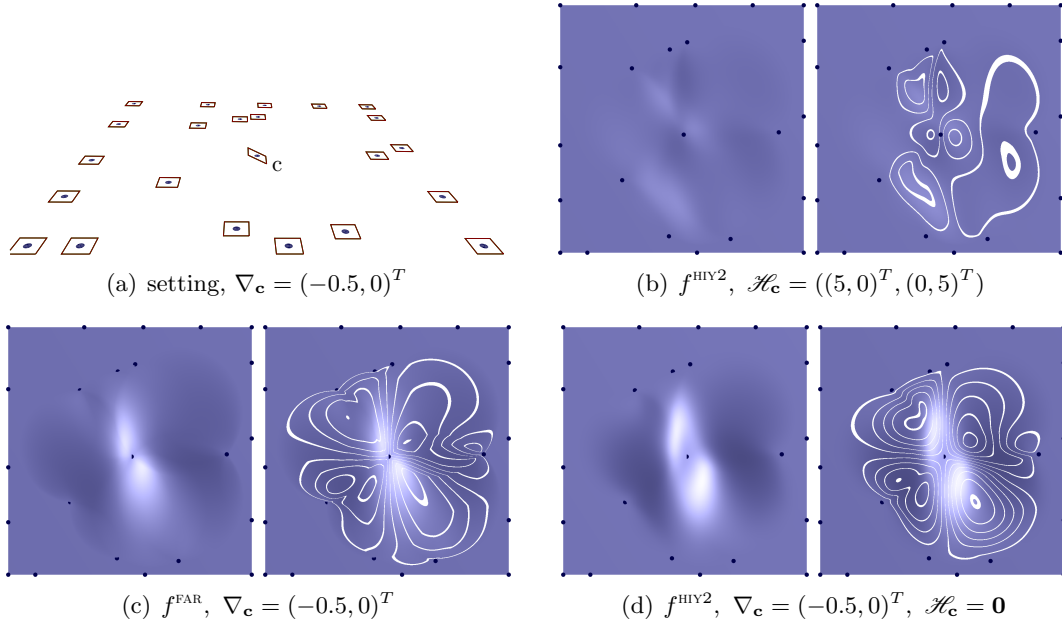


Figure 3.17: A setting for a Hermite-type basis function: the sites are chosen as in Figure 3.6, but the values are all zero. (a) Setting for (c) and (d). (b) Hiyoshi's interpolant for zero data except a parabolic function assigned to the center site. (c) Farin's interpolant  $f^{\text{FAR}}$ . (d) Hiyoshi's interpolant  $f^{\text{HIY}2}$ . Cusps in the reflection lines indicate where Farin's interpolant inherits the  $C^2$  discontinuities from the Sibson coordinates  $\lambda^{\text{SIB}}$ . The smooth shape of reflection lines for  $f^{\text{HIY}2}$  verifies the  $C^2$  smoothness of  $\lambda^{\text{HIY}}$ .

### 3.2.7.4 Clarkson's Interpolation

One special kind of local coordinates that are not convex is based on an idea of Clarkson [Cla96] and has been investigated and implemented in [Flö03b]. In the two-ring neighborhood of the query position, local coordinates are constructed specifically to reproduce spherical quadratics, i.e., functions of the form  $\mathbf{x} \mapsto a\|\mathbf{x} - \mathbf{b}\|^2$ ,  $a \in \mathbb{R}$ ,  $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$ . It is the only approach so far that aims at an implicit  $C^1$  construction and does not depend on prescribed derivative information. Clarkson's local coordinates differ significantly from those of Section 3.2.5, in that they

- depend on  $\bigcup_{i \in N_0} N_i$ , i.e., the two-ring neighborhood of  $\mathbf{x}_0$ ,
- are not convex,
- are conjectured in [Flö03b] to be  $C^1$  at  $\mathbf{x}_i$ .

The following briefly repeats the definition of Clarkson coordinates from [Flö03b] without the derivation.

Let  $\mathbf{X}^{+(i)} := \mathbf{X} \cup \{\mathbf{x}_0\} \setminus \{\mathbf{x}_i\}$ . Let  $I_1(\mathbf{x}_0)$  denote the indices of natural neighbors of  $\mathbf{x}_0$  in  $\mathbf{X}$ ,  $J_i(\mathbf{x}_0)$  the indices of natural neighbors of  $\mathbf{x}_i$  in  $\mathbf{X}^{+(i)}$  other than  $\mathbf{x}_0$ , and  $I_2(\mathbf{x}_0) := \bigcup_{i \in I_1(\mathbf{x}_0)} J_1(\mathbf{x}_i)$  the two-ring natural neighborhood of  $\mathbf{x}_0$  in  $\mathbf{X}$ . Let  $\pi_x^+(\mathbf{x}_i)$  and  $\pi_j^+(\mathbf{x}_i)$  be as shown in Figure 3.18, and  $\pi^+(\mathbf{x}_i)$  the volume of the virtual Voronoi tile of  $\mathbf{x}_i$  in the

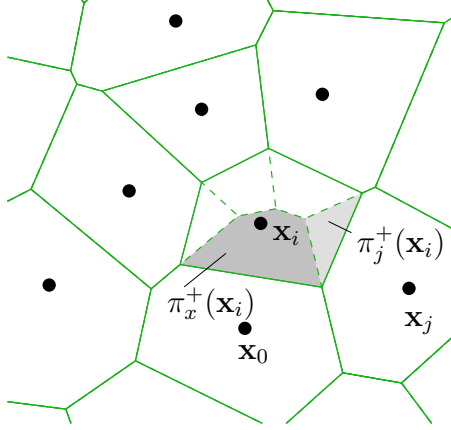


Figure 3.18: Notation used in the definition of Clarkson coordinates. The areas  $\pi_j^+(\mathbf{x}_i)$  and  $\pi_x^+(\mathbf{x}_i)$  are derived from the virtual tile of  $\mathbf{x}_i$  in  $\mathbf{X} \cup \{\mathbf{x}_0\} \setminus \{\mathbf{x}_i\}$ .

Voronoi diagram of  $\mathbf{X}^{+(i)}$ . Now,

$$\lambda_x^+(\mathbf{x}_i) := \pi_x^+(\mathbf{x}_i)/\pi^+(\mathbf{x}_i) \quad \text{and} \quad \lambda_j^+(\mathbf{x}_i) := \pi_j^+(\mathbf{x}_i)/\pi^+(\mathbf{x}_i)$$

are the natural neighbor coordinate of  $\mathbf{x}_i$  with respect to  $\mathbf{x}_0$  in  $\mathbf{X}^{+(i)}$ , and the natural neighbor coordinate of  $\mathbf{x}_i$  with respect to  $\mathbf{x}_j$  in the same set of points. The intermediate variables

$$e_x = \sum_{i \in I_1(\mathbf{x}_0)} \lambda_i(\mathbf{x}_0) \mathbf{x}_i^2 - \mathbf{x}_0^2,$$

$$e_i = \mathbf{x}_0^2 - \frac{1}{\lambda_x^+(\mathbf{x}_i)} (\mathbf{x}_i^2 - \sum_{j \in J_1(\mathbf{x}_i)} \lambda_j^+(\mathbf{x}_i) \mathbf{x}_j^2)$$

are then used to define local, non-convex coordinates of  $\mathbf{x}_0$  with respect to the one-ring neighborhood of each of its natural neighbors  $\mathbf{x}_i \in N(\mathbf{x}_0)$ ,

$$\gamma_i^i(\mathbf{x}_0) = \frac{e_i}{e_i + e_x} \lambda_i(\mathbf{x}_0) + \frac{e_x}{e_i + e_x} \frac{1}{\lambda_x^+(\mathbf{x}_i)},$$

$$\gamma_j^i(\mathbf{x}_0) = \frac{e_i}{e_i + e_x} \lambda_j(\mathbf{x}_0) - \frac{e_x}{e_i + e_x} \frac{\lambda_j^+(\mathbf{x}_i)}{\lambda_x^+(\mathbf{x}_i)}, \quad i \neq j.$$

Finally, this leads to Clarkson coordinates for  $\mathbf{x}_0$  with respect to its two-ring neighbors  $\mathbf{x}_j$ ,  $j \in I_2(\mathbf{x}_0)$ ,

$$\lambda_j^{\text{CLA}} = \sum_{i \in I_1(\mathbf{x}_0)} \lambda_i(\mathbf{x}_0) \gamma_j^i(\mathbf{x}_0).$$

In addition to the local coordinate property, Clarkson coordinates satisfy

$$\sum_{i \in I_1(\mathbf{x}_0) j \in I_2(\mathbf{x}_0)} \lambda_j^{\text{CLA}} \mathbf{x}_j^T \mathbf{x}_j = \mathbf{x}_0^T \mathbf{x}_0,$$

which corresponds to the reproduction of spherical quadratics if they are used for scattered data interpolation.

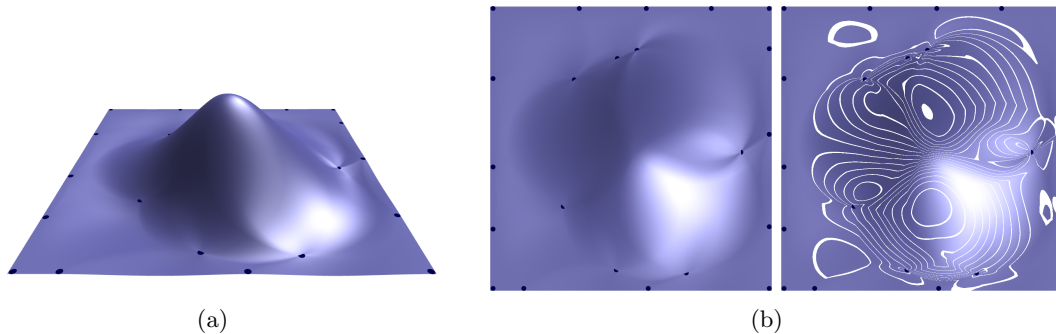


Figure 3.19: Basis function of the scattered data interpolant derived from Clarkson coordinates, the data chosen as in Figure 3.6(a) with additional zero data sites outside (not displayed) to provide big enough neighborhoods. (a) Perspective view. (b) The point setting is chosen such that an adequate region inside the convex hull has a full two-ring neighborhood, which is required to compute Clarkson coordinates.

We show the basis function for the interpolant derived from Clarkson’s coordinates in Figure 3.19.

### 3.2.8 Manifold Natural Neighbor Interpolation

The Voronoi diagram is defined by a set of points and a distance measure. For points on a manifold, this definition still holds, at the expense of potentially non-convex tiles due to a non-Euclidean metric, see [LL00]. The manifold setting results in bisectors of arbitrary complexity and computing areas (volumes) becomes tedious for non-trivial geometries. To our knowledge, there has been no work carried out on natural neighbor based interpolation on continuous manifolds.

In [BC00], however, it is shown that if the manifold has a sufficiently dense sampling, a less complicated approach is possible. The data sites on the manifold induce a Voronoi diagram in the embedding space  $\mathbb{R}^n$  of the manifold. The intersection of that Voronoi diagram and the manifold gives a partition of the manifold that locally converges to the Euclidean Voronoi diagram when the sampling density goes to infinity. Furthermore, the main result in [BC00] states that Sibson’s identity holds for an infinitely dense sampling of the surface.

Based on this work, natural neighbor based interpolation on point clouds issued from manifolds is developed in [Flö03b, BF04]. As a main result, a point on a manifold can be expressed in local coordinates in the tangent plane at that point, given the manifold is sampled densely enough. The intersection of the three-dimensional Voronoi diagram of the data sites with the tangent plane defined by the normal vector at the query position produces a power diagram in the tangent plane. [Flö03b] proves Sibson’s identity for power diagrams and develops natural neighbor coordinates for point clouds.

### 3.2.9 Implementation of Natural Neighbor Interpolation

Natural neighbor based interpolants are based on an underlying identity that provides generalized barycentric coordinates in the natural neighbors. The definition of those local coordinates is motivated geometrically on the Voronoi diagram of the input data sites. The computation, however, can often be carried out more elegantly and also more stably. These approaches can be classified as geometric, algebraic and approximate. For simple settings, the geometric approach is still feasible. For higher dimensions, higher orders of continuity and more complex input data sites, algebraic and approximate approaches yield more efficient and more stable solutions.

In the rest of this section we describe the computation of natural neighbor coordinates, since they are the main building block for all interpolants in this survey. The implementation of the  $C^1$  and  $C^2$  constructions at the data sites from Section 3.2.7 for point-shaped data sites is straightforward.

#### 3.2.9.1 Geometric Computation

The dual of the Voronoi diagram is the Delaunay tessellation, as introduced in Section 2.3.5. Therefore, evaluation and traversal of the Voronoi diagram of a set of points can be carried out on its Delaunay tessellation with corresponding adjacency information.

Laplace and Sibson coordinates relate to areas and volumes of intersections of Voronoi tiles which are easily computed in two dimensions, and implementations are known for three dimensions as well [Owe93, CGA08]. In case of line segment shaped data sites, the constrained Delaunay tessellation can be used. Input data sites of arbitrary shape are difficult to handle in classical geometric data structures and usually require more intricate representations of the Voronoi diagram. The common solution to this is a piecewise linear approximation of the input curves, which then act as constrained edges in the constrained Delaunay tessellation.

In three or more dimensions, the data structures required for storing the Delaunay tessellation and its adjacency graph become very complex, and traversing the topological neighborhood becomes error prone and cumbersome.

#### 3.2.9.2 Algebraic Computation

For algebraic computation the explicit construction of the Voronoi diagram is avoided, and computation is carried out on simpler geometric structures.

To compute Laplace coordinates in the two-dimensional setting, the calculation presented by Sugihara only assumes the natural neighbors of the query position to be given in counterclockwise order [Sug99]. The resulting identity also holds in the more general case of star-shaped neighborhoods, making this approach robust against topological inconsistencies as they appear from numerical noise. Considering the equivalence of Laplace coordinates and cotangent coordinates in Delaunay triangulations, it appears that this is closely related to the computation of cotangent coordinates.

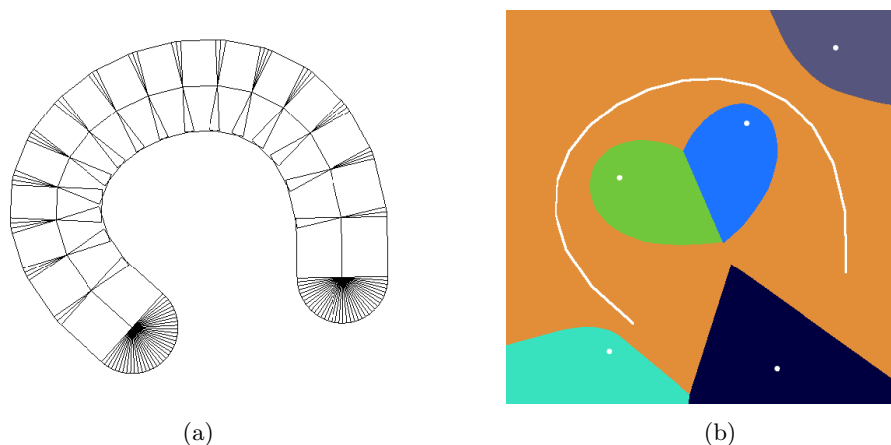


Figure 3.20: (a) Polygonal approximation of a distance function. (b) Generalized Voronoi diagram computed on graphics hardware. Pictures courtesy of [HCK<sup>+</sup>99].

Watson’s construction for the computation of Sibson’s coordinates as introduced in Section 3.2.5.4 is solely based on the computation of circumcenters and signed triangle areas. Since this method operates solely on simplices, it has a straightforward generalization to higher dimensions and is successfully implemented in 3D in the CGAL library [CGA08]. For positions located on the facets of Delaunay simplices, the *dual triangle* becomes degenerate, but by a limit argument the construction can be shown to be valid. However, in a straightforward implementations, the numerical instabilities near the facets of Delaunay simplices become apparent.

Building on Watson’s signed triangle decomposition of Voronoi tiles, Hiyoshi proposed in [Hiy05] a way to stably compute his coordinates of order  $k$  in  $\mathbb{R}^2$  by first encoding the construction of Voronoi entities into algebraic expressions in Delaunay triangles, which are then rearranged to circumvent numerical instabilities based on zero denominators as they might appear in the equations in Section 3.2.5.

A straightforward computation of Laplace and Sibson coordinates in any dimension exists once the one-ring Delaunay neighborhood is known. The content of the corresponding tile facets and tile intersections can be expressed as an intersection of half-spaces that are defined entirely by the query position and its Delaunay neighbors, as Braun and Sambridge mentioned in [BS95]. Thus, the computation of Laplace and Sibson coordinates reduces to the determination of Delaunay neighbors, described by Watson in [Wat81], and volume computation in  $n$  dimensions, which has been thoroughly analyzed by Bueler et al. in [BEF00]. We applied this approach to derive a construction of Hiyoshi coordinates in  $\mathbb{R}^2$ , which is described in detail in Chapter 5. Note that the average number of Delaunay neighbors grows exponentially with dimension, and so does the complexity of volume computations.

### 3.2.9.3 Approximate Computation

By allowing a small error for the local coordinates, an approximate formulation of natural neighbor coordinates can be given based on a discretization of the Voronoi diagram. The

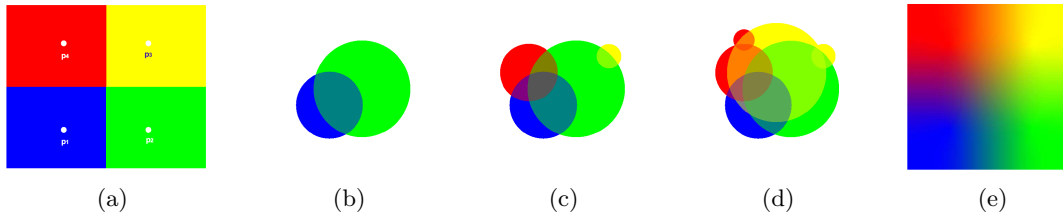


Figure 3.21: (a) Discrete computation of Sibson's interpolant for the setting in. (b)-(d) Each point in the domain gives rise to a disc colored with the value of the nearest data site and the distance to that data site as its radius, depicted. The translucent overlay of all discs is the discrete Sibson's interpolant. Pictures courtesy of [PLK<sup>+</sup>06].

computation of such a discretization for generalized Voronoi diagrams with the help of graphics hardware is discussed in [HCK<sup>+</sup>99]. Basically, the graph of the distance function from each of the data sites in the plane is represented by a geometric object. E.g., the Euclidean distance from a point is represented by a cone perpendicular to the plane and with apex at that point, the quadratic distance by a paraboloid. For curve-shaped data sites, this is a more tedious task, illustrated in Figure 3.20(a). Rendering these primitives leaves the minimum distances in the z-buffer and the associated data site in the color buffer, shown in Figure 3.20(b).

The computation of Sibson coordinates based on approximate areas can now be performed by counting pixels in the approximate Voronoi diagram with added query position [FEK<sup>+</sup>05]. However, this does not allow for an efficient or stable evaluation of Laplace or Hiyoshi's  $C^k$  coordinates, and faces severe difficulties when applied to higher dimensions.

If, instead of evaluating single point queries, the interpolant  $f^{\text{SIB}0}$  is to be evaluated over a region, the influence of the data values at the data sites can directly be *distributed* to the domain in a more efficient manner. The way described in [FEK<sup>+</sup>05] requires the Delaunay triangulation to be known, while [PLK<sup>+</sup>06] do without tessellation at all solely using a kd-tree to locate nearby points. This is illustrated in Figure 3.21.

### 3.2.10 Taxonomy of Natural Neighbor Interpolants

The properties of all interpolation schemes discussed so far are summarized in Table 3.2. A discussion of the four blocks is given below.

**Point Based Interpolation Schemes:** Schemes with global smoothness have only been proposed for the setting of point-shaped data sites. Both Farin's and Sibson's  $C^1$  constructions operate on Sibson coordinates and yield fairly straightforward implementations. Farin's construction has quadratic precision and adapts to a wider range of input constellations.

Hiyoshi's  $C^2$  scheme provides a high quality interpolant but is computationally expensive and tedious to implement even though explicit guidelines for its implementation in  $\mathbb{R}^2$  exist. At the data sites, it requires the construction of quintic Bézier control nets and bears considerable combinatorial complexity, as we examine in more detail in Section 6.5. In our experiments, we found increases in computation time for extreme situations with more

|                    | Shape of $\mathbf{x}_i$ <sup>(6)</sup> | Smoothness of $\lambda$<br>in $\mathcal{C}(\mathbf{X}) \setminus \mathbf{X}$ | Deriv. at $\mathbf{x}_i$ | Smoothness at $\mathbf{x}_i$ | Gener. to $\mathbb{R}^d$ | Gener. to $C^k$  | Precision           | Comput. Compl. <sup>(4)</sup> | Support <sup>(5)</sup> | Cont. dep. on $\mathbf{X}$ | References                          |
|--------------------|--|--|--------------------------|------------------------------|--------------------------|------------------|---------------------|-------------------------------|------------------------|----------------------------|-------------------------------------|
| <b>Point based</b> |  |  |                          |                              |                          |                  |                     |                               |                        |                            |                                     |
| $f^{\text{LAP}}$   | points                                 | $C^{d-2}$  | -                        | $C^0$                        | +                        | -                | linear              | ++                            | 1                      | +                          | [CFL82, Sug99, BIK <sup>+</sup> 97] |
| $f^{\text{SIB0}}$  |  | $C^{d-1}$  | -                        | $C^0$                        | +                        | -                |                     | ++                            | 1                      | +                          | [Sib81]                             |
| $f^{\text{HIY0}}$  |  | $C^k$  | -                        | $C^0$                        | (+) <sup>(1)</sup>       | +                |                     | -                             | 1                      | +                          | [HS00b]                             |
| $f^{\text{SIB1}}$  |  | $C^{d-1}$  | $\nabla$                 | $C^1$                        | +                        | -                | s.q. <sup>(3)</sup> | +                             | 1                      | +                          | [Sib81, Far90, Pip92]               |
| $f^{\text{FAR}}$   |  | $C^1$  | $\nabla$                 | $C^1$                        | +                        | + <sup>(2)</sup> | quad.               | +                             | 1                      | +                          | [Far90]                             |
| $f^{\text{HIY2}}$  |  | $C^2$  | $\nabla, \mathcal{H}$    | $C^2$                        | (+) <sup>(1)</sup>       | + <sup>(2)</sup> | cub.                | --                            | 1                      | +                          | [HS04, Hiy05]                       |
| $f^{\text{CLA}}$   |  | $C^1$  | -                        | $C^0$                        | +                        | -                | s.q. <sup>(3)</sup> | -                             | 2                      | +                          | [Cla96, Flö03b]                     |
| <b>Transfinite</b> |  |  |                          |                              |                          |                  |                     |                               |                        |                            |                                     |
| Gross              | pol, ci                                | $C^1$  | -                        | $C^0$                        | -                        | -                | linear              | -                             | 1                      | +                          | [GF99]                              |
| Anton              | pt, li                                 | $C^1$  | -                        | $C^0$                        | -                        | -                |                     | -                             | 1                      | +                          | [AMG98, AMG04]                      |
| Hiyoshi            | pt, li, ca                             | $C^1$  | -                        | $C^0$                        | -                        | -                |                     | -                             | 1                      | +                          | [HS00a]                             |
| <b>Manifold</b>    |  |  |                          |                              |                          |                  |                     |                               |                        |                            |                                     |
| Flötotto           | points                                 | $C^1$  | -                        | $C^0$                        | +                        | -                | n.a.                | -                             | 1                      | n.a.                       | [BC00, Flö03b]                      |
| <b>Other meth.</b> |  |  |                          |                              |                          |                  |                     |                               |                        |                            |                                     |
| $f^{\text{NEAR}}$  | points                                 | $C^{-1}$   | -                        | $C^\infty$                   | +                        | -                | -                   | ++                            | 0                      | +                          | [Thi11, OBSC00]                     |
| FEM                | points                                 | $C^k$  | -                        | $C^0$                        | +                        | -                | polyn.              | ++                            | 0                      | -                          | [ZS02]                              |
| RBF                | points                                 | $C^\infty$   | -                        | $C^\infty$                   | +                        | +                | polyn.              | -                             | g                      | -                          | [Har71, Wen04]                      |

<sup>(1)</sup> ongoing research. <sup>(2)</sup> based on the Bézier simplex approach. <sup>(3)</sup> spherical quadratics. <sup>(4)</sup> ++ low, -- high.  
<sup>(5)</sup> {012}-ring, g(lobal). <sup>(6)</sup> pt=points, li=lines, pol=polygons, ci=circles, ca=circular arcs.

Table 3.2: Overview of considered interpolation schemes.

than 20 natural neighbors, which is very likely to become an issue in higher dimensions. Besides these drawbacks, the  $C^2$  interpolant provided the best results when applied to data representing a smooth function, which has been verified in [BBU06a].

In contrast to the schemes above, Clarkson’s construction does not interpolate prescribed derivatives but achieves a smooth-looking interpolant based on an implicit construction. Since the final interpolant is a linear combination of data from the two ring neighborhood it is similar to the other  $C^0$  schemes over natural neighbor coordinates, but requires a larger support and results in non-convex coordinates.

**Transfinite Interpolation:** Research on transfinite natural neighbor interpolation has so far only concentrated on expressing the identity of Laplace and Sibson coordinates with respect to line- and circle-shaped data sites in two dimensions. Consequently, the resulting interpolants remain  $C^0$  across the data sites. In simple cases like line segments and circular arcs, closed form integration is possible, but more general shapes require approximations. In spite of these restrictions the improved flexibility provided by transfinite interpolation is useful for, e.g., fault modeling in geosciences.

**Interpolation on Manifolds:** Sibson’s identity holds on smooth manifolds for an infinitely dense sampling, but in general not for arbitrary samples. The Voronoi diagram



in that non-Euclidean metric does not have the same, simple geometric structure. However, local restriction of the Euclidean Voronoi diagram to the tangent plane reveals the lower-dimensional power diagram, for which the Laplace and Sibson's identity hold. As a result, all natural neighbor interpolants that can be computed on power diagrams can be adapted to manifolds with sufficiently dense sampling.

**Other Scattered Data Schemes:** Many other scattered data interpolation schemes exist besides those based on natural neighbors, among them finite element schemes, radial basis functions with global or local support, subdivision, and bivariate splines, all of which have advantages in certain applications. Yet, natural neighbor based interpolation offers a unique combination of the properties

- locality,
- support determined by truly automatic neighborhood,
- continuous dependency on positions of input sites.

Radial basis functions offer very good mathematical properties in terms of approximation order and smoothness and do, like natural neighbor based schemes, not depend on a particular tessellation. But even the construction of a compactly supported interpolant requires the solution of a global linear system. Finite element interpolants can be constructed with high orders of continuity but are defined over one fixed choice of elements, i.e., the tessellation of the domain, and thus do not continuously depend on the positions of the data sites. Similar arguments apply to bivariate splines and subdivision. In the more relaxed setting of scattered data *approximation*, approaches like hierarchical B-splines, thin plate splines, or moving least squares exist. Of these, only the latter has properties similar to natural neighbor based schemes and can even be integrated with natural neighbor coordinates as a replacement for inverse distance weights.

## *Related Work*

## 4 Splines over Iterated Voronoi Diagrams

This chapter presents some insights we gained after investigating an idea of Farin in his unpublished “Quadratic splines over iterated Voronoi diagrams” paper [Far03], who aimed at generalizing B-spline functions to arbitrary control meshes in the multivariate case, thus overcoming the limitations of tensor-product spline definitions. We will, after reviewing the idea, present our findings in the univariate and bivariate setting.

Specifically, we will first show an alternative argument why splines over iterated Voronoi diagrams in 1D are equivalent to B-splines up to degree two. Then, we will demonstrate that for two or more variables, the construction fails to inherit the desirable properties from the univariate case and explain the reasons for that. Finally, we point out recent findings from the finite element community who adapted the initial idea and achieved a B-spline-like construction.

For this chapter, we recommend reading Section 2.5.6 on B-spline functions.

### 4.1 Background

A B-spline function of degree  $d$ , often interchangeably used with the term B-spline, is a piecewise polynomial function that implicitly guarantees  $C^{d-1}$ -smoothness over a knot vector with non-repeating knots. Recall their definition in Section 2.5.6. B-spline functions are defined using a knot vector  $\mathcal{U} = (u_0, \dots, u_m)$ , a set of control points  $c_0, \dots, c_{m-d-1}$ , and the Cox-de Boor recursion formula (2.6)

The control points allow the user to locally control the shape of the function, while the knot vector allows to control the region of influence of each control point. This versatility and the ease of implementation made B-splines an indispensable tool in many scientific disciplines. However, limitations arise when B-spline functions are to be generalized to the multivariate case.

The standard approach here is the definition of multivariate B-splines as the tensor product of univariate B-splines, which results in a regular grid of control points and a knot vector for every variable. Such a tensor product B-spline is parameterized over a rectangular domain – a limitation that severely reduces its applicability in situations that do not exhibit such regular structure.

Another possibility is to turn to simplex splines, defined over a regular simplicial tessellation of the domain, but the regularity constraints are again a limiting factor.

True multivariate B-splines with arbitrary control point structures are defined as *B-polynomials* and *B-patches*, for which a concise treatment can be found in [PBP02]. These

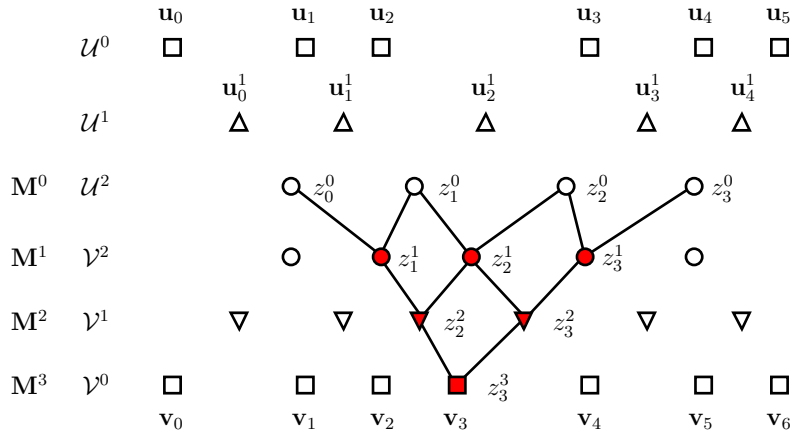


Figure 4.1: Illustration of the iterated Voronoi diagram scheme. Levels  $\mathcal{U}^0$  through  $\mathcal{U}^2$  generate the control grid. The shaded points in levels  $\mathcal{V}^0$  through  $\mathcal{V}^2$  depend on the query position  $\mathbf{v}_3$ .

multivariate splines exhibit a rather unintelligible connection between the function and the knots, which can be chosen with a great deal of freedom but lead to structures of rather theoretical interest. In [Nea01], the choice of so-called knot chains is discussed in an effort to find a suitable generalization of univariate splines to higher dimensions.

All the above methods, however, either lack flexibility or the elegance of the univariate B-splines. Realizing that the central operation in de Boor’s algorithm is repeated linear blending of two control points with ratios that differ from level to level, the transition into the terminology of natural neighbor coordinates promises increased flexibility while keeping the framework relatively elegant.

## 4.2 Farin’s Splines over Iterated Voronoi Diagrams

Next we revisit the idea described in Farin’s technical report entitled *Quadratic splines over iterated Voronoi diagrams* [Far03], and describe how it fits into the computation of local coordinates.

### 4.2.1 Iterated Sibson’s Interpolation

We will closely follow the notation used in the original paper and refer to the method as *Iterated Sibson’s Interpolation*. The individual steps are illustrated for the univariate case in Figure 4.1.

Let  $\mathcal{U}^0 \subset \mathbb{R}^2$  be a set of points  $\mathbf{u}_i = \mathbf{u}_i^0$ . We construct the Voronoi diagram of  $\mathcal{U}^0$  and define from its Voronoi vertices a set  $\mathcal{U}^1$  with points  $\mathbf{u}_i^1$ , and repeat this until we have a set  $\mathcal{U}^{k-1}$ . The integer  $k$  is called the *depth* of the scheme, and  $\mathcal{U}^{k-1}$  is called the *control grid*. With each point  $\mathbf{u}_i^{k-1}$  in the control grid we associate a value  $z_i^0$ , which gives us a *control mesh*  $\mathbf{M}^0 = (\mathcal{U}^{k-1}, Z^0)$  that shall influence the shape of the resulting surface just like the control polygon influences the shape of a B-spline curve.

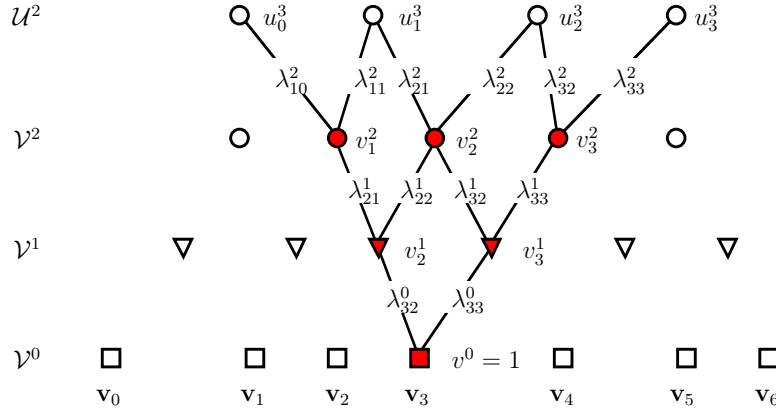


Figure 4.2: Scheme to compute the local coordinates by partitioning the value of one level by level.

The evaluation of that surface at a query point  $\mathbf{v}$  is described next. We insert  $\mathbf{v}$  into the set of knots which gives us  $\mathcal{V}^0 = \mathcal{U}^0 \cup \{\mathbf{v}\}$ . Like above we construct the Voronoi diagram of  $\mathcal{V}^0$ , whose vertices form the set  $\mathcal{V}^1$ , and repeat this until we get  $\mathcal{V}^{k-1}$ . In the following, running indices address those vertices in  $\mathcal{V}^{k-1}$  depending on  $\mathbf{v}$ . The vertices of  $\mathcal{V}^{k-1}$  may be expressed in terms of those of  $\mathcal{U}^{k-1}$  using Sibson's identity, which was defined in Section 3.2.5.3 and, without loss of generality, we can write

$$\mathbf{v}_i^{k-1} = \sum_j \lambda_{ij}^{k-1} \mathbf{u}_j^{k-1}.$$

With these coefficients, we evaluate Sibson's interpolant  $f^{\text{SIB}0}$  for  $\mathbf{M}^0$  at every point  $\mathbf{v}_i \in \mathcal{V}^{k-1}$ ,

$$z_i^1(\mathbf{v}) = \sum_j \lambda_{ij}^{k-1} z_j^0,$$

and obtain a mesh  $\mathbf{M}^1 = (\mathcal{V}^{k-1}, Z^1)$ . The same way we expressed  $\mathcal{V}^{k-1}$  in terms of  $\mathcal{U}^{k-1}$ , we can express  $\mathcal{V}^{k-2}$  in terms of  $\mathcal{V}^{k-1}$ , generating the control mesh  $\mathbf{M}^2 = (\mathcal{V}^{k-2}, Z^2)$ . After continuing for meshes  $\mathbf{M}^{k-r} = (\mathcal{V}^r, Z^{k-r})$  with

$$z_i^{r+1} = \sum_j \lambda_{ij}^r z_j^r, \quad (4.1)$$

we eventually obtain mesh  $\mathbf{M}^k = (\mathcal{V}^0, Z^k)$ . Only one point in  $\mathbf{M}^k$  depends on  $\mathbf{v}$ , namely  $(\mathbf{v}, z^k(\mathbf{v}))$ ; this is the desired point on the surface.

## 4.2.2 Representation as Local Coordinates

The algorithm describes the evaluation process by iteratively applying Sibson's interpolant to a hierarchy of grids. In the spirit of this thesis, however, we wish to express  $\mathbf{v}$  in local coordinates with respect to the control grid in  $\mathcal{U}^{k-1}$ . The above starts from the

control mesh  $\mathbf{M}^0$  at the bottom level, and for  $k = 2$ , we get

$$\begin{aligned}
 \mathbf{v} &= \sum_i \lambda_i^0(\mathbf{v}) \mathbf{v}_i^0, \\
 &= \sum_i \lambda_i^0(\mathbf{v}) \sum_j \lambda_{ij}^1 \mathbf{v}_j^1, \\
 &= \sum_i \lambda_i^0(\mathbf{v}) \sum_j \lambda_{ij}^1 \sum_l \lambda_{jl}^2 \mathbf{u}_l^1.
 \end{aligned} \tag{4.2}$$

Expanding the above and pulling  $\mathbf{v}_l^2$  to the front gives us

$$\mathbf{v} = \sum_l \left( \sum_{i,j} \lambda_i^0(\mathbf{v}) \lambda_{ij}^1 \lambda_{jl}^2 \right) \mathbf{u}_l^1 =: \sum_l \lambda_l^{\text{ITS}^2} \mathbf{u}_l^1,$$

where ITS stands for *iterated Sibson*. This lets us fit the proposed scheme into the general framework of this thesis, giving the resulting interpolant by

$$f^{\text{ITS}^k}(\mathbf{v}) = \sum_i \lambda_i^{\text{ITS}^k} z_i^0.$$

The following properties are obvious from the above exposition.

1. The support of  $f^{\text{ITS}^k}(\mathbf{v})$  is local but depends on  $k$ .
2. Because the scheme arising from (4.2) employs only affine combinations of barycentric coordinates, the interpolant has linear precision.

The scheme in (4.2) is illustrated in Figure 4.2 and consists of the steps described next. First, compute the hierarchy  $\mathcal{U}^0 \dots \mathcal{U}^{k-1}$ , the figure displays only the resulting  $\mathcal{U}^{k-1}$ . Create  $\mathcal{V}^0$  and associate a value of  $v^0 = 1$  with  $\mathbf{v}$ . Create  $\mathcal{V}^1$  and compute  $\lambda_i^1(\mathbf{v})$ . Split  $v^0$  proportional to  $\lambda_i^1(\mathbf{v})$  and add the fractions to values  $v_i^1$ . Create  $\mathcal{V}^2$  and compute for every  $\mathbf{v}_i^1$  with  $v_i \neq 0$  its local coordinates  $\lambda_{ij}^2$ . Split  $v_i^1$  proportional to  $\lambda_{ij}^2$ , and add the fractions to values  $v_j^2$ . Repeat this until  $\mathcal{V}^{k-1}$  is to be expressed in terms of  $\mathcal{U}^{k-1}$ . Computing  $\lambda_{ij}^k$ , splitting  $v_i^{k-1}$  accordingly and adding the fractions to  $u_j^k$  gives the local coordinates of  $\mathbf{v}$  with respect to the control points  $\mathbf{u}_j$  by  $\lambda_j^{\text{ITS}^k} = u_j^k$ ,

$$\mathbf{v} = \sum_j \lambda_j^{\text{ITS}^k} \mathbf{u}_j.$$

### 4.3 Equivalence with B-splines in the Univariate Case

Here, we retrace the reasoning given in [Far03] that the proposed iterated Sibson's interpolation produces quadratic B-splines for  $k = 2$  in the univariate, functional case. The original reasoning referenced de Boor's algorithm, yet we find it more suitable to relate the construction to repeated knot insertion, which is another well-known method to evaluate a B-spline curve.

### 4.3.1 Evaluating Quadratic B-Splines Using de Boor

We start by applying the de Boor algorithm step by step to evaluate a univariate B-spline of degree two, which allows us to associate the corresponding constructions in the iterated Sibson's interpolation.

We assume a knot vector  $(u_0, \dots, u_m)$  with control points  $(z_0, \dots, z_{m-1})$  assigned to the Greville abscissae  $(\xi_0, \dots, \xi_{m-1})$ ,  $\xi_i = (u_i + u_{i+1})/2$ . The evaluation is performed for  $v \in [u_j, u_{j+1})$ . The first step of the de Boor algorithm according to Section 2.5.7 computes  $z_j^1$  and  $z_{j+1}^1$ ,

$$\begin{aligned} z_j^1 &= \frac{u_{j+2} - v}{u_{j+2} - u_j} z_{j-1} + \frac{v - u_j}{u_{j+2} - u_j} z_j, \\ z_{j+1}^1 &= \frac{u_{j+3} - v}{u_{j+3} - u_{j+1}} z_j + \frac{v - u_{j+1}}{u_{j+3} - u_{j+1}} z_{j+1}. \end{aligned}$$

The second step computes  $z_{j+1}^2$ , which is the value of the B-spline curve at  $v$ , as

$$z_{j+1}^2 = \frac{u_{j+2} - v}{u_{j+2} - u_{j+1}} z_j^1 + \frac{v - u_{j+1}}{u_{j+2} - u_{j+1}} z_{j+1}^1.$$

### 4.3.2 Evaluating Quadratic B-Splines Using Repeated Knot Insertion

We start out as in the previous section, but this time performing two knot insertions at  $v \in [u_j, u_{j+1})$ ,

$$\begin{aligned} (\xi_0^1, \dots, \xi_m^1) &\leftarrow \text{Greville abscissae of } (u_0, \dots, u_j, v, u_{j+1} \dots u_m), \\ (\xi_0^2, \dots, \xi_{m+1}^2) &\leftarrow \text{Greville abscissae of } (u_0, \dots, u_j, v, v, u_{j+1} \dots u_{m+1}), \end{aligned}$$

which lead to the new Greville abscissae  $\xi_j^1$  and  $\xi_{j+1}^1$  in the first insertion and  $\xi_j^2 = v$  in the second. The corresponding control points  $(z_0^1, \dots, z_m^1)$  of the refined control polygon are found by evaluating the piecewise linear interpolant over  $(\xi_i, z_i)$ ,  $i = 0, \dots, m$ , at the new Greville abscissae  $\xi_k^1$ ,  $k = 0, \dots, m$ , specifically

$$\begin{aligned} z_j^1 &= \frac{\xi_j - \xi_j^1}{\xi_j - \xi_{j-1}} z_{j-1} + \frac{\xi_j^1 - \xi_{j-1}}{\xi_j - \xi_{j-1}} z_j, \\ z_{j+1}^1 &= \frac{\xi_{j+1} - \xi_{j+1}^1}{\xi_{j+1} - \xi_j} z_j + \frac{\xi_{j+1}^1 - \xi_j}{\xi_{j+1} - \xi_j} z_{j+1}. \end{aligned} \tag{4.3}$$

Now,  $(z_0^2, \dots, z_{m+1}^2)$  are found by evaluating the piecewise linear interpolant over  $(\xi_i^1, z_i^1)$ ,  $i = 0, \dots, m$ , at the new Greville abscissae  $\xi_k^2$ ,  $k = 0, \dots, m + 1$ , specifically

$$z_{j+1}^2 = \frac{\xi_{j+1}^1 - \xi_{j+1}^2}{\xi_{j+1}^1 - \xi_j^1} z_j^1 + \frac{\xi_{j+1}^2 - \xi_j^1}{\xi_{j+1}^1 - \xi_j^1} z_{j+1}^1. \tag{4.4}$$

Finally,  $z_{j+1}^2$  is the quadratic B-spline function  $b$  evaluated at  $v$ .

### 4.3.3 Evaluating Quadratic B-Splines Using Iterated Sibson's Interpolation

Now we switch to the notion of the univariate Voronoi diagram, which means  $(u_0, \dots, u_m)$  are Voronoi sites and  $(u_0^1, \dots, u_{m-1}^1)$  the corresponding Voronoi vertices, to which we assign control points  $z_i$ . In one dimension, linear interpolation between two Greville abscissae is identical to Sibson's interpolation in the set of Voronoi sites.

According to Farin, we evaluate the function by inserting the evaluation abscissa  $v \in [u_j, u_{j+1})$  into the set of Voronoi sites,

$$(v_0^0, \dots, v_{m+1}^0) := (u_0, \dots, u_j, v, u_{j+1}, \dots, u_m),$$

and get the corresponding Voronoi vertices

$$(v_0^1, \dots, v_m^1).$$

The only new Voronoi vertices that do not coincide with an old one are  $v_j^1$  and  $v_{j+1}^1$ . We then compute intermediate control points for the corresponding Voronoi vertices by evaluating Sibson's interpolant over  $(u_i^1, z_i)$ ,  $i = 0, \dots, m - 1$ , and find

$$\begin{aligned} z_j^1 &= \frac{u_j^1 - v_j^1}{u_j^1 - u_{j-1}^1} z_{j-1} + \frac{v_j^1 - u_{j-1}^1}{u_j^1 - u_{j-1}^1} z_j, \\ z_{j+1}^1 &= \frac{u_{j+1}^1 - v_{j+1}^1}{u_{j+1}^1 - u_j^1} z_j + \frac{v_{j+1}^1 - u_j^1}{u_{j+1}^1 - u_j^1} z_{j+1}, \end{aligned}$$

just like in (4.3). Now,  $v$  is to be expressed in terms of  $v_0^1, \dots, v_m^1$  and  $z_j^2$  to be interpolated accordingly. The piecewise interpolant over  $(v_i^1, z_i^1)$ ,  $i = 0, \dots, m$  at  $v$  yields

$$z_j^2 = \frac{v_{j+1}^1 - v}{v_{j+1}^1 - v_j^1} z_j^1 + \frac{v - v_j^1}{v_{j+1}^1 - v_j^1} z_{j+1}^1,$$

which together with  $v = \xi_{j+1}^2$  in (4.4) shows that for  $k = 2$ , iterated Sibson's interpolation is equivalent to de Boor's algorithm resp. knot multiplication and produces B-Spline functions of degree two.

### 4.3.4 Difference for Degree Greater Two

We showed in Section 4.3.2 and Section 4.3.3 the equivalence of the two algorithms based on their identical set of operations on the Greville abscissae  $\xi_i$  and the one-time iterated Voronoi vertices  $u_i^1$ .

The general formula for Greville abscissae  $\xi_i^{d-1}$  of a B-spline curve of degree  $d$  is

$$\xi_i^{d-1} = \frac{1}{d} \sum_{j=0}^{d-1} u_{i+j}.$$



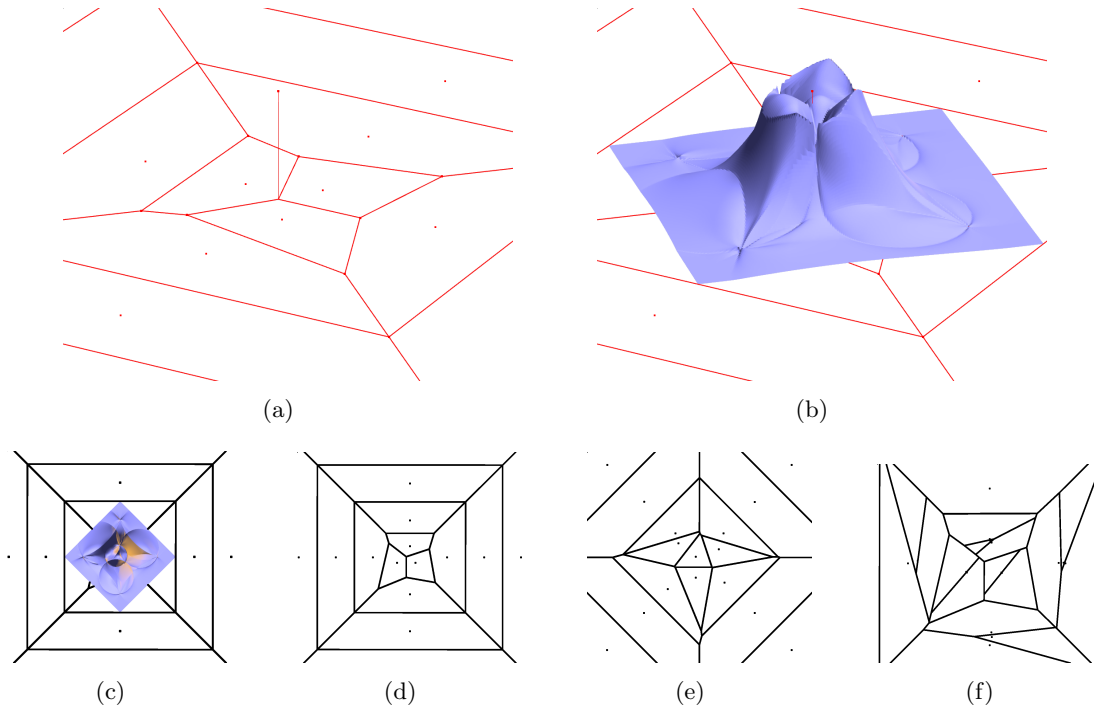


Figure 4.3: (a) Data set with central control point set to one. (b) The function generated by iterated Sibson's interpolation. (c) The function from above. (d)-(f) The control net configurations  $\mathcal{U}^0$ ,  $\mathcal{U}^1$ ,  $\mathcal{U}^2$ .

Now we look at the way the control mesh ordinates  $\mathcal{U}^{k-1}$  are computed, and find

$$\begin{aligned}
 u_i^{k-1} &= \frac{1}{2}(u_i^{k-2} + u_{i+1}^{k-2}) \\
 &= \frac{1}{2}\left(\frac{1}{2}(u_i^{k-3} + u_{i+1}^{k-3}) + \frac{1}{2}(u_{i+1}^{k-3} + u_{i+2}^{k-3})\right) \\
 &= \dots \\
 &= \frac{1}{2^{k-1}} \sum_{j=0}^{k-1} \binom{k-1}{i} u_{i+j}^0.
 \end{aligned}$$

Obviously,  $u_i^{k-1}$  and  $\xi_i^{d-1}$  agree for  $k = d = 1, 2$ . For  $k, d > 2$ , the control polygons for both methods are different in general and lead to different results, which can easily be verified.

## 4.4 Failure in the Bivariate Case

We have implemented iterated Sibson's interpolation in 2D for  $k = 2$ . The resulting function for a control mesh  $\mathbf{M}^0$  with all values set to zero except the one in the middle can be seen in Figure 4.3(a) and (b).

In Section 4.3 we have shown the equivalence of iterated Sibson’s interpolation in 1D and quadratic B-splines for the case  $k = 2$ . Although this cannot be the case in general for multivariate settings due to difference in the knot structures, we would expect the main properties of B-splines like smoothness to carry over to iterated Sibson’s interpolation. Figure 4.3(b) shows that this is not the case, and we discuss the reasons next.

In repeated knot insertion, control points at different levels receive values from other control points by means of convex combinations that reflect the monotone sequence of Greville abscissae they are associated with. This is also true for iterated Sibson’s interpolation in the univariate case, and for tensor-product B-splines, where the argument applies in every parameter direction. Common triangular functional subdivision algorithms, which can to some extent be regarded as a generalization of uniform B-splines to triangular domains, also build their refined control nets by computing convex combinations.

Applied to the steps of iterated Sibson’s interpolation, this means that the abscissa of a new control point should be contained in the convex hull of the points from which it is computed. However, since the abscissae  $\mathcal{U}^r$  and  $\mathcal{V}^r$  are the vertices in Voronoi diagrams, they are the circumcenters of Delaunay triangles, and fall within the convex hull of their generating points only if the corresponding triangle is non-obtuse. A sequence of control points with their associated Voronoi diagrams is shown in Figure 4.3(c) - (f). As a consequence, the intermediate abscissae that receive values by means of Sibson’s interpolation have no meaningful spatial correlation with their generators, which explains the unexpected shape of the basis function shown in Figure 4.3(b).

## 4.5 González’ Voronoi Splines

Based on Farin’s idea, a meaningful result has been published by González et al. in [GCD07]. Their method takes yet another perspective at the evaluation of B-spline curves and adapts it to the setting of Voronoi diagrams.

### 4.5.1 Method Description

Their key observation is related to the way ratios in the de Boor recursion are computed to mix control points. Looking at the de Boor algorithm in (2.7), the mixing ratio for two control points at level  $n$  for parameter  $u$  is defined from the linear interpolant between zero and one over the interval  $[u_i, u_{i+d-n})$ ,

$$\alpha_i^n = \frac{u - u_i}{u_{i+d-n} - u_i},$$

where the denominator is the length of an interval formed by removing  $u_{i-1}, \dots, u_{i+d-n-1}$  from the knot sequence. These removed knots are always contained in a certain neighborhood of  $u$ .

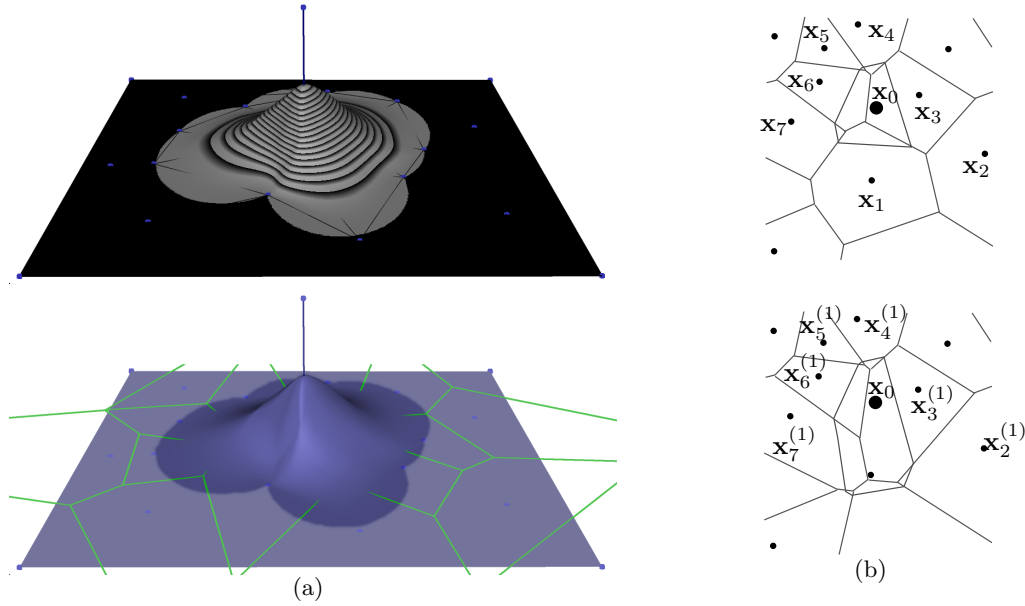


Figure 4.4: (a) Function over a data set with all values set to zero except for the middle one. This is not exactly a “basis” function since the equivalent to data points in this method are the values associated with the midpoints of the Delaunay edges. (b) Top: virtual tile used in computation of local coordinates for  $\mathbf{x}_0$  in  $\mathbf{X}$ . Bottom: virtual tile used in computation of local coordinates for  $\mathbf{x}_0$  in  $\mathbf{X}^{(\mathbf{x}_1)}$ .

Adapted to the setting of Voronoi diagrams<sup>1</sup>, this corresponds to removing knots  $\mathbf{x}_i \in N(\mathbf{x}_0)$  from the neighborhood of the query position  $\mathbf{x}_0$ , and computing its local coordinates with respect to the remaining knots  $\mathbf{X} \setminus \mathbf{x}_i$ . For B-spline curves, each set of removed knots corresponds to two neighboring control points at a certain level that are averaged. In the method of Gonzáles et al., who give details of their method only for the quadratic case, every pair of natural neighbors in the knot set is associated with a control point, which is naturally visualized at the mid-point of their connecting Delaunay edge.

We now describe their method for the quadratic case. Let  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_m\} \subset \mathbb{R}^n$ , and we seek to express  $\mathbf{x}_0$  as a convex combination of points in  $\mathbf{X}$ . We denote by  $N_0^{(j)}$  the natural neighbors of  $\mathbf{x}_0$  in  $\mathbf{X} \setminus \{\mathbf{x}_j\}$ , and by  $\lambda_i^{(j)}$ ,  $i \in N_0^{(j)}$  corresponding natural neighbor coordinates of  $\mathbf{x}_0$ , see Figure 4.4(b). By  $\lambda_i$ ,  $i \in N_0$  we denote some natural neighbor coordinates of  $\mathbf{x}_0$  in  $\mathbf{X}$ , i.e., any of  $\lambda^{\text{LAP}}$ ,  $\lambda^{\text{SIB}}$ ,  $\lambda^{\text{HIY}}$ . Gonzáles' coordinates in the quadratic setting are now defined as

$$\lambda_j^{\text{GON}} = \sum_{i \in N_0} \lambda_i \lambda_j^{(i)}. \quad (4.5)$$

An example of a surface computed using their method can be seen in Figure 4.4(a).

<sup>1</sup>Note that for reasons of consistency with the remaining chapters, we use  $\mathbf{X}$  instead of  $\mathcal{U}$  as in previous sections.

### 4.5.2 Discussion of González’ Method and Future Research

The method indeed creates functional surfaces very much the same way as B-splines, locally influencing the shape of the function with a control point.

However, the straightforward and elegant correspondence between the maxima of the *basis functions* and the control points is not very obvious. Therefore, the usefulness of this kind of B-spline-like functions for direct surface modeling based on the control points seems limited. It should, however, be possible to transfer the direct manipulation paradigm [HHK92] from B-spline curves and tensor-product B-spline surfaces to the Voronoi spline construction of González et al.

One crucial difference between tensor-product B-splines and the Voronoi spline construction is the lack of smoothness at the data sites. Because of the multiplicative combination of natural neighbor coordinates in (4.5), the coordinates  $\lambda^{\text{GON}}$  inherit the  $C^0$  limitation at the data sites. To overcome this, the evaluation of the interpolant  $f^{\text{GON}}$  could be split into the evaluation of Sibson’s interpolant at  $\mathbf{x}_0$  with every neighbor removed one after the other, and then interpolating the acquired function values, which are now associated with the natural neighbors, using  $f^{\text{FAR}}$ . The gradient required in this approach can be estimated like described in Chapter 6.

Voronoi splines are only  $C^0$  at the data sites. The control points can be chosen such that the interpolant has polynomial precision, yet the description of the scheme for Voronoi splines corresponding to degrees higher than two is not as straightforward as claimed in [GCD07]. The removal of knots in the neighborhood of the query position in the univariate can obviously be translated into the removal of natural neighbors in the Voronoi setting if only one “ring” of neighbors is to be removed. But if multiple stages of neighbors are involved, it is not clear whether to remove the whole one-ring and compute the natural neighbor coordinates with each of the then-natural neighbors one by one removed, or whether to apply this to any combination of one- and two-ring neighbors removed. This ambiguity needs to be clarified by further investigation.

One could extend the method to knots that are not point-shaped, as has been done in the interpolation methods presented in Section 3.2.6 on transfinite natural neighbor interpolation. This generalization would allow more flexible control over the shape of the function and could greatly widen the range of potential applications.

## 4.6 Conclusion

We have reviewed and implemented an idea by Farin that is based on an observed correspondence between structures and steps used in the evaluation of B-spline curves and natural neighbor interpolation in the Voronoi diagram.

An equivalence of the two methods is easy to show in the univariate case for B-spline curves of degrees one and two based on the interpretation of the process of repeated knot insertion, also known as the de Boor algorithm. However, this equivalence is lost in generalizations to higher degrees because of the difference in the generation of the Greville abscissae and their counterparts in the iterated Sibson’s interpolation.

Our implementation of the method for two dimensions revealed that the hoped-for similarity in the behavior of the functional surface with respect to the control points is not present. We traced the reasons back to the difference in the construction of Greville abscissae in the case of B-splines and their corresponding Voronoi vertices at a certain level of Farin's construction. In B-splines, the Greville abscissae at different levels of the de Boor algorithm are always formed by convex combinations of Greville abscissae of the last level. In the iterated Sibson's construction, new Voronoi vertices are by definition placed at the circumcenters of Voronoi vertices of the last level – a non-convex construction in general.

Finally we focused on findings by González et al., whose method fulfills most of the properties that the original idea by Farin set out to achieve, and sketched potential improvements and possibilities for further research.



# 5 Practical Implementation of Higher Order Natural Neighbor Coordinates

This chapter builds on the concepts introduced in Section 2.1 on linear algebra, and Section 2.3.2 on polyhedra and polyhedral complexes for the understanding of algebraic volume computation. The implementation of natural neighbor coordinates draws from Section 3.2.5 on the definition of natural neighbor coordinates and the Voronoi and power diagram from Section 2.3.

The main contribution of this chapter consists of guidelines for an efficient and robust implementation of Hiyoshi  $C^k$  coordinates in 2D with a possible extension to 3D or higher dimensions. The results given in this section have been published in [BBU06b, BBU06c].

## 5.1 Background

The computation of local coordinates in point clouds is a crucial building block of many algorithms that operate on local subsets of larger data sets. In such settings, natural neighbor coordinates have proved useful tools, and both Laplace and Sibson coordinates have been successfully applied. While all natural neighbor coordinate schemes presented in Section 3.2.5 are  $C^0$  continuous in  $\mathbf{X}$ , they differ in their continuity in  $\mathcal{C}(\mathbf{X}) \setminus \mathbf{X}$ , specifically in their continuity across the Delaunay circumspheres. With the introduction of natural neighbor coordinates that are  $C^k$ -continuous in  $\mathcal{C}(\mathbf{X}) \setminus \mathbf{X}$  by Hiyoshi in [HS00b], natural neighbor coordinates became interesting for problem settings such as higher order meshfree finite element methods. However, no compelling instructions for their implementation were given, and in contrast to Laplace and Sibson coordinates, their geometrical definition on the Voronoi diagram is complex and involved, which limited their adoption in practice.

For the computation of Sibson coordinates in Delaunay triangulations, Watson has already proposed an alternative approach in [Wat92], which unfortunately suffers from numerical instabilities if the coordinates are to be computed for positions on or near facets of Delaunay simplices, as discussed in Section 3.2.5.4. Another alternative was given by Braun and Sambridge in [BS95], who proposed to treat Voronoi tiles in the  $\mathcal{H}$ -representation of convex polytopes instead of their  $\mathcal{V}$ -representation, as introduced in Section 2.3.2. This way, they can use the recursive volume computation formula of Lasserre [Las83], which in addition provides the volumes of all sub-polyhedra of the polyhedral complex in the course of computation – which is required in, e.g., the computation of Laplace coordinates. A good overview on volume computation methods for convex polyhedra in  $\mathcal{H}$ - and  $\mathcal{V}$ -representation can be found in the work by Bueler et al. in [BEF00].

Inspired by this, we developed a scheme that allows to compute Hiyoshi coordinates for a point based on the  $\mathcal{H}$ -representation of its Voronoi tile.

We like to also point to Hiyoshi who presented directions for the implementation of his  $C^k$  coordinates based on a modification of Watson's approach in [Hiy05], where he put special emphasis on the robustness of the computation. His approach differs from the one we present here and might be considered another option.

## 5.2 Algebraic Volume Computation

The main motivation for the investigation of algebraic volume computation methods lies in the nature of natural neighbor coordinates: they are defined by geometric entities of the Voronoi diagram, which are convex polytopes. As we will show at the end of this section, the geometric entities required in the computation of Laplace-, Sibson-, and Hiyoshi-coordinates can be given in  $\mathcal{H}$ -representation in order to evaluate them using Lasserre's method. This representation has the same low complexity in any dimension, and is considerably less complex than the  $\mathcal{V}$ -representation already for dimensions greater than two.

We first point out why the volume computation based on a polytope's  $\mathcal{V}$ -representation is not practical, then describe Lasserre's method in detail.

### 5.2.1 Triangulation of the Convex Hull

Consider the  $\mathcal{V}$ -representation of a convex,  $n$ -dimensional polytope with vertices  $\mathbf{v}_1, \dots, \mathbf{v}_m$ . There exists a triangulation of  $\mathbf{v}_1, \dots, \mathbf{v}_m$  into simplices  $S_1, \dots, S_k$ , from which the polytope volume can be determined by summing up the volumes of  $S_1, \dots, S_k$ , where the volume of an  $n$ -simplex  $S \in \mathbb{R}^n$  with vertices  $\mathbf{x}_0, \dots, \mathbf{x}_n$  can be computed by

$$\text{Vol}(n, S) = \frac{1}{n!} \det \begin{bmatrix} \mathbf{x}_1 - \mathbf{x}_0 & \dots & \mathbf{x}_n - \mathbf{x}_0 \end{bmatrix}.$$

A thorough investigation of volume computation codes has been conducted in [BEF00]. Although the above definition based on the  $\mathcal{V}$ -representation is viable in theory, practical considerations suggest other alternatives: Each polyhedron facet has an individual number of vertices, which makes them hard to handle in data structures, and with increasing dimension, the handling of the topological entities requires rather abstract combinatorial treatment.

In  $\mathcal{H}$ -representation, however, each facet of the polyhedron is represented by a vector and a scalar, which is considerably more efficient to store and process.

### 5.2.2 Lasserre's Method

The following revisits Lasserre's recursive algebraic volume computation, introduced in [Las83]. Another approach to compute the volume of a convex polytope has been sug-



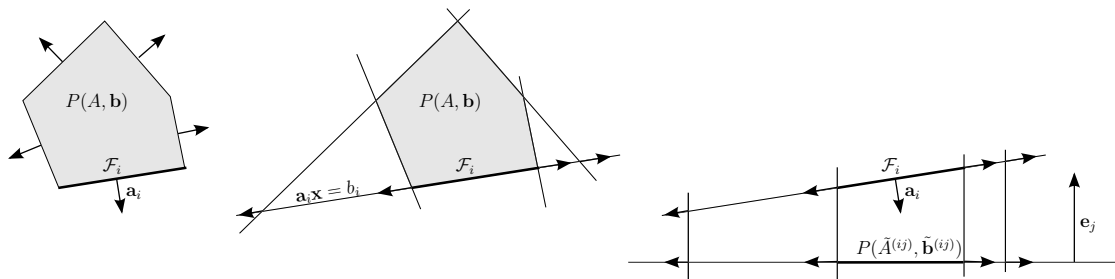


Figure 5.1: (Left) The polytope,  $\mathcal{F}_i$  drawn bold. (Middle) The restriction of the polytope to  $\mathbf{a}_i \mathbf{x} = b_i$ . (Right) The projection of the restriction along  $\mathbf{e}_j$ .

gested by Lasserre et al. in [LZ01], yet it lacks the easy, recursive structure of the scheme we consider here and does not fit in our context. Lasserre states a relation between the  $n$ -dimensional volume of a polyhedron in  $\mathbb{R}^n$  and the  $n - 1$ -dimensional volumes of its facets, allowing to break the computation of the polyhedral volume recursively down into the computation of interval lengths in  $\mathbb{R}$ .

We will use the following notation for an  $n$ -dimensional convex polyhedron with  $m$  facets in half-space representation as defined in Section 2.3.2:

$$P(A, \mathbf{b}) := \{ \mathbf{x} : A\mathbf{x} \leq \mathbf{b} \},$$

where  $A = (\mathbf{a}_1, \dots, \mathbf{a}_m)^T \in \mathbb{R}^{m \times n}$  is the matrix of face normals  $\mathbf{a}_i \in \mathbb{R}^n$ , and  $\mathbf{b} \in \mathbb{R}^m$ .

We denote by  $\text{Vol}(n, A, \mathbf{b}) = \text{Vol}(n, P(A, \mathbf{b}))$  the volume of  $P(A, \mathbf{b})$ , and by

$$\text{Vol}_i(n - 1, A, \mathbf{b}) = \text{Vol}(n - 1, P(A, \mathbf{b}) \cap \{ \mathbf{x} : \mathbf{a}_i \mathbf{x} = b_i \})$$

the volume of the  $i$ -th face, where  $\text{Vol}(n - 1, \cdot)$  denotes the volume in the appropriate  $n - 1$ -dimensional affine subspace  $\{ \mathbf{x} \in \mathbb{R}^n : \mathbf{a}_i \mathbf{x} = b_i \}$ .

Lasserre states the relation

$$\text{Vol}(n, A, \mathbf{b}) = \frac{1}{n} \sum_{i=1, \dots, m} \frac{b_i}{\|\mathbf{a}_i\|} \text{Vol}_i(n - 1, A, \mathbf{b}), \quad (5.1)$$

which tells us how to compute the  $n$ -dimensional volume of  $P$  from the  $n - 1$ -dimensional volumes of its faces. To apply the approach recursively to face  $i$  of  $P$ , which is an  $n - 1$ -dimensional polytope in  $\mathbb{R}^n$ , we need to find a proper  $\mathcal{H}$ -representation for it in  $\mathbb{R}^{n-1}$ .

To this end, Lasserre proposes to project the facet to a suitable Cartesian subspace  $\{ \mathbf{x} : \mathbf{x}^T \mathbf{e}_j = 0 \}$ , where  $\mathbf{e}_j$  is the  $j$ -th orthonormal basis vector, thus reducing the problem to a truly  $n - 1$ -dimensional one, by the following process. From the constraint  $\mathbf{a}_i \mathbf{x} = b_i$  defining facet  $\mathcal{F}_i$ , we substitute

$$x_j = (b_i - \sum_{k \neq j} a_{ik} x_k) / a_{ij} \quad (5.2)$$

into  $A$ , which allows us to eliminate column  $j$  from  $A$  and the constraint in row  $i$  from  $A$

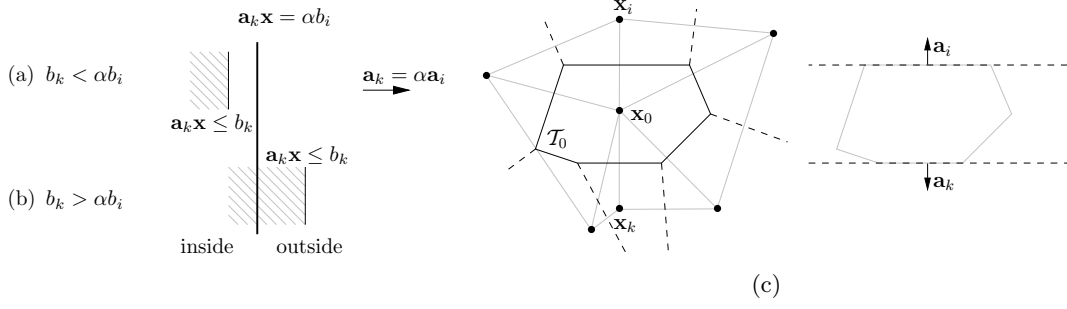


Figure 5.2: The half-space defined by  $\mathbf{a}_k \mathbf{x} \leq b_k$  either (a) is in conflict with constraint  $\mathbf{a}_i \mathbf{x} = b_i$ , or (b) is redundant since it does not limit facet  $i$  in any way. (c) Three collinear Voronoi sites  $\mathbf{x}_0$ ,  $\mathbf{x}_i$ ,  $\mathbf{x}_k$  and the resulting tile  $\mathcal{T}_0$  with two parallel facets (edges in this 2D illustration).

and  $\mathbf{b}$ , and we denote the projection of  $\mathcal{F}_i$  along  $\mathbf{e}_j$  by

$$\tilde{P}^{(ij)} = \left\{ \tilde{\mathbf{x}} \in \mathbb{R}^{n-1} : \tilde{A}^{(ij)} \tilde{\mathbf{x}} \leq \tilde{\mathbf{b}}^{(ij)} \right\}, \quad \tilde{A}^{(ij)} \in \mathbb{R}^{m-1 \times n-1}, \quad \tilde{\mathbf{b}}^{(ij)} \in \mathbb{R}^{n-1}.$$

The projection changes the volume of  $\tilde{P}^{(ij)}$  by the factor  $a_{ij}$ , which now allows to replace the face volume in (5.1) by one that can be computed by the directly applicable recursion

$$\text{Vol}(n, A, \mathbf{b}) = \frac{1}{n} \sum_{i=1, \dots, m} \frac{b_i}{|a_{ij}|} \text{Vol}(n-1, \tilde{A}^{(ij)}, \tilde{\mathbf{b}}^{(ij)}). \quad (5.3)$$

A good choice for  $\mathbf{e}_j$  is such that  $|a_{ij}| = \max(|a_{i1}|, \dots, |a_{in}|)$ , as this is the direction in which face  $i$  is least distorted during projection, and exhibits the least round-off errors in the transformation from  $A$  to  $\tilde{A}^{(ij)}$ .

To substitute (5.2) into  $A$  in the computation of  $\tilde{A}^{(ij)}$ , we assume without loss of generality  $i = m$ , and  $j = n$ , i.e., we eliminate the last constraint and project along the last orthonormal basis vector. For  $1 \leq k < m$  and  $1 \leq l < n$ , we compute

$$\tilde{a}_{kl} = a_{kl} - \frac{a_{kn}}{a_{mn}} a_{ml}, \quad \tilde{b}_k = b_k - \frac{a_{kn}}{a_{mn}} b_m. \quad (5.4)$$

The occurrence of a zero row  $\tilde{\mathbf{a}}_k = \mathbf{0}$  in  $\tilde{A}^{(ij)}$  results from collinearity of  $\mathbf{a}_i$  and  $\mathbf{a}_k$  and shows that facets  $i$  and  $k$  are parallel. We have found that checking for  $\|\tilde{\mathbf{a}}_k\|_1 < \varepsilon |\tilde{b}_k|$  provides a good estimate for collinearity, where  $\varepsilon$  is chosen to accommodate for a suitable multiple of the floating point precision  $fp$ , say  $\varepsilon = 100fp$ . As [BEF00] pointed out, in every level of the recursion it is required that the set of defining half-spaces is minimal, since if a single facet is represented by two or more constraints it would be counted more than once in the evaluation. More generally, if an empty row  $k$  is detected during substitution of (5.2) into  $A\mathbf{x} \leq \mathbf{b}$ , the constraint

$$\tilde{\mathbf{a}}_k \mathbf{x} = \mathbf{0} \mathbf{x} \leq \tilde{b}_k, \quad (5.5)$$

indicates identical, redundant or incompatible constraints, as shown in Figure 5.2. Constraint  $k$  is identical to constraint  $i$  if  $\tilde{b}_k = 0$ . In that case, no contribution of the facet

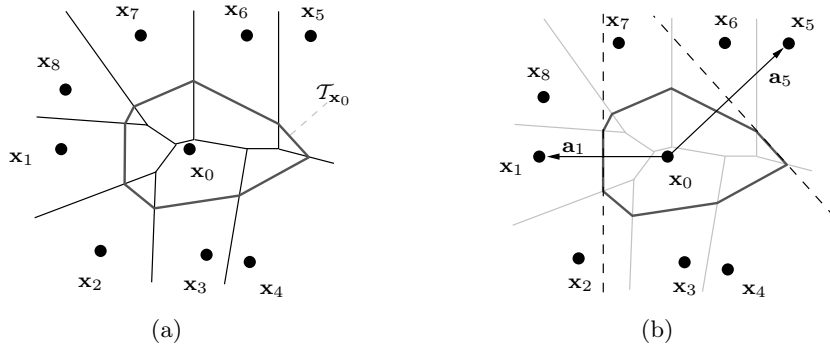


Figure 5.3: Voronoi tile  $\mathcal{T}_0$  in half-plane representation. (a) Voronoi tile  $\mathcal{T}_0$  with natural neighbors  $\mathbf{x}_1, \dots, \mathbf{x}_8$ . (b)  $\mathcal{H}_1^<$  and  $\mathcal{H}_5^<$  indicated by dashed lines and their normals  $\mathbf{a}_1$  and  $\mathbf{a}_5$ .

associated with constraint  $k$  must be computed. If  $\tilde{b}_k < 0$ , constraint  $k$  is incompatible with constraint  $i$ . In that case, the plane  $\mathbf{a}_i \mathbf{x} = \alpha b_i$  is not contained in the half-space defined by constraint  $k$  and the volume of facet  $i$  is zero. Otherwise,  $\tilde{b}_k > 0$  and constraint  $k$  is redundant. It defines a half-space that completely contains plane  $\mathbf{a}_i \mathbf{x} = \alpha b_i$ . The volume computation for facet  $k$  can then be skipped since constraint  $i$  will be in conflict with it and the volume of facet  $k$  be evaluated to zero. Identifying identical constraints is essential, while special treatment of redundant and incompatible constraints speeds up the computation by early pruning / termination of the recursion. An example for a polytope with parallel constraints is illustrated in Figure 5.2(c), which shows a Voronoi tile in 2D. Actually, this is the only point in the algorithm where numerical issues require attention.

At the bottom of the recursion,  $P(A, \mathbf{b}) \subset \mathbb{R}$  is an interval,  $A \in \mathbb{R}^{m,1}$ ,  $\mathbf{b} \in \mathbb{R}^m$ , bounded by a set of upper and / or lower bounds  $a_{i1}x_1 \leq b_i$ ,  $i = 1, \dots, m$ . If there is no upper or lower bound,  $P(A, \mathbf{b})$  is unbounded and the polytope has infinite volume. Otherwise, the interval length is fed back into (5.3).

**Remark 5.1** *As a practical sidenote, Bueler et al. showed in [BEF00] Lasserre's method has complexity of  $\mathcal{O}(n!)$  in  $n$  dimensions for constant-sized input. They also proposed a revised implementation of Lasserre's method that prunes repeated recursion based on bookkeeping, and exhibits a lower, yet still exponential complexity. Our goal being the application of Lasserre's method to natural neighbor-induced polytopes, and the average number of natural neighbors growing exponentially as well, this fact needs to be given serious consideration.*

### 5.2.3 Voronoi (Sub-)Tiles in $\mathcal{H}$ -Representation

Given a point set  $\mathbf{X} \subset \mathbb{R}^n$ , the Voronoi tile  $\mathcal{T}_0$  of a point  $\mathbf{x}_0 \in \mathbf{X}$  has an  $\mathcal{H}$ -representation which only incorporates its natural neighbors  $N(\mathbf{x}_0) = \{\mathbf{x}_1, \dots, \mathbf{x}_m\} \subset \mathbf{X}$ , as illustrated in Figure 5.3(a). The half-spaces containing  $\mathcal{T}_0$  are bounded by the bisector hyperplanes between  $\mathbf{x}_0$  and  $\mathbf{x}_1, \dots, \mathbf{x}_m$ , and are given by

$$\mathcal{H}_i^< = \{ \mathbf{x} : \mathbf{a}_i \mathbf{x} \leq b_i \}, \quad i = 1, \dots, m,$$

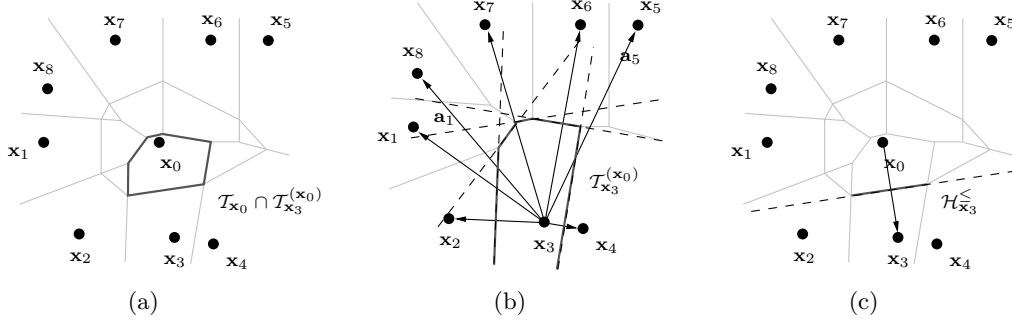


Figure 5.4: (a) Subtile  $\mathcal{T}_0 \cap \mathcal{T}_3^{(x_0)}$  in context of the other natural neighbors. (b) Constraints resulting from the natural neighbors  $N(\mathbf{x}_0) \setminus \{\mathbf{x}_3\}$ . (c) Constraint corresponding to  $\mathbf{x}_0$ .

with  $\mathbf{a}_i = \mathbf{x}_i - \mathbf{x}_0$  and  $b_i = \mathbf{a}_i^T(\mathbf{x}_0 + \mathbf{x}_i)/2$ , an example is shown in Figure 5.3(b). The Voronoi tile  $\mathcal{T}_0$  is consequently given by

$$\mathcal{T}_0 = \mathcal{H}_1^{\leq} \cap \dots \cap \mathcal{H}_m^{\leq} = \{ \mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{b} \}.$$

The Voronoi sub-tile  $\mathcal{T}_{\mathbf{x}_i}^s = \mathcal{T}_0 \cap \mathcal{T}_{\mathbf{x}_i}^{(x_0)}$  that is used in the computation of Sibson coordinates can be defined similarly. In Figure 5.4(a), a subtile for a point with eight natural neighbors is shown. The Voronoi tile of  $\mathbf{x}_i$  in  $\mathbf{X} \setminus \{\mathbf{x}_0\}$ , denoted by  $\mathcal{T}_{\mathbf{x}_i}^{(x_0)}$ , is given by the intersection of half-spaces retrieved from the natural neighbors of  $\mathbf{x}_i$  in  $\mathbf{X} \setminus \{\mathbf{x}_0\}$ , as illustrated in Figure 5.4(b). We assume the  $\mathcal{H}$ -representation  $\mathcal{T}_{\mathbf{x}_i}^{(x_0)} = \{ \mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{b} \}$ . Now, the intersection of  $\mathcal{T}_{\mathbf{x}_i}^{(x_0)}$  with  $\mathcal{T}_0$  is given by adding to  $\mathbf{A}$  and  $\mathbf{b}$  the constraint representing the bisector between  $\mathbf{x}_0$  and  $\mathbf{x}_i$ , shown in Figure 5.4(c):

$$\mathcal{T}_{\mathbf{x}_i}^s = \left\{ \mathbf{x} : \begin{bmatrix} \mathbf{A} \\ \mathbf{a}_{0i}^T \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} \mathbf{b} \\ b_{0i} \end{bmatrix} \right\}. \quad (5.6)$$

## 5.2.4 Weight Dependent Power Voronoi Tile

We now derive the half-space representation of the power Voronoi tile as a function of the power weight,

$$\mathcal{T}_0(w) := \mathcal{T}_0|_{w_i = \delta_{i0}w},$$

i.e., all power weights are zero except for  $w_0 = w$ . The tile  $\mathcal{T}_0(0)$  has  $m$  neighbors  $N(\mathbf{x}_0) = \{\mathbf{x}_1, \dots, \mathbf{x}_m\} \subset \mathbf{X}$  as defined in the classical Voronoi diagram. The bisector hyperplane corresponding to facet  $\mathcal{F}_i$  between  $\mathbf{x}_0$  and  $\mathbf{x}_i$  is the  $n - 1$ -dimensional affine space

$$\begin{aligned} \mathcal{H}_i(w) &= \{ \mathbf{x} : d^{\text{pow}}(\mathbf{x}_0, \mathbf{x}) = d^{\text{pow}}(\mathbf{x}_i, \mathbf{x}) \} \\ &= \{ \mathbf{x} : d(\mathbf{x}_0, \mathbf{x})^2 - w = d(\mathbf{x}_i, \mathbf{x})^2 - 0 \} \\ &= \{ \mathbf{x} : \mathbf{x}_0^T \mathbf{x}_0 - 2\mathbf{x}_0^T \mathbf{x} - w = \mathbf{x}_i^T \mathbf{x}_i - 2\mathbf{x}_i^T \mathbf{x} \} \\ &= \left\{ \mathbf{x} : (\mathbf{x}_i - \mathbf{x}_0)^T \mathbf{x} = \frac{1}{2}(\mathbf{x}_i^T \mathbf{x}_i - \mathbf{x}_0^T \mathbf{x}_0) + \frac{1}{2}w \right\}. \end{aligned}$$

We set  $\mathbf{a}_i := \mathbf{x}_i - \mathbf{x}_0$ ,  $c_i := 1/2$ ,  $d_i := (\mathbf{x}_i^T \mathbf{x}_i - \mathbf{x}_0^T \mathbf{x}_0)/2$ , and can write the half-space containing  $\mathcal{T}_0(w)$  as  $\{\mathbf{x} : \mathbf{a}_i^T \mathbf{x} \leq c_i w + d_i\}$ , such that with  $A = (\mathbf{a}_1, \dots, \mathbf{a}_m)^T$ , and  $\mathbf{b}(w) = \mathbf{c}w + \mathbf{d}$ ,

$$\mathcal{T}_0(w) = \bigcap_{\mathcal{F}_i \in \mathcal{T}_0(w)} \mathcal{H}_i(w) = P(A, \mathbf{b}(w)). \quad (5.7)$$

As  $w$  decreases,  $\mathcal{T}(w)$  shrinks and the set of neighbors becomes smaller, such that  $N_0|_{w_0 < 0} \subseteq N_0|_{w_0 = 0}$ . This does not invalidate the  $\mathcal{H}$ -representation of  $\mathcal{T}_0(w)$  as the half-spaces associated with the lost neighbors are redundant and can be detected during the recursive volume evaluation.

## 5.3 Laplace and Sibson Natural Neighbor Coordinates with Lasserre's method

Lasserre's method allows the computation of Laplace coordinates and Sibson coordinates in a straightforward manner. Here, we give guidelines for the implementation but remind the reader of the considerable computational complexity of that scheme for higher dimensions.

### 5.3.1 Laplace Coordinates

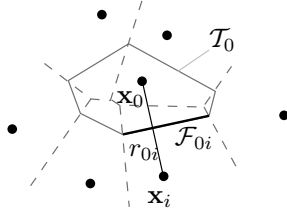


Figure 5.5: Two-dimensional facet  $\mathcal{F}_{0i}$  with the corresponding tile  $\mathcal{T}_0$  and the line segment  $r_{0i} = \|\mathbf{x}_i - \mathbf{x}_0\|$ .

For reasons of simplicity we will present the method for  $\mathbb{R}^3$ , although the generalization to  $\mathbb{R}^n$  is straightforward.

We assume a point cloud  $\mathbf{X} \subset \mathbb{R}^3$ . Without loss of generality we seek to compute Laplace coordinates for  $\mathbf{x}_0 \in \mathbf{X}$  with  $N(\mathbf{x}_0) = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ . As introduced in Section 3.2.5.2 and illustrated in Figure 5.5, homogeneous Laplace coordinates of  $\mathbf{x}_0$  with respect to  $N(\mathbf{x}_0)$  are defined by the ratios of areas of Voronoi tile facets  $\mathcal{F}_{0i} = \mathcal{T}_{\mathbf{x}_0} \cap \mathcal{T}_{\mathbf{x}_i}$  over the distance of  $\mathbf{x}_0$  to the respective neighbor,  $r_{0i} = \|\mathbf{x}_i - \mathbf{x}_0\|$ , by

$$\lambda_i^{\text{LAP}}(\mathbf{x}_0) = \frac{\hat{\lambda}_i^{\text{LAP}}(\mathbf{x}_0)}{\sum_j \hat{\lambda}_j^{\text{LAP}}(\mathbf{x}_0)}, \quad \hat{\lambda}_i^{\text{LAP}}(\mathbf{x}_0) = \frac{|\mathcal{F}_{0i}|}{r_{0i}}, \quad i = 1, \dots, m.$$

Given the  $\mathcal{H}$ -representation  $\mathcal{T}_0 = P(A, \mathbf{b})$ , the topmost level in recursion (5.3) directly gives the areas of the Voronoi tile facets  $\mathcal{F}_{01}, \dots, \mathcal{F}_{0m}$  as

$$|\mathcal{F}_{0i}| = \frac{\|\mathbf{a}_i\|}{|a_{ij}|} \text{Vol}(2, \tilde{A}^{(ij)}, \tilde{\mathbf{b}}^{(ij)}),$$

which can be easily computed by extracting the values from the appropriate level of the recursion. With  $r_{0i} = \|\mathbf{a}_i\|$ , homogeneous Laplace coordinates  $\hat{\lambda}_1^{\text{LAP}}, \dots, \hat{\lambda}_m^{\text{LAP}}$  for  $\mathbf{x}_0$  are given by

$$\hat{\lambda}_i^{\text{LAP}} = \frac{1}{|a_{ij}|} \text{Vol}(2, \tilde{A}^{(ij)}, \tilde{\mathbf{b}}^{(ij)}).$$

For the general case of Laplace coordinates in  $\mathbb{R}^n$ , we simply need to replace the term *area* by *n – 1-dimensional volume* and 2 by *n – 1* above.

### 5.3.2 Sibson Coordinates

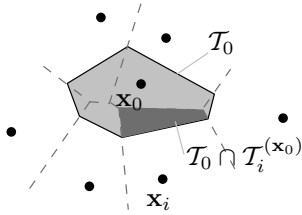


Figure 5.6: Two-dimensional subtile  $\mathcal{T}_0 \cap \mathcal{T}_i^{(\mathbf{x}_0)}$  in context.

Sibson coordinates for a point  $\mathbf{x}_0 \in \mathbf{X} \subset \mathbb{R}^n$  are defined by the ratios of subtiles  $\mathcal{T}_{\mathbf{x}_i}^s = \mathcal{T}_0 \cap \mathcal{T}_{\mathbf{x}_i}^{(\mathbf{x}_0)}$ , resulting from intersections with the tiles of the natural neighbors  $\mathbf{x}_1, \dots, \mathbf{x}_m$  of  $\mathbf{x}_0$  in  $\mathbf{X} \setminus \{\mathbf{x}_0\}$ , shown in Figure 5.6.

With the  $\mathcal{H}$ -representation  $\mathcal{T}_{\mathbf{x}_i}^s = P(A_i, \mathbf{b}_i)$  from (5.6), Sibson homogeneous coordinates  $\hat{\lambda}_i$  are given by

$$\lambda_i^{\text{SIB}} = \frac{\hat{\lambda}_i^{\text{SIB}}}{\sum_j \hat{\lambda}_j^{\text{SIB}}}, \quad \hat{\lambda}_i^{\text{SIB}} = \text{Vol}(n, A_i, \mathbf{b}_i).$$

## 5.4 Parametric Recursive Volume Representation

We now describe the above process of evaluating Lasserre’s recursion for a parametric right-hand side  $\mathbf{b}(w) = \mathbf{c}w + \mathbf{d}$ ,  $w \in \mathbb{R}$ , where the polytope  $P$  is the power Voronoi tile  $\mathcal{T}_0(w)$ . The result will express the volume of the tile as a function of  $w$ .

### 5.4.1 Recursion with Variable Right Side

After replacing the right sides of the polytope descriptions by linear functions, the recursion (5.3) for  $m$  facets now reads

$$\text{Vol}(n, A, \mathbf{b}(w)) = \frac{1}{n} \sum_{i=1, \dots, m} \frac{b_i(w)}{|a_{ij}|} \text{Vol}(n-1, \tilde{A}^{(ij)}, \tilde{\mathbf{b}}^{(ij)}(w))$$

and can be carried out similar to the process described in Section 5.2.2. Again, we assume  $i = m$  and  $j = n$ . We know that  $A$  is independent from  $w$ , and for  $1 \leq k < m$  and

$1 \leq l < n$ , (5.4) becomes

$$\tilde{a}_{kl} = a_{kl} - \frac{a_{kn}}{a_{mn}} a_{ml}, \quad \tilde{b}_k(w) = b_k(w) - \frac{a_{kn}}{a_{mn}} b_m(w).$$

### 5.4.2 Descending the Recursion

As we step down the recursion, eventually  $\tilde{\mathbf{a}}_k = \mathbf{0}$ , indicating parallel constraints  $i, k$  of the form

$$0 \leq \tilde{b}_k(w) = \tilde{c}_k w + \tilde{d}_k.$$

These cases are handled depending on whether the right hand side is below, equal to, or above zero for *any* values of  $w$ , mainly adopting the actions following (5.5). We only need to consider  $\tilde{b}_k(w)$  for  $w \leq 0$ .

$\tilde{c}_k > 0$ : There is a lower bound  $\underline{w}_{i,k} < w$  such that  $P(\tilde{A}^{(ij)}, \tilde{\mathbf{b}}^{(ij)}(w)) \neq \emptyset$ , given by

$$\underline{w}_{i,k} = -\frac{\tilde{d}_k}{\tilde{c}_k}.$$

Obviously,  $\underline{w}_{i,k} \geq 0$  implies that  $\text{Vol}(n, \tilde{A}^{(ij)}, \tilde{\mathbf{b}}^{(ij)}(w)) = 0$  for  $w \leq 0$ . Otherwise, no definite answer can be given at this point, and constraint  $k$  must be traced further down the recursion.

$\tilde{c}_k = 0$ : The right hand side is always equal to  $\tilde{d}_k$ , so

$\tilde{d}_k < 0$ :  $\text{Vol}(n-1, \tilde{A}^{(ij)}, \tilde{\mathbf{b}}^{(ij)}(w)) = 0$ ,

$\tilde{d}_k = 0$ : constraint  $k$  is equal to constraint  $i$  and must not be evaluated.

$\tilde{d}_k > 0$ : constraint  $k$  is redundant and is discarded from further recursion.

$\tilde{c}_k < 0$ : Eventually,  $\tilde{b}_k(w) > 0$  for  $w < \bar{w}_{i,k}$ , with

$$\bar{w}_{i,k} = -\frac{\tilde{d}_k}{\tilde{c}_k}.$$

If  $\bar{w}_{i,k} < \min(\underline{w}_{i,q}, 0)$  for some  $q \in \{1, \dots, m-1\}$ , the right side never becomes positive and constraint  $k$  is discarded from evaluation. Otherwise, no definite answer can be given at this point and constraint  $k$  must be traced further down the recursion.

As we follow the recursion further down, we remember the lowest occurring bound  $\underline{w}_{i,k}$ , which we use as a starting bound  $\underline{w}_{i,0}$  for the next level to decide on possible pruning of the recursion. If no such bound is present, we set  $\underline{w}_{i,0} := -\infty$ , thus effectively disabling bounds for  $w$ .

### 5.4.3 Ascending the Recursion

At the bottom of the recursion, we find for the one-dimensional facet  $P(\tilde{A}^{(ij)}, \tilde{\mathbf{b}}^{(ij)}(w))$  and a possible lower bound  $\underline{w}$  for  $w$  a set of constraints  $\tilde{a}_{k1}x \leq \tilde{b}_k(w)$ , which we can

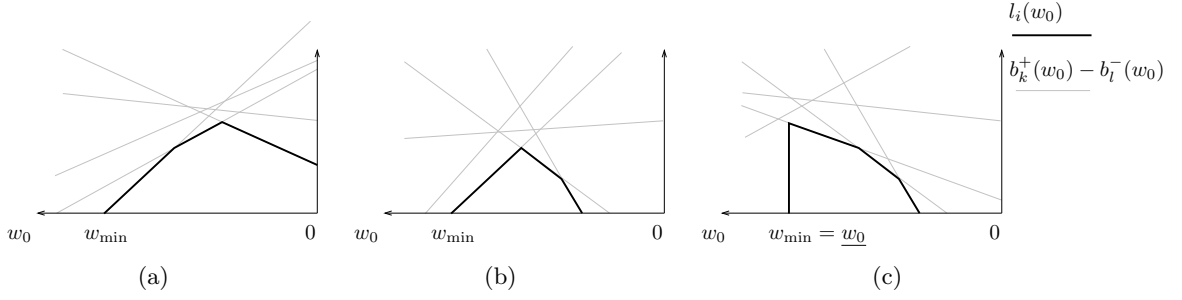


Figure 5.7: Some examples for graphs of  $l_i(w)$ , which represents the length of one of the projected edges bounding a two-dimensional Voronoi tile. We see that the convex, piecewise linear function is not necessarily monotonous, may be non-zero away from  $w = 0$ , and even have discontinuities in the presence of some lower bound  $\underline{w}$ .

classify into upper (superscript  $+$ ) and lower (superscript  $-$ ) bounds

$$x_1 \leq c_k^+ w + d_k^+ =: b_k^+(w) \quad \text{and} \quad x_1 \geq c_l^- w + d_l^- =: b_l^-(w).$$

This yields for the length  $l_i(w) := \text{Vol}(1, \tilde{A}^{(ij)}, \tilde{\mathbf{b}}^{(ij)}(w))$  of the interval,

$$l_i(w) = \begin{cases} \max\{0, \min_k\{b_k^+(w)\} - \max_l\{b_l^-(w)\}\}, & \underline{w} \leq w \leq 0, \\ 0, & \text{else} \end{cases} \quad (5.8)$$

As illustrated in Figure 5.7, each  $l_i(w)$  is a piecewise linear function over intervals  $[u_{i,g}, u_{i,g+1})$ , where the number of intervals differs from facet to facet.

The piecewise linear lengths  $l_1(w), \dots, l_m(w)$  of the edges  $e_1, \dots, e_m$  are substituted into (5.3) to compute

$$\text{Vol}(2, A, \mathbf{b}(w)) = \frac{1}{2} \sum_{i=1, \dots, m} h_i, \quad h_i = \frac{1}{|a_{ij}|} b_i(w) \text{Vol}(1, \tilde{A}^{(ij)}, \tilde{\mathbf{b}}^{(ij)}(w)).$$

The corresponding knot values denoting the interval boundaries of the piecewise functions form the monotonously growing sequence  $\mathcal{U}$ . Obviously,

$$\begin{aligned} h_i &= \frac{1}{|a_{ij}|} b_i(w) l_i(w) \\ &= \frac{1}{|a_{ij}|} \left( c_i \tilde{c}_i w^2 + (c_i \tilde{d}_i + \tilde{c}_i d_i) w + d_i \tilde{d}_i \right), \end{aligned}$$

and  $\text{Vol}(2, A, \mathbf{b}(w))$  is the sum of piecewise quadratic functions, defined over the intervals defined by  $\mathcal{U}$ . Since  $\text{Vol}(n, A, \mathbf{b}(w)) \geq 0$  must hold at any time, we check for roots of the polynomials to determine the correct intervals in which the volume is positive.

As we see, already for volumes of two-dimensional power tiles we need to compute the roots of quadratic polynomials, with  $n$ -th degree polynomials for volumes in  $\mathbb{R}^n$ . This might be feasible for volumes up to dimension three, but quickly the numerical burden of root-finding and the combinatorial complexity induced by the recursive scheme makes an application questionable.



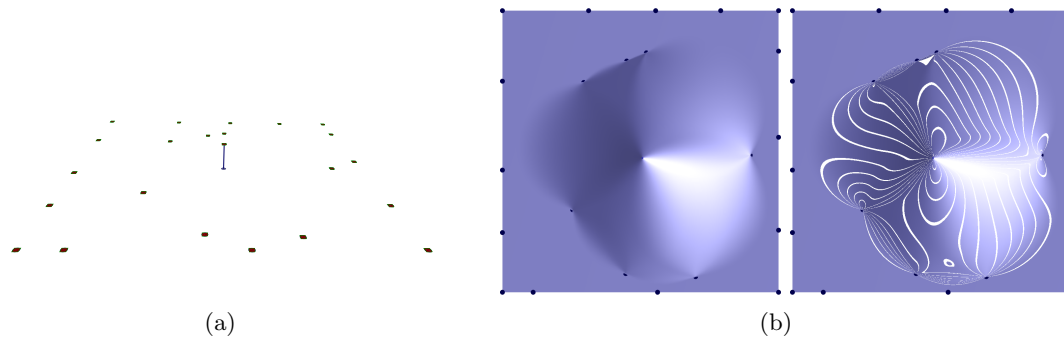


Figure 5.8: Hiyoshi's interpolant  $f^{\text{HIY}0}$ , which is  $C^2$  in  $\mathcal{C}(\mathbf{X})$  and  $C^0$  in  $\mathbf{X}$ . (a) Data set with all values set to zero except for the center point. (b) Shaded heightfield viewed from above, the right picture with reflection lines.

## 5.5 Implementation of Hiyoshi Coordinates in 2D

We implemented Hiyoshi coordinates in the plane based on the above described modification of Lasserre's method and another, straight forward, geometric approach. A heightfield representation of Hiyoshi's interpolant  $f^{\text{HIY}0}$  for the setting in Figure 5.8(a) is shown in Figure 5.8(b).

The computation of Hiyoshi coordinates can be split into two steps. The first step is the computation of homogeneous Laplace coordinates  $\hat{\lambda}_i^{\text{LAP}}(w) := \hat{\lambda}_i^{\text{LAP}}|_{w_0=w}$  in the power diagram, which is a piecewise polynomial function of  $w$ . In 2D, this is piecewise linear and the output of the first step is a sequence  $(u_j, \hat{\lambda}_i^{\text{LAP}}(u_j))_{j=0, \dots, m}$ ,  $u_0 < \dots < u_m$ , giving the control points of a piecewise linear function. We describe the computation of this sequence using a geometric approach in Section 5.5.1 and Lasserre's method in Section 5.5.2. The second step, described in Section 5.5.3 is the evaluation of the integral expression (3.5), which requires special consideration because of the piecewise structure of  $\hat{\lambda}_i^{\text{LAP}}(w)$ .

### 5.5.1 Geometric Computation

For a fix value of  $w$ , the tile  $\mathcal{T}_0(w)$  is bounded by a number of line segments connecting a number of Voronoi vertices, say  $\mathbf{v}_1, \dots, \mathbf{v}_m$ . In the CGAL library [CGA08], the retrieval of  $\mathcal{T}_0(w)$  is easy to accomplish, which facilitates a quadrature of the integral in Hiyoshi's definition over the value of  $w$ .

But we can aim for exact computation of the integral based on following observation. Figure 5.9(a) shows how the power tile shrinks with  $w$ , and reveals that the length of a tile edge is linear in  $w$  as long as its endpoints are located on the same Voronoi edges of the parent Voronoi diagram. As a consequence, the length of each tile edge  $\overline{\mathbf{v}_i, \mathbf{v}_{i+1}}$  is a piecewise linear function of  $w$ , with different slopes over intervals in an ascending knot sequence  $u_1 = \underline{w}, \dots, u_k = 0$ .

Upon close inspection of Figure 5.9(a) and (b) we see that the slope of some of the linear functions changes when a pair of vertices  $\mathbf{v}_i, \mathbf{v}_{i+1}$  merges, which happens if they meet at a vertex of the parent Voronoi diagram. This event is characterized by the bisector

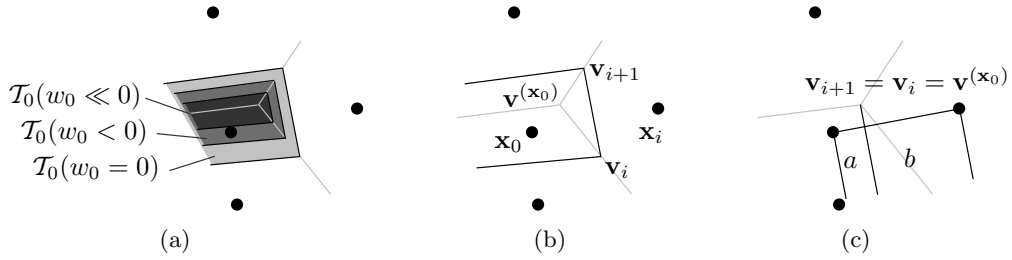


Figure 5.9: (a) Development of  $\mathcal{T}_0(w)$  as  $w$  decreases. (b) Line segment  $\overline{\mathbf{v}_i \mathbf{v}_{i+1}}$ , the bisector between  $\mathbf{x}_0$  and  $\mathbf{x}_i$ . (c) Event at which  $\mathbf{v}_i$  and  $\mathbf{v}_{i+1}$  merge based on the location of the bisector, where according to the power distance,  $a^2 - w = b^2$ .

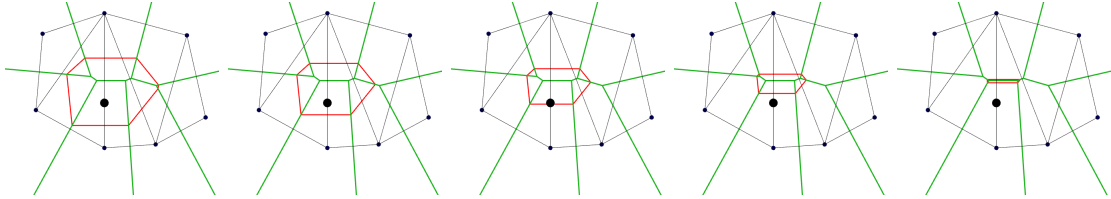


Figure 5.10: In red,  $\mathcal{T}_0(w)$  is depicted for decreasing values of  $w = 0, \dots, \underline{w}$ . In this setting,  $\mathbf{x}_0$  is located on a Delaunay edge, which results in two parallel Voronoi facets. The value of  $w$  at which the parallel facets are identical denotes a lower bound  $\underline{w}$ .

between  $\mathbf{x}_0$  and  $\mathbf{x}_i$  passing the vertex  $\mathbf{v}^{(\mathbf{x}_0)}$  of the parent Voronoi diagram, i.e., when the segment lengths  $a$  and  $b$  in Figure 5.9(c) are

$$a = (\mathbf{x}_i - \mathbf{x}_0)^T (\mathbf{v}^{(\mathbf{x}_0)} - \mathbf{x}_0) / \|\mathbf{x}_i - \mathbf{x}_0\|, \quad \text{and} \quad b = \|\mathbf{x}_i - \mathbf{x}_0\| - a.$$

The bisector has equal power distance to both  $\mathbf{x}_0$  and  $\mathbf{x}_i$ , and we get  $a^2 - w = b^2 - 0$ , providing a knot value  $a^2 - b^2$ . By collecting and sorting all knot values, one for each natural neighbor, we arrive at the desired sequence  $u_1, \dots, u_k$ .

In Figure 5.10, the evolution of  $\mathcal{T}_0(w)$  is shown for  $w = 0, \dots, \underline{w}$ . The computation of  $\hat{\lambda}_i^{\text{LAP}}(w)$  is numerically unstable. Almost always, the virtual tile vanishes in a point for  $w = \underline{w}$  and we can assume that there, the edge lengths are zero. Problems arise if  $\mathbf{x}_0$  is located on or very close to a Delaunay edge, in which case there are (almost) parallel power tile edges, and the virtual tile does not vanish in a point but in a line. To cover such cases, the slope of the first linear piece of  $\hat{\lambda}_i^{\text{LAP}}(w)$  is not computed from the Laplace coordinates at  $u_1 = \underline{w}$  and  $u_2$ , but at  $(u_1 + u_2)/2$  and  $u_2$ . This way, the geometric computation becomes very robust.

## 5.5.2 Computation with Lasserre's Method

Based on Lasserre's method, the approach described in Section 5.4 provides a parametric description of a Voronoi tile with respect to the tile's power weight. The lengths  $l_i(w)$  of the line segments that are the facets of the tile are known already after ascending one step of the recursion. This fact allows us to skip the full computation of the two-dimensional Voronoi tile. In the following we give step-by-step instructions on the implementation.

Without loss of generality, let the natural neighbors of  $\mathbf{x}_0$  be  $\{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ , and denote the edges of  $\mathcal{T}_0(w)$  by  $e_1(w), \dots, e_m(w)$ .

As shown in (5.7), we can now express the geometry of  $\mathcal{T}_0(w)$  as an algebraic set parameterized by  $w$ ,

$$\mathcal{T}_0(w) = P(A, \mathbf{b}(w)), \quad A \in \mathbb{R}^{m \times 2}, \mathbf{b}(w) \in \mathbb{R}^m, \quad (5.9)$$

with

$$\begin{aligned} \mathbf{a}_i &= \mathbf{x}_i - \mathbf{x}_0, & b_i(w) &= c_i w + d_i, \\ & & c_i &= 1/2, \\ & & d_i &= (\mathbf{x}_i^T \mathbf{x}_i - \mathbf{x}_0^T \mathbf{x}_0)/2. \end{aligned}$$

To improve numerical stability, we translate all points such that  $\mathbf{x}_0 = \mathbf{0}$ , and  $\mathbf{x}_0^T \mathbf{x}_0 = 0$  can be discarded from the above expression.

Now,  $e_i(w)$  is represented by  $\mathbf{a}_i \mathbf{x} = b_i(w)$ , which can be used to eliminate  $x_j$ ,  $j \in \{1, 2\}$ , from (5.9), yielding

$$\begin{aligned} \tilde{e}_{ij}(w) &:= P(\tilde{A}^{(ij)}, \tilde{\mathbf{b}}^{(ij)}(w)) \\ &= \left\{ x : \tilde{a}_{k1} x \leq \tilde{c}_k w + \tilde{d}_k, k = 1, \dots, m \right\}. \end{aligned} \quad (5.10)$$

We choose  $j$  such that  $|a_{ij}| = \max\{|a_{i1}|, |a_{i2}|\}$ , which avoids division by zero and admits the least round-off errors in sums. The elimination of  $x_j$  is equal to a projection along the  $j$ -th Cartesian basis vector  $\mathbf{e}_j$ , as pictured in Figure 5.1. The edge  $e_i(w)$  is shortened by a factor of  $|a_{ij}|$ , thus we get

$$l_i(w) := |e_i(w)| = |\tilde{e}_{ij}(w)|/|a_{ij}|. \quad (5.11)$$

In case there is a line segment  $e_k$ ,  $k \neq i$ , parallel to  $e_i$ , this shows up as a constraint

$$0 \leq \tilde{c}_i w + \tilde{d}_i, \quad (5.12)$$

which becomes a conflicting constraint for  $w < \tilde{d}_i/\tilde{c}_i =: \underline{w}$ , and we store  $\underline{w}$  as a lower bound for  $w$  to indicate  $\tilde{e}_{ij}(w) = \emptyset$  for  $w < \underline{w}$ . We can safely assume  $\tilde{c}_i < 0$  as this is equivalent to parallel facets for a 2D tile having opposite normals, which is true by construction. Furthermore, construction guarantees that no more than one line of  $A$  indicates a constraint parallel to the  $i$ -th constraint, and  $\underline{w} < 0$  because the set of half-planes defining  $\mathcal{T}_0(w)$  is minimal, i.e., no conflicting or multiple constraints have been set up. A situation in which parallel constraints like in (5.12) impose a lower bound  $\underline{w}$  is illustrated in Figure 5.10. If no situation as in (5.12) occurs during elimination of  $x_j$ , we set  $\underline{w} := -\infty$  to indicate the absence of a lower bound.

We classify the inequalities (5.10) as upper (superscript  $+$ ) and lower (superscript  $-$ ) bounds,

$$x \leq c_k^+ w + d_k^+ =: b_k^+(w) \quad \text{and} \quad x \geq c_l^- w + d_l^- =: b_l^-(w).$$

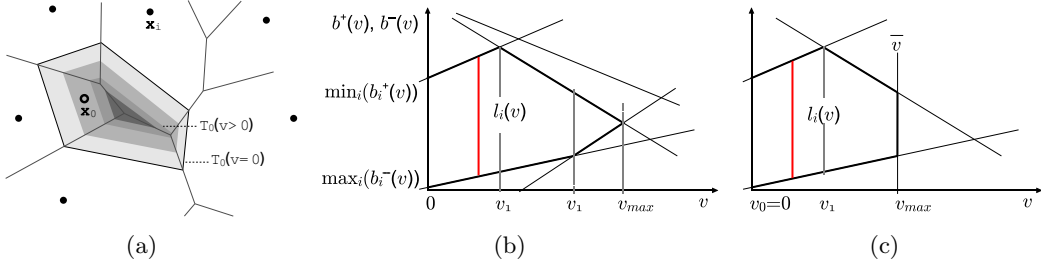


Figure 5.11: (a) shows part of the power diagram of  $\mathbf{X}^{\text{pow}}$ , the black dots showing  $\mathbf{X}$ , for uniform power weights  $w_i = 0$ ,  $i \geq 1$ , and the power tile  $\mathcal{T}_0(w)$  for different values of  $w = w_0 \leq 0$ . (b) Two examples for  $l_i(w)$  and its bounding constraints  $a_{ij}x \leq b_{i,j}(w)$ .

As illustrated in Figure 5.11(b), this yields for the length of  $\tilde{e}_{ij}(w)$ , according to (5.8),

$$|\tilde{e}_{ij}(w)| = \begin{cases} \max \{0, \min_k \{b_k^+(w)\} - \max_l \{b_l^-(w)\}\}, & 0 \leq w \leq \underline{w}, \\ 0, & \text{else.} \end{cases}$$

To determine the individual linear functions representing  $l_i(w)$  we solve a linear programming problem whose explicit solution is given in Algorithms 1-3 on page 87.

The corresponding homogeneous Laplace coordinates  $\hat{\lambda}_i^{\text{LAP}}(w) = l_i(w)/r_i$ , can be written as piecewise linear functions

$$\hat{\lambda}_i^{\text{LAP}}(w) = \begin{cases} \alpha_{ij}w + \beta_{ij} & w \in [u_j, u_{j+1}), \\ 0 & \text{else,} \end{cases}$$

with coefficients  $\alpha_{ij}, \beta_{ij} \in \mathbb{R}$ , and interval boundaries  $\underline{w} = u_1 < \dots < u_{m-1} = 0 \in \mathbb{R}$ . There are at most  $m - 1$  interval boundaries because for  $m$  neighbors, there are  $m - 2$  Voronoi vertices, each of which leads to one interval boundary, and an additional interval boundary for  $w = 0$ .

### 5.5.3 Integral Evaluation

Both the geometric and the algebraic approach result in a piecewise linear description of  $\hat{\lambda}_i^0(w)$  over intervals  $[u_j, u_{j+1})$ , where  $\alpha_{ij}$  and  $\beta_{ij}$  denote the coefficients of linear functions to describe the edge lengths  $l_i(w)$

$$\hat{\lambda}_i^{\text{LAP}}(w) = \begin{cases} \alpha_{ij}w + \beta_{ij} & w \in [u_j, u_{j+1}), \\ 0 & \text{else.} \end{cases}$$

With this, we can evaluate Hiyoshi's recursion (3.5), which reads

$$\hat{\lambda}_i^k = \hat{\lambda}_i^k(0), \quad \hat{\lambda}_i^k(w) = \int_{\underline{w}}^w \hat{\lambda}_i^{k-1}(t) dt, \quad \hat{\lambda}_i^0(w) = \hat{\lambda}_i^{\text{LAP}}(w), \quad (5.13)$$

where  $\underline{w} := \sup \{w_0 : \mathcal{T}_0(w_0) = \emptyset\}$ .

**Remark 5.2** *Hiyoshi's original exposition in [HS00b] presented the integration in (5.13) over the displacement  $w' = w/(2r_i)$  of an arbitrarily chosen bisector  $e_i(w)$  of  $\mathcal{T}_0(w)$  in the power diagram, which in fact is arc-length integration of the Voronoi tile edge length  $l_i(w)$  along the normal of  $e_i(w)$ . Since  $w'$  is the integration variable for the coordinates of all neighbors, this amounts to integration of the Laplace coordinate via the power weight, introducing a constant factor for every coordinate that cancels out in the normalization step.*

In the following, we fix  $i$  and discuss the evaluation for  $k = 2$ .

Equation (5.13) for  $C^2$  continuity reads

$$\hat{\lambda}_i^2 = \int_{\underline{w}}^0 \int_{\underline{w}}^u \hat{\lambda}_i^0(v) \, dv du. \quad (5.14)$$

Denote by  $S_{ij}(u) = \int_{\underline{w}}^u \hat{\lambda}_i^0(w) \, dw$  the inner integral of (5.14) for  $u \in [u_j, u_{j+1})$ , with

$$\begin{aligned} S_{i,-1} &:= 0, \\ S_{ij} &= S_{ij-1}(u_j) + ((u^2 - u_j^2)\alpha_{ij}/2 + (u - u_j)\beta_{ij}). \end{aligned}$$

After simple algebraic operations this gives for  $\hat{\lambda}_i^2$  the expression

$$\begin{aligned} \hat{\lambda}_i^2 &= \sum_j \left( \frac{\alpha_{ij}}{6} (u_{j+1}^3 - u_j^3) + \right. \\ &\quad \left. \frac{\beta_{ij}}{2} (u_{j+1}^2 - u_j^2) + \right. \\ &\quad \left. (S_{ij-1}(u_j) - \frac{u_j^2 \alpha_{ij} - 2u_j \beta_{ij}}{2}) (u_{j+1} - u_j) \right). \end{aligned}$$

**Remark 5.3** *In [HS00b], recursive integral reduction was applied to  $l_i(w)$ . In a practical implementation like above, however, this is not applicable due to the piecewise nature of  $l_i(w)$ .*

## 5.6 Conclusion

In this chapter we provided detailed guidelines for the implementation of Hiyoshi coordinates. They are derived from Laplace coordinates in the power diagram of a weighted point set, which was the main focus of this chapter. We first reviewed Lasserre's recursive volume computation method for polyhedra in half-space representation and how it can be applied to compute Laplace coordinates in the power diagram of a weighted point set. In contrast to the geometrical computation of Laplace and Sibson coordinates, which is straight-forward and feasible in 2D but gets increasingly complicated in higher dimensions, Lasserre's method is readily applicable unmodified in any dimension.

We then proposed a modification of Lasserre's method that allows to compute Laplace coordinates as a function of the power weight upon which Hiyoshi coordinates are defined.

One big advantage of this approach is that it only requires the Cartesian coordinates of the natural neighbors of a point. As an alternative, we devised a geometric approach to the computation of Laplace coordinates as a function of the power weight, which is considerably less involved than that based on Lasserre's method but requires functionality for the computation of Laplace coordinates in the power diagram. Based on an empiric comparison of both implementations we found that the modified Lasserre's method yields faster run times by a factor of three.

In general, the computation of natural neighbor coordinates in higher dimensions suffers from an explosion in combinatorial complexity, no matter what algorithm is actually used. This drawback also applies to the proposed computation based on Lasserre's algorithm. Furthermore, to actually compute Hiyoshi coordinates in  $\mathbb{R}^n$  using the modified Lasserre method, the roots of polynomials up to degree  $n - 1$  have to be found in the course of the method, which severely limits the applicability of this generalization for  $n > 3$ .

Finally, an argument for the application of Lasserre's approach to the computation of Sibson and Laplace coordinates in higher dimensions lies in numerical stability. The only feasible alternative that easily extends to higher dimensions is Watson's signed triangle decomposition, yet it suffers numerical instabilities arising from the ratios of infinite volumes in certain situations.

---

**Algorithm 1**  $\text{comp}(A, \mathbf{b}) : l_i(w)$ 


---

**for**  $i = 1$  to  $m$  **do**  
 move  $i$ -th constraint of  $A$  and  $\mathbf{b}$  to top in  $A'$  and  $\mathbf{b}'$   
 $\underline{w}, \tilde{A}, \tilde{\mathbf{b}} \leftarrow \text{comp}(A', \mathbf{b}')$   
 $l_i(w) \leftarrow \text{comp}(\tilde{A}, \tilde{\mathbf{b}}, \underline{w})$   
**end for**

---



---

**Algorithm 2**  $\text{comp}(A, \mathbf{b}) : \underline{w}, \tilde{A}, \tilde{\mathbf{b}}$ 


---

choose  $i, j \in \{1, 2\}, i \neq j$  such that  $|a_{1i}| \geq |a_{1j}|$ .  
 $\underline{w} \leftarrow -\infty$ .  
**for**  $k = 2$  to  $\text{rows}(A)$  **do**  
 $\gamma \leftarrow a_{ki}/a_{1i}$  // factor used to eliminate  $x_i$   
 $\tilde{a}_{k1} \leftarrow a_{kj} - \gamma a_{1j}$   
 $\tilde{b}_k(w) \leftarrow b_k(w) - \gamma b_1(w)$   
**if**  $|\tilde{a}_{k1}| \leq \varepsilon \cdot (|c_k| + |d_k| + |\gamma c_1| + |\gamma d_1|)$  **then**  
 $\underline{w} \leftarrow \max\{\underline{w}, \tilde{d}_k/\tilde{c}_k\}$   
 remove constraint  $\mathbf{a}_k \mathbf{x} \leq b_k$  from  $A$  and  $\mathbf{b}$   
 $k \leftarrow k - 1$  // do not skip the constraint following the removed one  
**end if**  
**end for**

---



---

**Algorithm 3**  $\text{comp}(\tilde{A}, \tilde{\mathbf{b}}, \underline{w}) : l(w)$ 


---

split constraints into linear functions bounding from above ( $B^+$ ) and below ( $B^-$ )  
 $B^- = \{b_j^-(w)\}_j = \left\{ \tilde{c}_i/\tilde{a}_{i1}w + \tilde{d}_i/\tilde{a}_{i1} : \tilde{a}_{i1} > 0, 1 \leq i \leq m \right\}$   
 $B^+ = \{b_j^+(w)\}_j = \left\{ \tilde{c}_i/\tilde{a}_{i1}w + \tilde{d}_i/\tilde{a}_{i1} : \tilde{a}_{i1} \leq 0, 1 \leq i \leq m \right\}$   
 reindex  $B^-$  and  $B^+$  as an ascending resp. descending sequence with respect to the following order, where  $\circ \in \{+, -\}$ :

$$b_y^\circ(w) < b_z^\circ(w) \iff \begin{cases} d_y^\circ < d_z^\circ, & |c_y^\circ - c_z^\circ| < \varepsilon(|c_y^\circ| + |c_z^\circ|), & \text{(collinearity)} \\ c_y^\circ < c_z^\circ, & \text{else.} \end{cases}$$

// The ordering allows to sequentially retrieve the strictest bounds.

$i \leftarrow 1, j \leftarrow 1, u \leftarrow \underline{w}$

increment  $i$  and/or  $j$  such that  $b_j^+(u) - b_i^-(u) > 0$  is minimal

**while**  $b_i^- \in B^-$  **and**  $b_j^+ \in B^+$  **and**  $u < 0$  **do**

find the smallest  $u' > u$  at which  $b_i^-$  and/or  $b_j^+$  are replaced by stricter bounds

$u' \leftarrow \max(0, u')$

append  $(b_j^+(v) - b_i^-(v))|_{v \in [u, u']}$  to  $l(w)$

increment  $i$  and/or  $j$  such that  $b_j^+(u) - b_i^-(u) > 0$  is minimal

$u \leftarrow u'$

**end while**

multiply  $l(w)$  by  $1/|a_{ij}|$  to account for (5.11).

---





# 6 Derivative Generation for Natural Neighbors

This chapter presents two local, natural neighbor based methods for the generation of derivatives from scattered data. The first is an ad-hoc approach applied to data collected from an iterated natural neighborhood, thereby automatically providing a spatial data site distribution to stably determine higher order derivatives. The second, which is the main contribution of this chapter, applies a combination of Sibson's original gradient fit and Akima's iterative scheme to generate higher order derivatives in natural neighbors. Derivatives generated using this method continuously depend on the coordinates of the data sites, thus being robust against perturbations in the input.

The methods have been applied to generate derivatives for synthetic data sets, and results are shown for  $C^1$  and  $C^2$  natural neighbor interpolants.

Finally, we present an observation about complexity issues in the computation of globally smooth interpolants based on Farin's projected Bézier simplex construction.

This chapter employs concepts and notation introduced in Section 2.2.1 on multivariate derivatives and Section 2.1.4 on least squares methods.

## 6.1 Background

Following the argument by Alfeld in [Alf89], we speak of derivative *generation* rather than *estimation* in the following if our only knowledge of the underlying function is the implicit assumption of smoothness, and we do not look for derivatives of a known function but rather for derivatives best fitting our assumption.

In most local scattered data interpolation methods, globally smooth interpolation is achieved by locally interpolating values and higher order derivatives at the joints of piecewise defined functions. Examples of this are the Clough-Tocher interpolant over triangles [CT65], and the globally smooth natural neighbor interpolants of Sibson ( $f^{\text{SIB}}$ ), Farin ( $f^{\text{FAR}}$ ), and Hiyoshi ( $f^{\text{HIY}^2}$ ). To use such a smooth interpolant with a data set composed only of positions and values, the derivatives need to be generated. Derivative generation can be separated into global and local methods; the most prominent of each class we mention next. Further references can be found in [Alf89].

Global derivative generation is basically done by constructing an interpolating or approximating function that globally optimizes certain objectives, and then computing the derivative of that explicit function at the data sites. The objectives allow to incorporate knowledge about the modeled physical problem, like the minimization of bending

energy for fair surface design problems. For spline surfaces it is easy to derive a quadratic approximation of the thin plate energy, which in turn allows an efficient global fitting procedure given a mostly regular data distribution. In the finite element approximation, over some tessellation of the data set, even more sophisticated problems can be modeled, drawing from vast knowledge present in the finite element community. Alfeld used such an approach in [Alf85] to generate derivatives based on the minimization of higher order derivatives, producing especially visually pleasing functional surfaces. The radial basis function approach based on infinitely smooth basis functions provides a smooth field over the whole data set and allows to generate derivatives of arbitrary order by differentiating and evaluating the basis functions. In general, the advantage of global methods lies in their implicitly *consistent* generation of derivatives over the whole data set in the sense that they mutually depend on each other to optimize some objective function.

This advantage loses importance as the size of the data set grows, since data in a sufficiently large neighborhood already describes the local behavior well enough in the vast majority of cases. Here, local derivative generation techniques allow for more efficient data processing, often giving comparable results to what global schemes can achieve. Every local scheme depends on a neighborhood from which to generate the derivatives, with some weighting function favoring closer data. As argued earlier for interpolants, a desirable property of generated derivatives is to adequately reflect small perturbations in the input data, i.e., the result should continuously depend on the input. With this in mind, it is in the determination of neighborhood and weighting function that problems arise.

A common local ad-hoc approach collects data sites in the neighborhood of a data site, and reads derivatives off a Taylor approximation of a function that minimizes the weighted squared differences to the neighboring data. If the data is inhomogeneously distributed, the neighborhood and the weighting function should vary with the data site density. In [Sib81], Sibson, generates first order derivatives for a point  $\mathbf{x}_0$  from the least squares plane through the values at its natural neighbors  $\mathbf{x}_i \in N(\mathbf{x}_0)$ . By weighting each neighboring data site  $\mathbf{x}_i \in N(\mathbf{x}_0)$  by the product of the corresponding natural neighbor coordinate  $\lambda_i$  and the inverse of its distance,  $1/\|\mathbf{x}_i - \mathbf{x}_0\|$ , he ensures that the neighborhood is always adequately adapted to the data site density, thus eliminating the need for an a priori or user-defined radius to collect neighbors from. Instead of fitting a local polynomial approximation, other methods compute a smooth interpolant to the neighborhood of a point, and evaluate the derivatives of that interpolant.

Another approach goes back to Akima in [Aki84], and is closely related to the approach we present in the following. In a given triangulation, the gradients of the piecewise linear functions over the simplices adjacent to a vertex are combined to generate the derivative at the vertex. In subsequent steps, this can be repeated to combine the Hessians of the piecewise quadratic functions in the adjacent simplices and so on. The only drawback of this method lies in the dependence on a triangulation, which may introduce discontinuous changes in the result for small perturbations in the data point configuration.

Sibson's approach for the generation of gradients is only defined for first order derivatives. We will address the problem of fitting a polynomial approximation of higher degree to the data gathered from iterated natural neighborhoods in Section 6.2. We then present

an iterative scheme combining Sibson's and Akima's approaches in Section 6.3, which requires only a very small neighborhood at a time. Results of the estimation techniques applied to synthetic data are shown in Section 6.4. We conclude this chapter with an observation concerning the run-time complexity of evaluation in multidimensional quintic Bézier simplices in Hiyoshi's  $C^2$  scheme in Section 6.5.

## 6.2 Direct Approach for Derivative Generation

We will first review an ad-hoc approach which operates on a sufficiently large neighborhood of data sites to generate the coefficients of the first  $n$  terms of the Taylor series expansion of the interpolant as defined in Section 2.2.3 at a data site.

### 6.2.1 Fitting Arbitrary Derivatives

We assume a set of data sites  $(\mathbf{x}_1, z_1), \dots, (\mathbf{x}_m, z_m)$  from the neighborhood of a data site  $\mathbf{x}_0$ , at which we want to determine the coefficients of a best-approximating Taylor polynomial of total degree  $d \geq 0$ . For a function  $f$ , we denote by  $\tilde{f}(\mathbf{x}_0 + \mathbf{x})$  its truncated Taylor series expansion at  $\mathbf{x}_i$  that has terms up to total degree  $d$ . We can without loss of generality assume  $\mathbf{x}_0 = \mathbf{0}$ , and write

$$\tilde{f}(\mathbf{0} + \mathbf{x}) = z_0 + \sum_{1 \leq |\mathbf{j}| \leq d} \frac{1}{\mathbf{j}!} \frac{d^{\mathbf{j}} f(\mathbf{0})}{d\mathbf{x}^{\mathbf{j}}} \mathbf{x}^{\mathbf{j}} \quad (6.1)$$

where we will denote by

$$c_{\mathbf{j}} := \frac{d^{\mathbf{j}} f(\mathbf{0})}{d\mathbf{x}^{\mathbf{j}}} \quad (6.2)$$

the partial derivatives of total degree  $|\mathbf{j}|$ . If  $\beta_i = 1/\|\mathbf{x}_i\|$  is the inverse distance weight that ensures stronger influence of nearby data, these partial derivatives are determined in a least squares sense as the minimizers of

$$\sum_{i=1, \dots, m} (\beta_i |\tilde{f}(\mathbf{x}_i) - z_i|)^2.$$

Collecting the coefficients of the partial derivatives in (6.1) into

$$A = (\tilde{a}_{i,\mathbf{j}}) \quad \text{with} \quad \tilde{a}_{i,\mathbf{j}} = \beta_i \frac{1}{\mathbf{j}!} \mathbf{x}_i^{\mathbf{j}}, \quad i = 1, \dots, m, \quad 0 \leq |\mathbf{j}| \leq d,$$

$$\mathbf{z} = (\beta_1 z_1, \dots, \beta_m z_m),$$

The partial derivatives  $\mathbf{c} = (c_{\mathbf{0}}, \dots)$  are the minimizers of

$$\|A\mathbf{c} - \mathbf{z}\|^2.$$

For stability reasons, it is in general advisable to have an over-constrained system of equations; otherwise the coefficients  $\mathbf{c}$  of the Taylor terms become the minimum-norm

solution to the system, which lacks a meaningful connection to the problem. This means that  $\tilde{A}$  should have full column rank and its number of rows should exceed its number of columns. It is obvious that  $\tilde{A}$  can only have full column rank if the number of partial derivatives does not exceed the cardinality  $m$  of the considered neighborhood. Using the formula for the number of multi-indices with a certain degree from (2.2), we see that the number of distinct terms in the truncated Taylor expansion of order  $d$  is

$$|\{\mathbf{j} \in \mathbb{N}^m : |\mathbf{j}| \leq d\}| = \sum_{n=0}^d \binom{n+m-1}{n},$$

and we need at least as many neighbor vertices to avoid an under-constrained fitting problem.

## 6.2.2 Choosing the Neighborhood

As Haber et al. pointed out in [HZDS01],  $A$  can have rank deficiency if the neighboring points are aligned along certain algebraic curves whose degree is correlated with the number of derivatives we want to generate. It is not feasible to determine this condition a priori, but after computing the SVD of  $A$  the occurrence of almost-zero singular values hints at such cases.

For the generation of derivatives up to degree  $d$ , we suggest as neighborhood all data sites with a topological distance smaller or equal to  $d$  in the Delaunay triangulation, which is identical to the  $d$ -times iterated natural neighborhood of  $\mathbf{x}_0$ , recursively defined as

$$N_0^d = \bigcup_{i \in N_0^{d-1}} N_i \setminus \{0\}, \quad \text{with } N_0^1 = N_0.$$

It is not obvious why this neighborhood definition should provide a sufficiently well-distributed set of neighbors to avoid small singular values for  $A$ . Let us consider as an example the setting of  $\mathbf{X} \subset \mathbb{R}^2$  with the goal to fit the Taylor series terms of derivatives up to order two. For an interior vertex of the 2D Delaunay triangulation, the average valence is six, with a minimum valence of three. Fitting the gradient requires at least two neighbors  $\mathbf{x}_a$  and  $\mathbf{x}_b$  such that  $\mathbf{x}_a - \mathbf{x}_0$  and  $\mathbf{x}_b - \mathbf{x}_0$  are linearly independent. Since  $\mathbf{x}_0$  is contained in the convex hull of its neighbors,  $N_0^1$  has at least three such points, which makes the fit of gradients stable. Now we move on to second order derivatives. We need to generate an additional three partial derivatives,  $\partial^2/\partial x_1^2$ ,  $\partial^2/(\partial x_1 \partial x_2)$ , and  $\partial^2/\partial x_2^2$ . The argument for the first set of neighbor sites also applies to their neighbors: at least two new data sites are added to the set for each  $\mathbf{x}_i \in N(\mathbf{x}_0)$ . It is impossible for all natural neighbors of a point to lie on the same algebraic curve of degree two with the point itself. This would imply that the point is not contained within the convex hull of its natural neighbors and contradict the properties of the Voronoi diagram.

### 6.2.3 Polynomial Precision for Natural Neighbor Interpolants

Because of its construction, the interpolant  $f^{\text{FAR}}$  reproduces a quadratic polynomial if  $z_i$  and  $\nabla_i$  agree with them. Consequently, by fitting quadratic polynomials, i.e., Taylor series expansion terms up to order two, and assigning  $\nabla_i$  to each site  $\mathbf{x}_i$ ,  $f^{\text{FAR}}$  reproduces quadratic polynomials.

The same is applicable for  $f^{\text{HIY}}$ , with which we can reproduce cubic polynomials if  $z_i$ ,  $\nabla_i$  and  $\mathcal{H}_i$  are extracted from a cubic polynomial fit as described in Section 6.2.2.

## 6.3 Iterative Derivative Generation

In the direct approach described above, we employed a least squares approximation in a neighborhood which we heuristically argued to be large enough to provide sufficient information for robust generation of derivatives.

Now we describe an iterative derivative generation scheme that generates the terms of the Taylor series expansions of degree  $d$  by minimizing the difference of the derivatives up to degree  $d + 1$  in a single natural neighborhood at a time. This guarantees in every stage that the neighborhood is of a structure that circumvents ill-conditioned fitting matrices. Furthermore, the inverse distance weighting that was applied above can now be augmented by natural neighbor coordinates, which gives two advantages: The spatial setting is better reflected by the weights and the resulting derivatives do continuously depend on the coordinates of the point set.

### 6.3.1 A Univariate Example

We illustrate the steps for the univariate case before giving explicit formulas for the multivariate case.

Denote by  $\tilde{f}_i^d(x) = \tilde{f}^d(x_i + x)$  the degree  $d$  Taylor polynomial centered at and associated with  $\mathbf{x}_i$ . Similar to (6.2), denote the derivatives of  $f$  at  $x_i$  by

$$c_{i,n}^d := \frac{d^n}{dx^n} f(x_i + x).$$

Then,

$$\begin{aligned} \tilde{f}^d(x_i + x) &= \tilde{f}_i^d(x) = c_{i,0}^d + c_{i,1}^d(x - x_i) + \frac{1}{2}c_{i,1}^d(x - x_i)^2 + \cdots + \frac{1}{d!}c_{i,d}^d(x - x_i)^d, \\ &= \sum_{j=0,\dots,d} \frac{1}{j!}c_{i,j}^d(x - x_i)^j. \end{aligned}$$

Assume that  $\tilde{f}_i^{d-1}$  is known and we seek coefficients of  $\tilde{f}_i^d$ . We introduce a sequence of weights  $\alpha_0, \dots, \alpha_{d-1}$  to balance the influence of derivatives of degree lower than  $d$ , where  $\alpha_0 > \cdots > \alpha_{d-1} > 0$  seems a good choice to limit the influence of higher order derivatives. These coefficients have the intuitive function of putting a stronger emphasis

on the lower-order derivatives similar to low-pass filtering in signal processing. Now, we define the residual between the prescribed derivative of order  $n$  from the last stage,  $c_{j,n}^{d-1}$ , and the  $n$ -th derivative of  $\tilde{f}^d(x_i + x_j)$  as

$$\begin{aligned}
 R_j^n(\tilde{f}_i^d) &:= \alpha_n \left( \frac{d^n}{dx^n} \tilde{f}^d(x_j) \right. && - c_{j,n}^{d-1} \left. \right) \\
 &= \alpha_n \left( \sum_{k=n}^d \frac{d^n}{dx^n} \frac{1}{k!} c_{i,k}^d (x_j - x_i)^k \right. && - c_{j,n}^{d-1} \left. \right) \\
 &= \alpha_n \left( \sum_{k=n}^d \frac{k!}{(k-n)!} \frac{1}{k!} c_{i,k}^d (x_j - x_i)^{(k-n)} \right. && - c_{j,n}^{d-1} \left. \right) \\
 &= \alpha_n \left( \sum_{k=n}^d \frac{1}{(k-n)!} c_{i,k}^d (x_j - x_i)^{(k-n)} \right. && - c_{j,n}^{d-1} \left. \right) \quad \text{set } l := k - n \\
 &= \alpha_n \left( \sum_{l=0}^{d-n} \frac{1}{l!} c_{i,n+l}^d (x_j - x_i)^l \right. && - c_{j,n}^{d-1} \left. \right). \tag{6.3}
 \end{aligned}$$

Without loss of generality, we assume  $x_i = 0$  and rewrite (6.3) as a vector product

$$\begin{aligned}
 R_j^n(\tilde{f}_i^d) &= \alpha_n \left( \overbrace{(0, \dots, 0, (x_j)^0/0!, \dots, (x_j)^{d-n}/(d-n)!)}^n \cdot (c_{i,0}^d, \dots, c_{i,d}^d)^T - c_{j,n}^{d-1} \right) \\
 &=: \alpha_n \cdot (\mathbf{r}_{j,n} \cdot \mathbf{c}_i^d)^T - c_{j,n}^{d-1}.
 \end{aligned}$$

For reasons of consistency with the multivariate case, we denote the set of neighbors of  $x_i$  by  $N(x_i)$  in spite of the fact that  $|N(x_i)| = 2$ . Now the weighted set of residuals over all derivatives up to order  $d-1$  for neighbor  $x_j$  reads

$$\begin{aligned}
 R_j(\tilde{f}_i^d) &= (R_j^0(\tilde{f}_i^d), \dots, R_j^{d-1}(\tilde{f}_i^d))^T \\
 &= \begin{bmatrix} \alpha_0 & \mathbf{r}_{j,0}^T \\ & \vdots \\ \alpha_{d-1} & \mathbf{r}_{j,d-1}^T \end{bmatrix} \begin{bmatrix} c_{i,0}^d \\ \vdots \\ c_{i,d}^d \end{bmatrix} - \begin{bmatrix} \alpha_0 & c_{j,0}^{d-1} \\ & \vdots \\ \alpha_{d-1} & c_{j,d-1}^{d-1} \end{bmatrix}.
 \end{aligned}$$

With  $\beta_{ij} := 1/|x_j - x_i|$ , we define the inverse distance weight by which nearby neighbors receive stronger consideration.

Assuming  $N(x_i)$  numbered as  $x_1, \dots, x_m$ , the concatenation of the weighted contributions of all neighbors, finally expresses the residual as a linear function of the Taylor series coefficients,

$$\begin{aligned}
 R &= \left( \beta_{i1} R_1(\tilde{f}_i^d) \dots \beta_{im} R_m(\tilde{f}_i^d) \right) \\
 &= \begin{bmatrix} \beta_{i1} \cdot \alpha_0 & \cdot \mathbf{r}_{1,0}^T \\ \vdots & \\ \beta_{i1} \cdot \alpha_{d-1} & \cdot \mathbf{r}_{1,d-1}^T \\ \vdots & \\ \beta_{im} \cdot \alpha_0 & \cdot \mathbf{r}_{m,0}^T \\ \vdots & \\ \beta_{im} \cdot \alpha_{d-1} & \cdot \mathbf{r}_{m,d-1}^T \end{bmatrix} \begin{bmatrix} c_{i,0}^d \\ \vdots \\ c_{i,d}^d \end{bmatrix} - \begin{bmatrix} \beta_{i1} \cdot \alpha_0 & \cdot c_{1,0}^{d-1} \\ \vdots & \\ \beta_{i1} \cdot \alpha_{d-1} & \cdot c_{1,d-1}^{d-1} \\ \vdots & \\ \beta_{im} \cdot \alpha_0 & \cdot c_{m,0}^{d-1} \\ \vdots & \\ \beta_{im} \cdot \alpha_{d-1} & \cdot c_{m,d-1}^{d-1} \end{bmatrix} \\
 &=: \mathbf{A} \mathbf{c}_i^d - \mathbf{c},
 \end{aligned}$$

with  $A \in \mathbb{R}^{md \times (d+1)}$ . The minimizer  $\mathbf{c}_i^d$  of

$$\sum_{j \in N_i} \|\beta_{ij} R_j(\tilde{f}_i^d)\|^2 = \|\mathbf{A} \mathbf{c}_i^d - \mathbf{c}\|^2$$

is given by

$$\mathbf{c}_i^d = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{c}.$$

The solution is unique because  $A$  has full column rank. In order to see why, note that the upper left  $d \times d$  block of  $A$ , is an upper triangle matrix with nonzero entries on the diagonal. Now consider row  $d+1$ , whose entries are monomials over a different non-zero value than any of the rows before, which proves full rank for  $A$ .

### 6.3.2 Stages One and Two

As an introduction to the multivariate case, we start by describing how to fit first and second order derivatives in two stages, from which a generalization follows. Without loss of generality, we assume  $\mathbf{x}_0 = \mathbf{0} \in \mathbb{R}^n$  and  $N_0 = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ . This way, vectors to neighbors,  $\mathbf{x}_i - \mathbf{x}_0$ , are simply written as  $\mathbf{x}_i$ . We seek to generate for  $f(\mathbf{x}_0)$  the gradient  $\mathbf{g}_0 = \nabla f(\mathbf{x}_0)$  and Hessian  $\mathbf{H}_0 = \mathcal{H} f(\mathbf{x}_0)$  of the second degree Taylor polynomials

$$f(\mathbf{x}) = z_0 + \mathbf{g}_0^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \mathbf{H}_0 \mathbf{x}. \quad (6.4)$$

The procedure for higher order derivatives then follows by analogy.

The first stage is identical to the one described by Sibson in [Sib81]. We determine the coefficients  $\mathbf{g}_0^1$  in the first-order function  $\tilde{f}^1(\mathbf{x}) = (z_0 + \mathbf{g}_0^1 \mathbf{x})$  as the minimizers to

$$\sum_{i \in N_0} \beta_i |\tilde{f}^1(\mathbf{x}_i) - z_i|^2,$$

where  $\beta_i = \lambda_i^{\text{SIB0}}/r_i^2$ . This leads to the linear least-squares system with the explicit solution

$$\mathbf{c} = A^{-1}\mathbf{b}, \quad A = \sum_{j \in N_0} \beta_j \mathbf{x}_j \mathbf{x}_j^T, \quad b_i = \sum_{j \in N_0} \beta_j (z_j - z_0) x_{ji}.$$

The above is constructed specifically such that if the function values lie on a spherical quadratic, the gradient is recovered exactly. Note that application of the Moore-Penrose inverse is viable because  $A$  is non-singular thanks to the linear independence of at least  $n$  of the vectors from  $\mathbf{x}_0$  to its natural neighbors.

If  $\mathbf{x}_0$  is a boundary point, we find that  $\mathbf{x}_0 \notin \mathcal{C}(N(\mathbf{x}_0))$ , and we do not necessarily have a full neighborhood with  $n + 1$  or more points. However, a boundary point is the vertex of at least one Delaunay simplex and as such connected to at least  $n$  natural neighbors via Delaunay edges. Being a superset of the vertices of a simplex with  $\mathbf{x}_0 = \mathbf{0}$ , the set  $N(\mathbf{x}_0)$  contains  $n$  linearly independent vectors  $\mathbf{x}_i - \mathbf{0}$ , so even in the worst case we get a well-defined gradient estimate.

Now we turn our attention to fitting the second degree Taylor polynomial from (6.4). We assume values  $z_i$  and gradients  $\mathbf{g}_i^1$  have been fitted in the first stage. Again, we consider a point  $\mathbf{x}_0 = \mathbf{0}$ , and seek the coefficients  $\mathbf{g}_0^2$  and  $\mathbf{H}_0^2$  in  $\tilde{f}^2(\mathbf{x}) = (z_0 + \mathbf{g}_0^2 \mathbf{x} + \mathbf{x}^T \mathbf{H}_0^2 \mathbf{x})$  minimizing

$$\begin{aligned} & \sum_{i \in N_0} \beta_i \left( \alpha |\tilde{f}^1(\mathbf{x}_i) - z_i|^2 + (1 - \alpha) |\nabla \tilde{f}^1(\mathbf{x}_i) - \mathbf{g}_i^1|^2 \right) \\ &= \sum_{i \in N_0} \beta_i \left( \alpha |\mathbf{x}_i^T \mathbf{H}_0^2 \mathbf{x}_i + \mathbf{g}_0^2 \mathbf{x}_i + z_0 - z_i|^2 + (1 - \alpha) |2\mathbf{H}_0^2 \mathbf{x}_i + \mathbf{g}_0^2 - \mathbf{g}_i^1|^2 \right), \end{aligned}$$

where  $\alpha \in [0, 1]$  allows to choose a preference between matching the absolute values or rather agreeing with the derivatives in the neighborhood. We see that already in the quadratic setting, it is beneficial to switch to multi-index notation to describe the algorithm.

### 6.3.3 An Explicit Construction for the General Setting

We now give explicit formulas for the iterative fitting of Taylor terms up to order  $d$  in data sets  $\{(\mathbf{x}_i, z_i)\}_i \subset \mathbb{R}^n \times \mathbb{R}$ , following closely the presentation for the univariate case.

Following the convention from Section 6.2.1, we denote the derivatives in the terms of the Taylor series expansion by

$$c_{i,\mathbf{j}} := d^{|\mathbf{j}|} f(\mathbf{x}_i) / d\mathbf{x}^{\mathbf{j}}.$$

Assume we have already generated the truncated Taylor series expansion  $\tilde{f}_j^{d-1}$  at every point in  $\mathbf{X}$  and now want to fit the Taylor terms up to degree  $d$  in point  $\mathbf{x}_i$ ,

$$\tilde{f}^d(\mathbf{x}_i + \mathbf{x}) = \sum_{|\mathbf{j}| \leq d} \frac{\mathbf{x}^{\mathbf{j}}}{\mathbf{j}!} c_{i,\mathbf{j}}^d.$$



We seek to determine  $\mathbf{c}_i^d$  as the minimizer of

$$\sum_{j \in N_i, |\mathbf{k}| \leq d-1} \left| \beta_j R_j^{\mathbf{k}}(\tilde{f}_i^d) \right|^2,$$

where  $R_j^{\mathbf{k}}(\tilde{f}_i^d)$  is the difference between the  $\mathbf{k}$ -th derivative of  $\tilde{f}_i^d$  and prescribed value  $c_{j,\mathbf{k}}^{d-1}$  at neighbor  $j$  (cf. Section 2.2.1). We introduce the convention that  $\mathbf{d}$  may denote the largest multi-index with norm  $|\mathbf{d}| = d$  such that

$$\mathbf{i} \leq \mathbf{j} \leq \mathbf{d}$$

enumerates all multi-indices  $\mathbf{j} : |\mathbf{j} + \mathbf{i}| \leq d$ . In particular, with  $|\mathbf{k}| = n$ , we get

$$\begin{aligned} R_j^{\mathbf{k}}(\tilde{f}_i^d) &= \alpha_n \left( \frac{d^n}{d\mathbf{x}^{\mathbf{k}}} \tilde{f}_i^d(\mathbf{x}_j) - c_{i,\mathbf{k}}^{d-1} \right) \\ &= \alpha_n \left( \sum_{\mathbf{k} \leq \mathbf{j} \leq \mathbf{d}} \frac{1}{(\mathbf{j} - \mathbf{k})!} (\mathbf{x}_j - \mathbf{x}_i)^{\mathbf{j} - \mathbf{k}} c_{i,\mathbf{j}}^d - c_{j,\mathbf{k}}^{d-1} \right). \end{aligned}$$

In the above sum, nonzero terms occur for  $\mathbf{j} \geq \mathbf{k}$ . If we assume  $\mathbf{x}_i = \mathbf{0}$ , which can be met by simple translation of  $\mathbf{X}$ , we can conveniently encode the above sum as the product

$$\begin{aligned} R_j^{\mathbf{k}}(\tilde{f}_i^d) &= \alpha_n \left( \underbrace{(0, \dots, 0, \mathbf{x}_j^{\mathbf{j}} / (\mathbf{j} - \mathbf{k})!, \dots)}_{\mathbf{j} < \mathbf{k}} \underbrace{(\dots)}_{\mathbf{j} \geq \mathbf{k}, |\mathbf{j}| \leq d} \underbrace{(c_{i,\mathbf{0}}^d, \dots, c_{i,\mathbf{j}}^d, \dots)}_{|\mathbf{j}| \leq d} \right)^T - c_{j,\mathbf{k}}^{d-1} \\ &=: \alpha_n (\mathbf{r}_{j,\mathbf{k}} \cdot \mathbf{c}_i^d - c_{j,\mathbf{k}}^{d-1}). \end{aligned}$$

The residual can consequently be expressed as

$$\begin{aligned} R &= \begin{bmatrix} \beta_{i1} \cdot \alpha_0 \cdot \mathbf{r}_{1,\mathbf{0}}^T \\ \vdots_{|\mathbf{j}| \leq d-1} \\ \beta_{i1} \cdot \alpha_{d-1} \cdot \mathbf{r}_{1,\mathbf{d}-1}^T \\ \vdots \\ \beta_{im} \cdot \alpha_0 \cdot \mathbf{r}_{m,\mathbf{0}}^T \\ \vdots_{|\mathbf{j}| \leq d-1} \\ \beta_{im} \cdot \alpha_{d-1} \cdot \mathbf{r}_{m,\mathbf{d}-1}^T \end{bmatrix} \begin{bmatrix} c_{i,\mathbf{0}}^d \\ \vdots \\ c_{i,\mathbf{d}}^d \end{bmatrix} - \begin{bmatrix} \beta_{i1} \cdot \alpha_0 \cdot c_{1,\mathbf{0}}^{d-1} \\ \vdots \\ \beta_{i1} \cdot \alpha_{d-1} \cdot c_{1,\mathbf{d}-1}^{d-1} \\ \vdots \\ \beta_{im} \cdot \alpha_0 \cdot c_{m,\mathbf{0}}^{d-1} \\ \vdots \\ \beta_{im} \cdot \alpha_{d-1} \cdot c_{m,\mathbf{d}-1}^{d-1} \end{bmatrix} \\ &=: \mathbf{A} \mathbf{c}_i^d - \mathbf{c}. \end{aligned}$$

Now, we find  $\mathbf{c}_i^d$  minimizing

$$\|\mathbf{A} \mathbf{c}_i^d - \mathbf{c}\|^2$$

as the solution  $(A^T A)^{-1} A^T \mathbf{c}$  of an over-constrained least squares system. We omit the proof that  $A$  is non-singular and note that similar arguments apply as in the univariate case.

### **6.3.4 Discussion**

It must be noted that the iterative fitting procedure is unable to infer exact derivatives from data sampled from any superlinear polynomials. This is due to the fact that after stage one, the inferred gradients are exact only for linear functions but inexact for general polynomials. Fitting higher order terms in subsequent stages is therefore based on inexact information.

An advantage lies, however, in the iterative nature that allows for a multi-pass processing of data sets where every pass requires the same, simple neighborhood structure. Furthermore, the localization of weights by natural neighbor coordinates guarantees a continuous change of the gradients when the coordinates of the point cloud change.

The supporting subset of data sites from which Taylor terms up to order  $d$  are generated are actually identical for the direct approach with the proposed neighborhood selection and the  $d$ -times iterated derivative generation. This becomes obvious after comparing the recursive definition of  $d$ -times iterated natural neighborhood with the set of data sites that contributes to the derivative in every stage of the iterative scheme.

## 6.4 Results: Generated Derivatives Applied in Smooth Natural Neighbor Interpolation

We applied Farin’s  $C^1$  smooth and Hiyoshi’s  $C^2$  smooth interpolant to derivative information generated with the discussed methods, with data sets of increasing sampling density. The resulting interpolants are visually presented in Section 6.4.1 and compared numerically in Section 6.4.2.

As reference serves Franke’s function, an analytic function given by

$$f(x, y) = \frac{3}{4} \exp\left(-\frac{(9x-2)^2 + (9y-2)^2}{4}\right) + \frac{3}{4} \exp\left(-\frac{(9x+1)^2}{49} - \frac{9y+1}{10}\right) \\ + \frac{1}{2} \exp\left(-\frac{(9x-7)^2 + (9y-3)^2}{4}\right) - \frac{1}{5} \exp(-(9x-4)^2 - (9y-7)^2),$$

where  $\exp(x) = e^x$ . It was introduced by Franke in [FN80] and has since been used widely as a benchmark, although comparison to other methods using this benchmark is difficult since the distribution of data sites varies across individual assessments.

We refer to the interpolation input data as  $Z \in \mathbb{R} \times \mathbb{R}^2 \times \mathbb{R}^{2 \times 2}$ , i.e., tuples of values, gradients and Hessians for data sites in  $\mathbb{R}^2$ . Gradients and Hessians directly sampled from this function serve as reference data to provide an optimal basis for the interpolation schemes and are denoted by  $Z^{\text{ref}}$ . We use the following abbreviations to refer to certain direct derivative generation methods:

$Z^{\text{GN1}}$  derivatives up to order one generated using Sibson’s gradient fit from [Sib81].

$Z^{\text{HN1}}$  derivatives up to order two stemming from fitting a quadratic Taylor polynomial to the one-ring neighborhood of each data site.

$Z^{\text{HN2}}$  derivatives up to order two stemming from fitting a quadratic Taylor polynomial to the two-ring neighborhood of each data site.

$Z^{\text{HN3}}$  derivatives up to order two stemming from fitting a cubic Taylor polynomial to the three-ring neighborhood of each data site.

The iterated derivative generation has been implemented as the described two-step procedure for quadratic Taylor polynomials. The user parameter that controls the influence lower order derivatives in subsequent fitting steps has been assigned the values  $\alpha \in \{0.1, 0.01, 0.001, 0.0001\}$ ; the corresponding input data is referred to by  $Z^{\text{HI}\alpha}$ .

Only gradients are considered when interpolating with Farin’s  $C^1$  approach, and Hessians as well when using Hiyoshi’s  $C^2$  approach.

### 6.4.1 Visual Examples for Franke’s Function

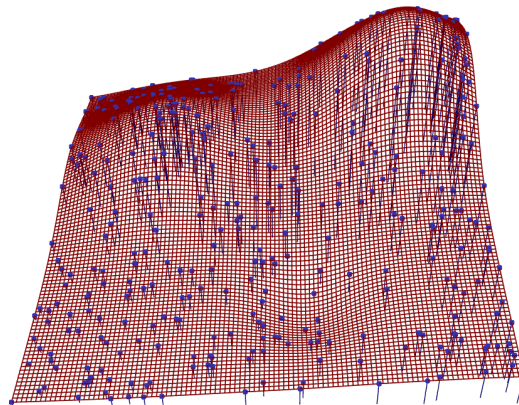
Figure 6.1 shows three different samplings of Franke’s function which have been generated by gradually adding new sample sites. To illustrate the good approximation property of natural neighbor interpolation in general, Figure 6.2 shows Farin’s and Hiyoshi’s interpolants under increasing sampling density with exact derivative data.

The derivatives produced by the iterated scheme result in interpolants as shown in Figure 6.3. Obviously, decreasing the influence  $\alpha$  of lower order derivatives has the effect of smoothing the result, which corresponds to stabilizing the often underdetermined second order fit in the one-ring neighborhood of a data site. However, close inspection revealed that locally, the results may worsen again if  $\alpha$  is decreased further, making it difficult to provide some reasonable default.

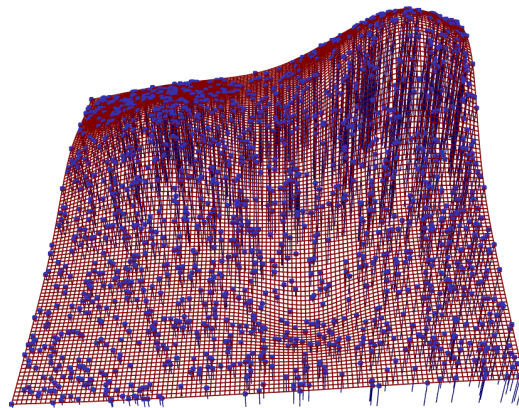
In Figures 6.4 and 6.5 the effect of fitting Taylor approximations of varying degree is shown. It is apparent in Figures 6.4(b) and 6.5(b) that fitting in an underdetermined or badly conditioned neighborhood produces bad results.

Interestingly, the results are worse if gradients are taken out of a second degree Taylor fit than for gradients directly determined by a first degree Taylor fit, as can be seen in Figures 6.4(a) and 6.4(c) for  $C^1$ -smooth interpolation and in Figures 6.5(a) and 6.5(c) for  $C^2$ -smooth interpolation. However, when fitting third degree Taylor approximations to the three-ring neighborhood of each data site to determine gradients and Hessians, a considerable improvement can be observed both for  $C^1$  and  $C^2$  interpolation in Figures 6.4(d) and 6.5(d).

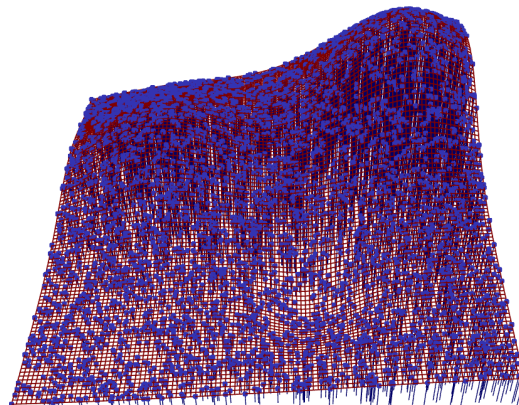
It can be observed that the shape of the interpolant has more oscillations for  $C^2$  interpolation with Hessians that do not agree with the reference data than for  $C^1$  interpolation merely operating on gradients. This is indicated by the increased wiggles in the reflection lines in the right columns of Figure 6.7 and Figure 6.7.



(a) 356 sites, mean edge length 0.0601

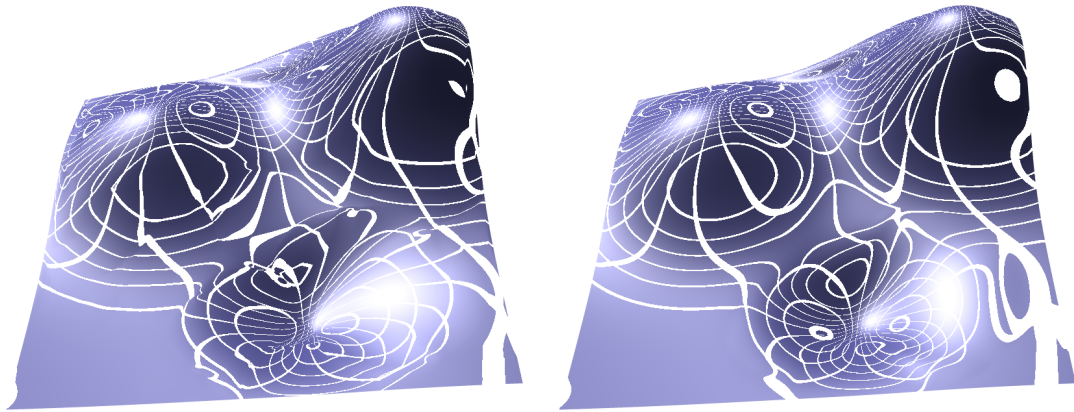


(b) 1536 sites, mean edge length 0.0284



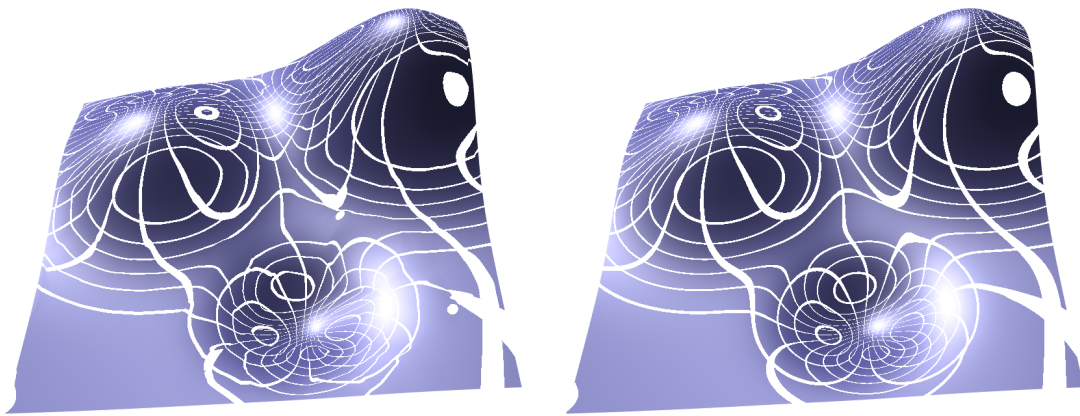
(c) 6642 sites, mean edge length 0.0134

Figure 6.1: Franke's function subsampled at different resolutions.



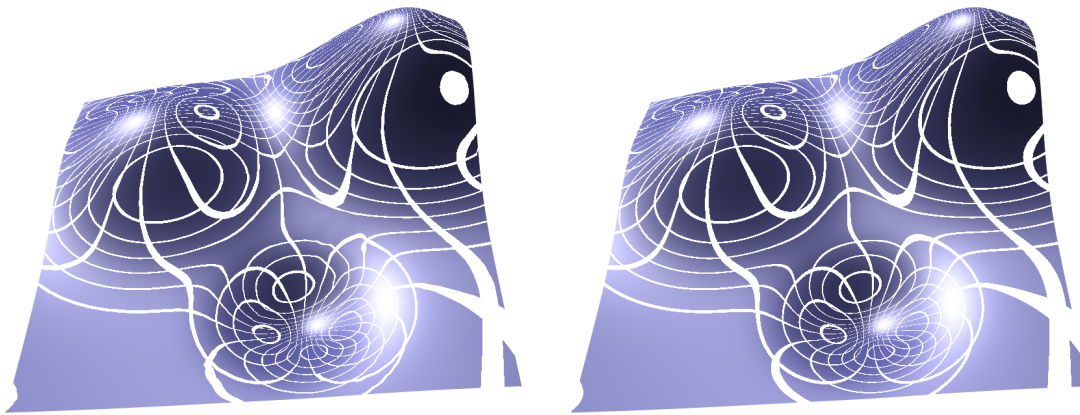
(a) Farin's  $C^1$  on  $Z^{\text{ref}}$ , 111 sites

(b) Hiyoshi's  $C^2$  on  $Z^{\text{ref}}$ , 111 sites



(c) Farin's  $C^1$  on  $Z^{\text{ref}}$ , 356 sites

(d) Hiyoshi's  $C^2$  on  $Z^{\text{ref}}$ , 356 sites



(e) Farin's  $C^1$  on  $Z^{\text{ref}}$ , 1536 sites

(f) Hiyoshi's  $C^2$  on  $Z^{\text{ref}}$ , 1536 sites

Figure 6.2: Farin's and Hiyoshi's interpolants applied to analytic derivatives sampled from Franke's function. With raising sampling density, the wiggles in the reflection lines vanish, indicating good approximation property of smooth natural neighbor interpolation.

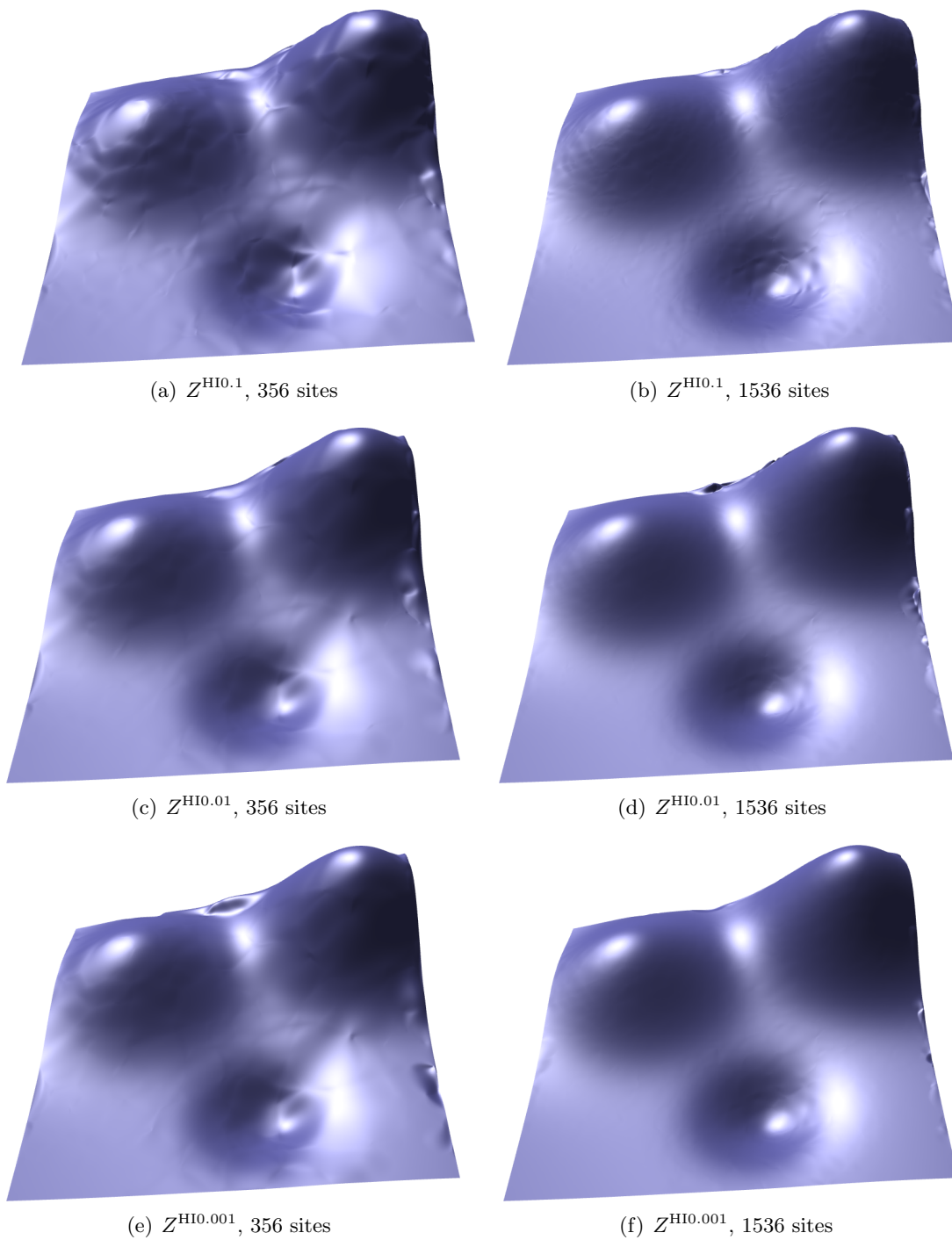


Figure 6.3: The influence of  $\alpha$  on the iterated derivative generation by means of Hiyoshi's  $C^2$  interpolant. Left to right: increase of data set density. Top to bottom: decreasing  $\alpha$ .



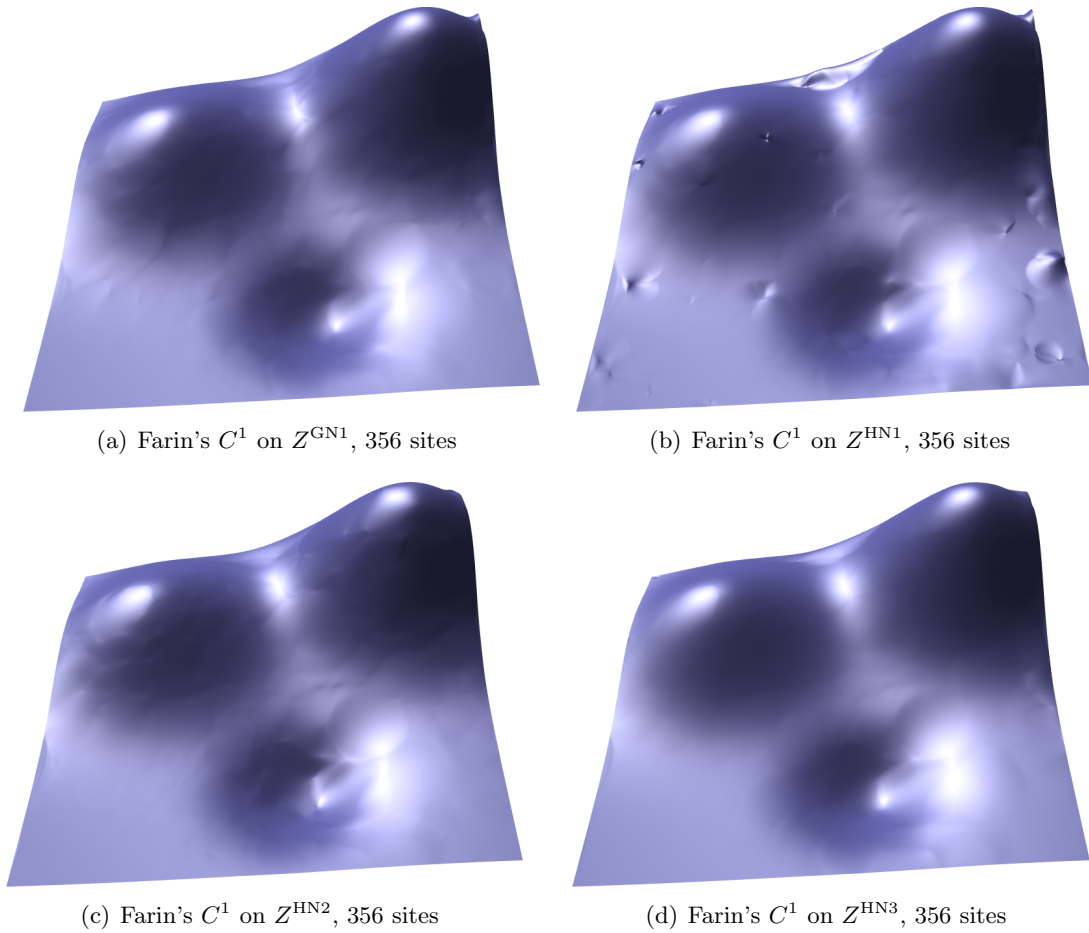
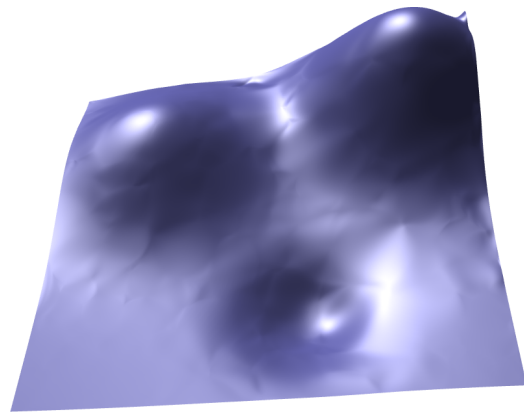
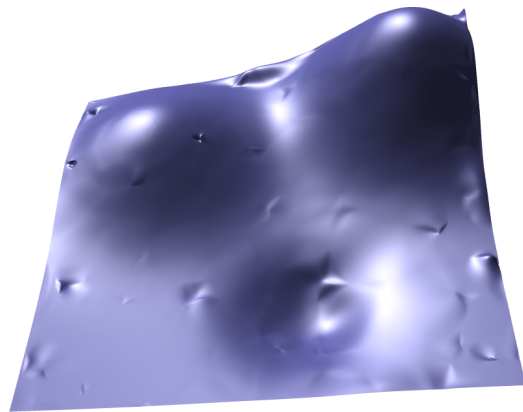


Figure 6.4: The images show Farin's interpolant applied to data with gradients. (a) Using Sibson's method. (b) Using the first order part of a second order Taylor polynomial fitted to the one-ring neighborhood of each site. (c) Like (b), but fitted to the two-ring neighborhood. (d) Using the first order part of a third order Taylor polynomial fitted to the three-ring neighborhood of each site.

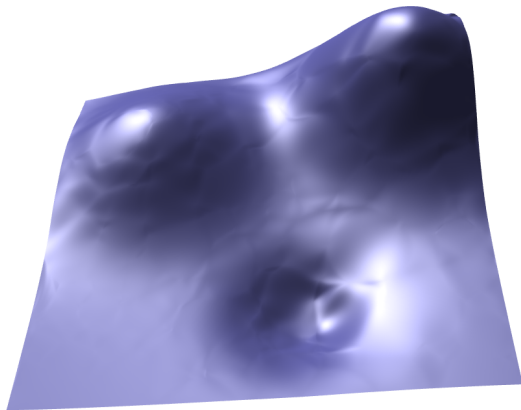




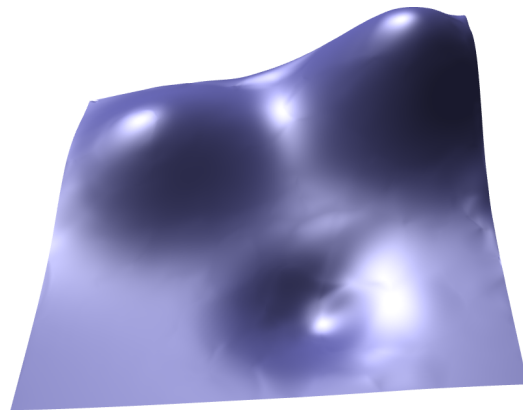
(a) Hiyoshi's  $C^2$  on  $Z^{\text{GN}1}$ , 356 sites



(b) Hiyoshi's  $C^2$  on  $Z^{\text{HN}1}$ , 356 sites



(c) Hiyoshi's  $C^2$  on  $Z^{\text{HN}2}$ , 356 sites



(d) Hiyoshi's  $C^2$  on  $Z^{\text{HN}3}$ , 356 sites

Figure 6.5: The images show Hiyoshi's interpolant applied to data with gradients and Hessians. (a) Using Sibson's method, where the second order part is set to zero. (b) Second order Taylor polynomial fitted to the one-ring neighborhood of each site. (c) Like (b), but fitted to the two-ring neighborhood. (d) Using the first and second order part of a third order Taylor polynomial fitted to the three-ring neighborhood of each site.

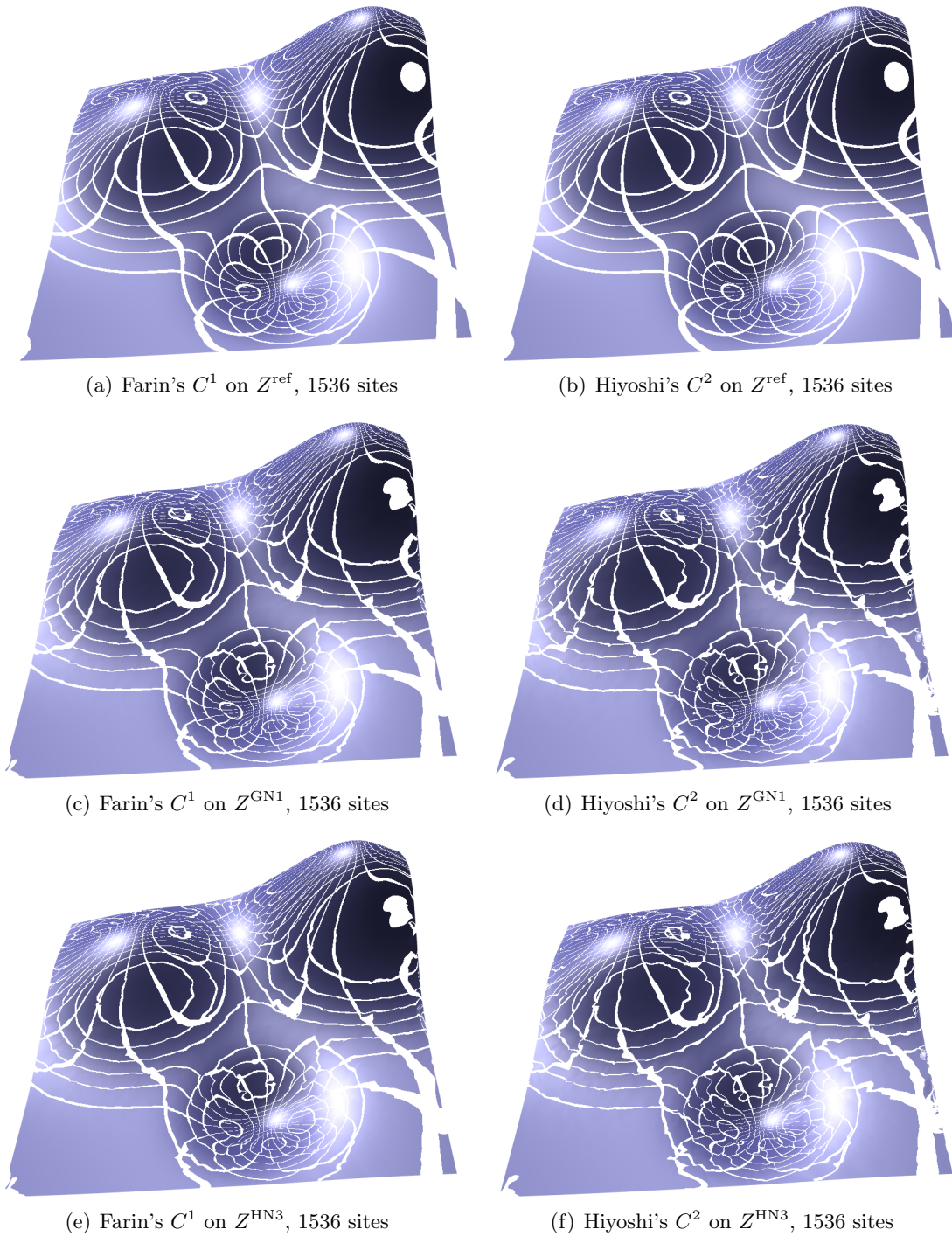
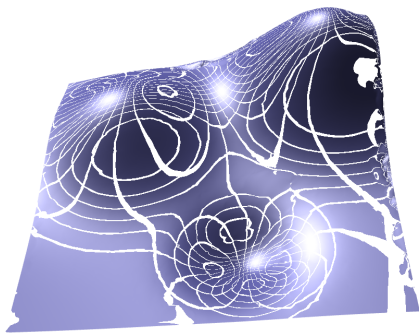
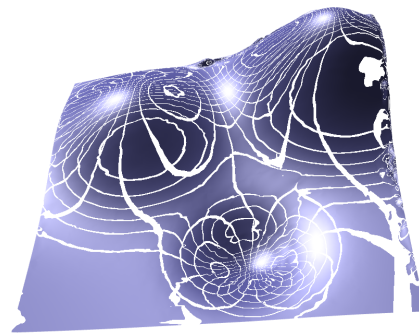


Figure 6.6: (Left) Farin's  $C^1$  interpolant. (Right) Hiyoshi's  $C^2$  interpolant. Interpolants computed from (top) reference data, (middle) first-order functions fitted to the one-ring neighborhood of each data site, (bottom) gradients and Hessians agreeing with third-order functions fitted to the three-ring neighborhood of each data site.

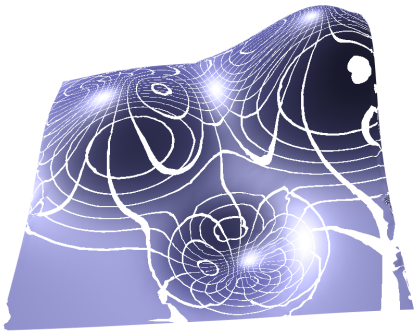
6.4 Results: Generated Derivatives Applied in Smooth Natural Neighbor Interpolation



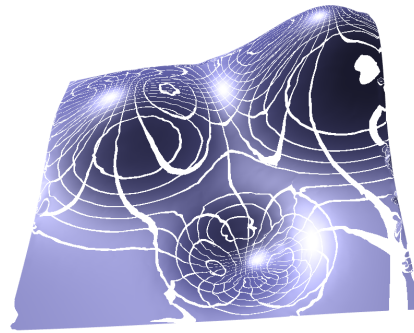
(a) Farin's  $C^1$  on  $Z^{\text{HI}0.01}$ , 1536 sites



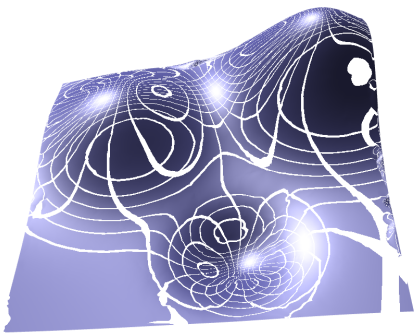
(b) Hiyoshi's  $C^2$  on  $Z^{\text{HI}0.01}$ , 1536 sites



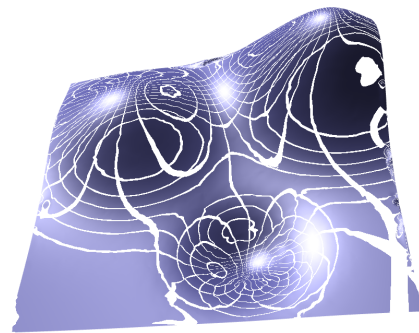
(c) Farin's  $C^1$  on  $Z^{\text{HI}0.001}$ , 1536 sites



(d) Hiyoshi's  $C^2$  on  $Z^{\text{HI}0.001}$ , 1536 sites



(e) Farin's  $C^1$  on  $Z^{\text{HI}0.0001}$ , 1536 sites



(f) Hiyoshi's  $C^2$  on  $Z^{\text{HI}0.0001}$ , 1536 sites

Figure 6.7: (Left) Farin's  $C^1$  interpolant. (Right) Hiyoshi's  $C^2$  interpolant. Interpolants computed from iterated fits to the one-ring neighborhood, using weighting parameters of (top)  $\alpha = 0.01$ , (middle)  $\alpha = 0.001$ , (bottom)  $\alpha = 0.0001$ .

### 6.4.2 Numerical Assessment for Franke's Function

In this section we present a numerical assessment of the different derivative generation methods, based on the root mean square error (RMSE) between interpolants and the ground truth given by Franke's function. To empirically analyze the convergence of the resulting interpolants under refinement, a series of data sets with increasing density was generated. The mean edge lengths of subsequent data sets have a ratio of 0.9, starting at 0.116 and going down to 0.009. The numerical assessment has been done by densely sampling the domain over a grid of 800 by 800 points, in which the difference to the reference data set in values, gradients and Hessians has been computed. Thus, the discretization from which the numerical assessment was computed still provides an oversampling of roughly ten for the finest data set. All assessments have been done using Farin's  $C^1$  and Hiyoshi's  $C^2$  smooth interpolant.

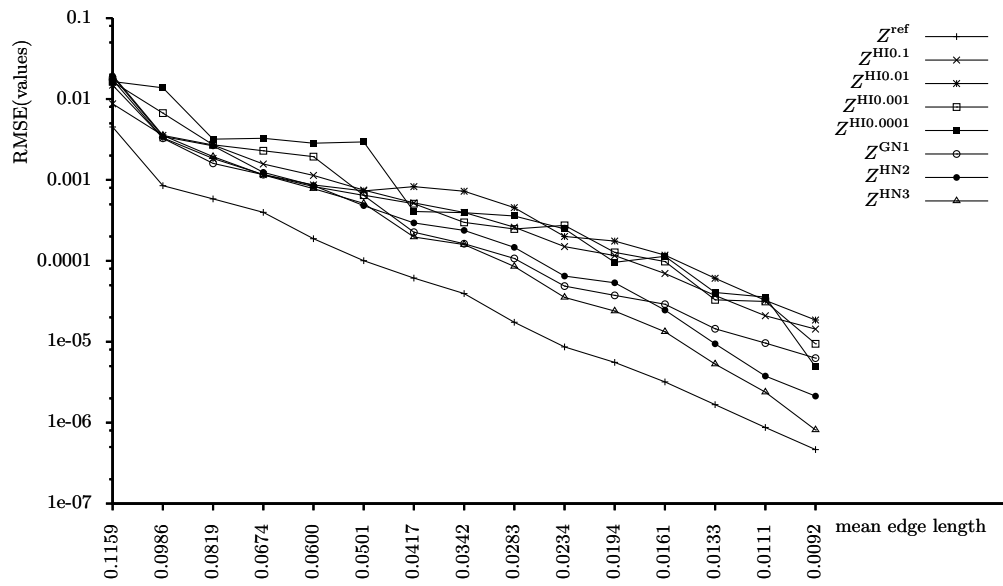
The first thing to notice about all graphs is that the derivative data  $Z^{\text{ref}}$  sampled from Franke's function leads to interpolants that are in every aspect significantly closer to the reference than any of the proposed derivative generation schemes.

The RMSE of values converges to zero for every derivative generation scheme, with no significant difference between Farin's  $C^1$  interpolant and Hiyoshi's  $C^2$  interpolant despite the fact that the latter also interpolates Hessians. In contrast, the reference data set leads to significantly faster convergence with Hiyoshi's  $C^2$  interpolant. Figure 6.8 indicates that the methods that directly fit derivatives to a neighborhood of adequate size converge faster than the iterated derivative generation.

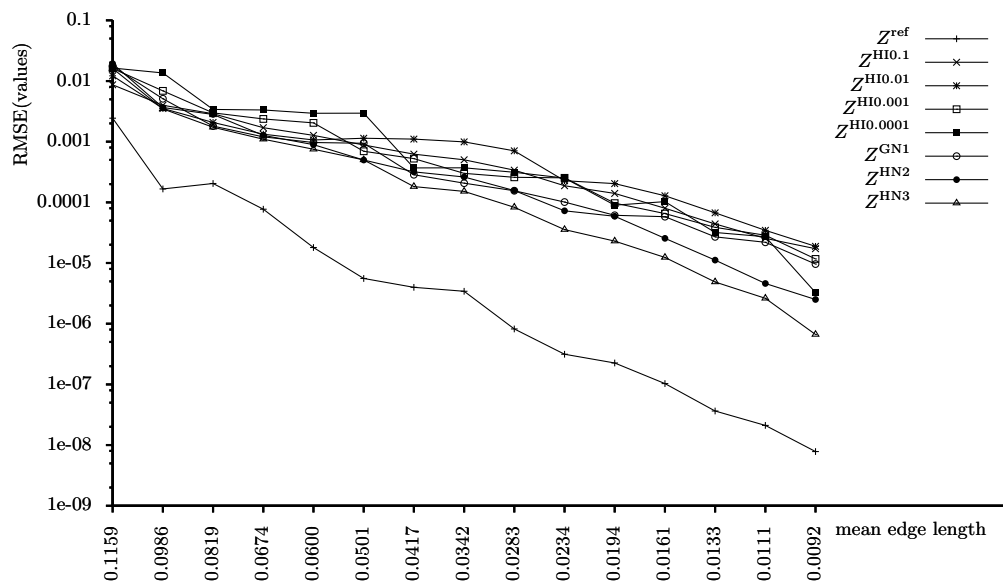
A similar result is obtained for the RMSE of gradients in Figure 6.9. The reference data set leads to faster convergence with Hiyoshi's  $C^2$  method, while for generated derivatives the choice of interpolation scheme is unimportant. Derivatives generated by direct fits to large neighborhoods lead to faster convergence.

The main difference of the results on the RMSE of Hessians from those on values and gradients is the almost unnoticeable convergence for iterated derivative generation, as depicted in Figure 6.10.

6.4 Results: Generated Derivatives Applied in Smooth Natural Neighbor Interpolation

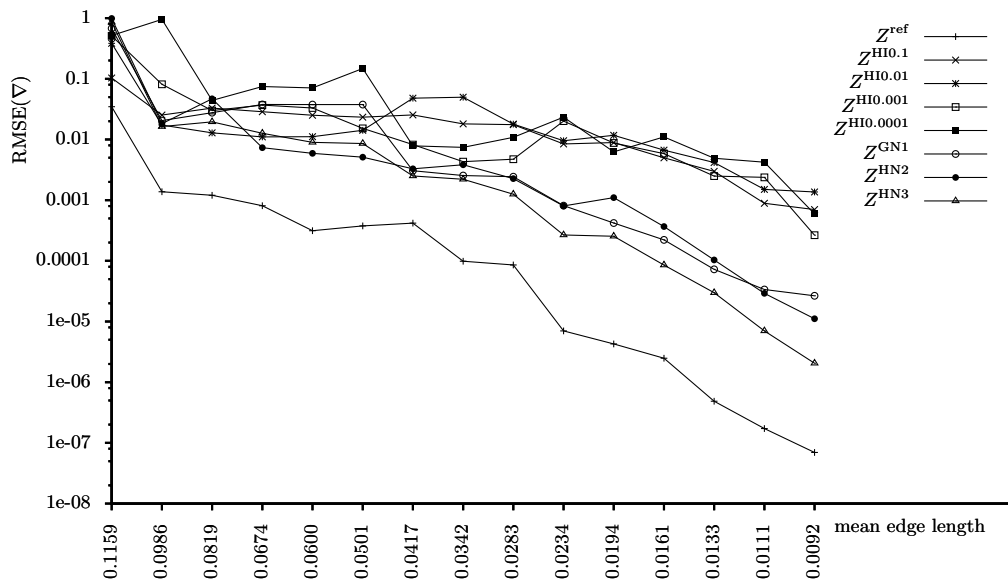


(a)

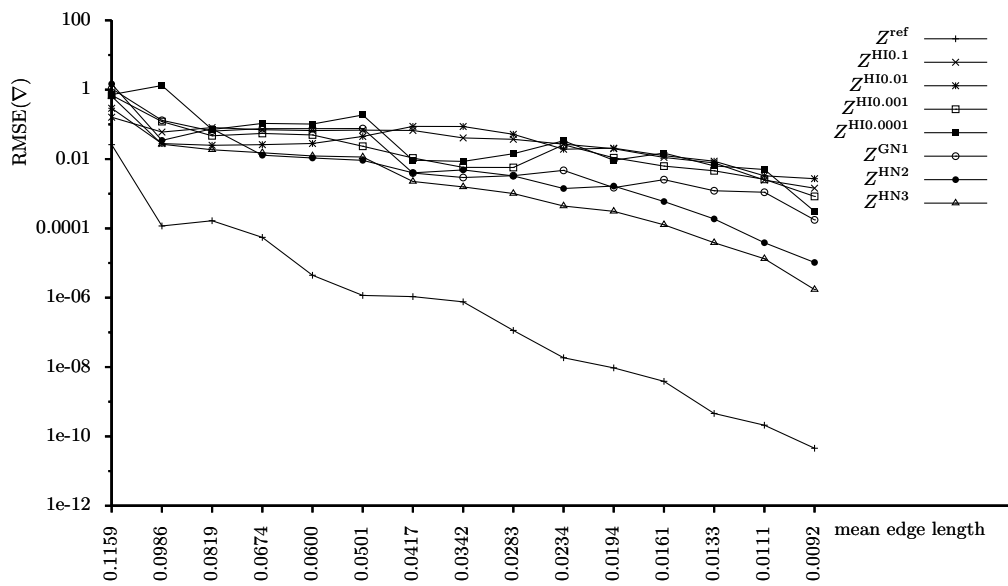


(b)

Figure 6.8: Root Mean Square Error (RSME) between the values of the interpolants and the reference function. (a) Farin's  $C^1$  interpolant based on gradients at the data sites. (b) Hiyoshi's  $C^2$  interpolant based on gradients and Hessians at the data sites.



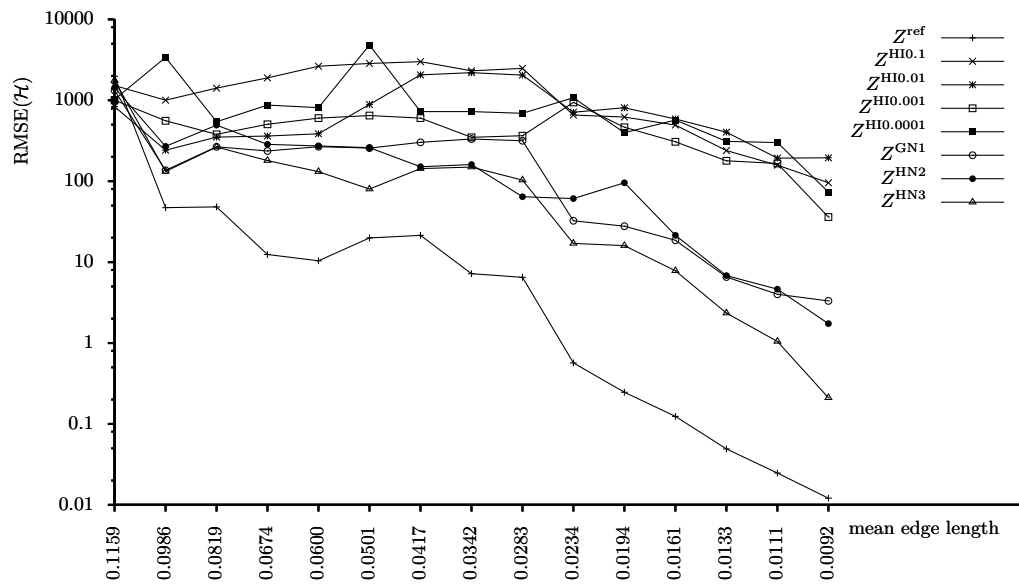
(a)



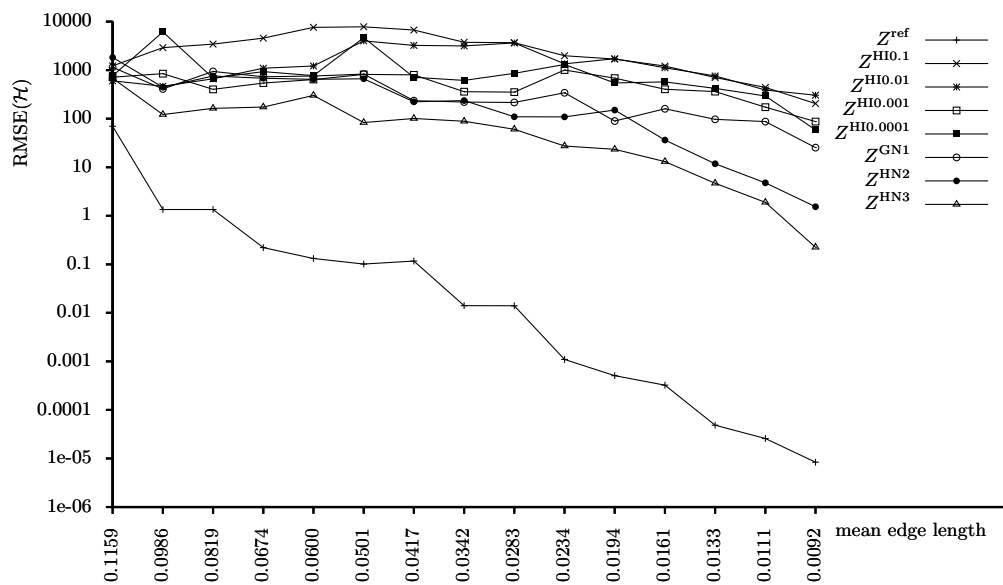
(b)

Figure 6.9: Root Mean Square Error (RSME) between the gradients of the interpolants and the reference function. (a) Farin's  $C^1$  interpolant based on gradients at the data sites. (b) Hiyoshi's  $C^2$  interpolant based on gradients and Hessians at the data sites.

6.4 Results: Generated Derivatives Applied in Smooth Natural Neighbor Interpolation



(a)



(b)

Figure 6.10: Root Mean Square Error (RSME) between the Hessians of the interpolants and the reference function. (a) Farin's  $C^1$  interpolant based on gradients at the data sites. (b) Hiyoshi's  $C^2$  interpolant based on gradients and Hessians at the data sites.

## 6.5 Complexity Issues for Quintic Bézier Simplices in Hiyoshi's Method

The following presents observations about the runtime complexity of  $C^2$  natural neighbor interpolation, which is based on Hiyoshi's extension of Farin's Bézier simplex construction as introduced in Section 3.2.7.

During experiments with the  $C^2$  interpolant  $f^{\text{HIY}2}$  over scattered data we noticed an explosion in run-time for partially structured input like the crater lake data set shown in Figure 6.11(a). Close examination unveiled that in very inhomogeneous data sets, the number of natural neighbors can grow arbitrarily large. We now take a closer look at the combinatorial impact of the number of neighbors. Consider again the construction of  $f^{\text{HIY}2}$  for a point  $\mathbf{x}_0$  with neighbors  $N(\mathbf{x}_0) = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ . After computation of the local coordinates  $\boldsymbol{\lambda}^{\text{HIY}} \in \mathbb{R}^m$ , they are used to evaluate the quintic  $m$ -variate Bézier simplex  $b^5(\boldsymbol{\lambda}^{\text{HIY}})$  with control points  $c_j$  as given in (3.11).

From the formula for the number of multi-indices of fixed norm in (2.2), we get the number of control points in degree five Bézier simplices as

$$\frac{(m+4)!}{(m-1)!5!} \in \mathcal{O}(m^5),$$

which is a significant impediment, especially when considering the computational cost associated with the construction of each control point. Counting the individual operations required to compute the directional derivatives in  $n$ -dimensional data points we find the following table.

|  | #lookup    | #+             | #*        |
|--|------------|----------------|-----------|
| $z_i$  | 1          |                |           |
| $z_{ij} = \nabla_i^T(\mathbf{x}_j - \mathbf{x}_i)$                                     | $3n$       | $2n$           | $n$       |
| $z_{ijk} = (\mathbf{x}_j - \mathbf{x}_i)^T \mathcal{H}_i(\mathbf{x}_k - \mathbf{x}_i)$ | $n^2 + 3n$ | $n^2 + 2n - 1$ | $n^2 + n$ |

We count the occurrences of these in the definition of control points  $c_j$ :

|             | # $z_i$ | # $z_{ij}$ | # $z_{ijk}$ |
|-------------|---------|------------|-------------|
| $c_i$       | 1       |            |             |
| $c_{ij}$    | 1       | 1          |             |
| $c_{ijk}$   | 1       | 2          | 1           |
| $c_{ijjl}$  | 2       | 4          | 4           |
| $c_{ijkl}$  | 4       | 12         | 12          |
| $c_{ijklm}$ | 5       | 20         | 30          |

This number of operations that leads to a large constant factor in the computation complexity. The most numerous control point type is  $c_{ijk}$ . If lookup, addition and multiplication are equally expensive, the cost  $C$  of evaluating the interpolant at a single position with  $m$  neighboring data sites is

$$C > 48 * m^5 \text{operations},$$

resulting in more than 153.6 million operations for 20 neighbors.



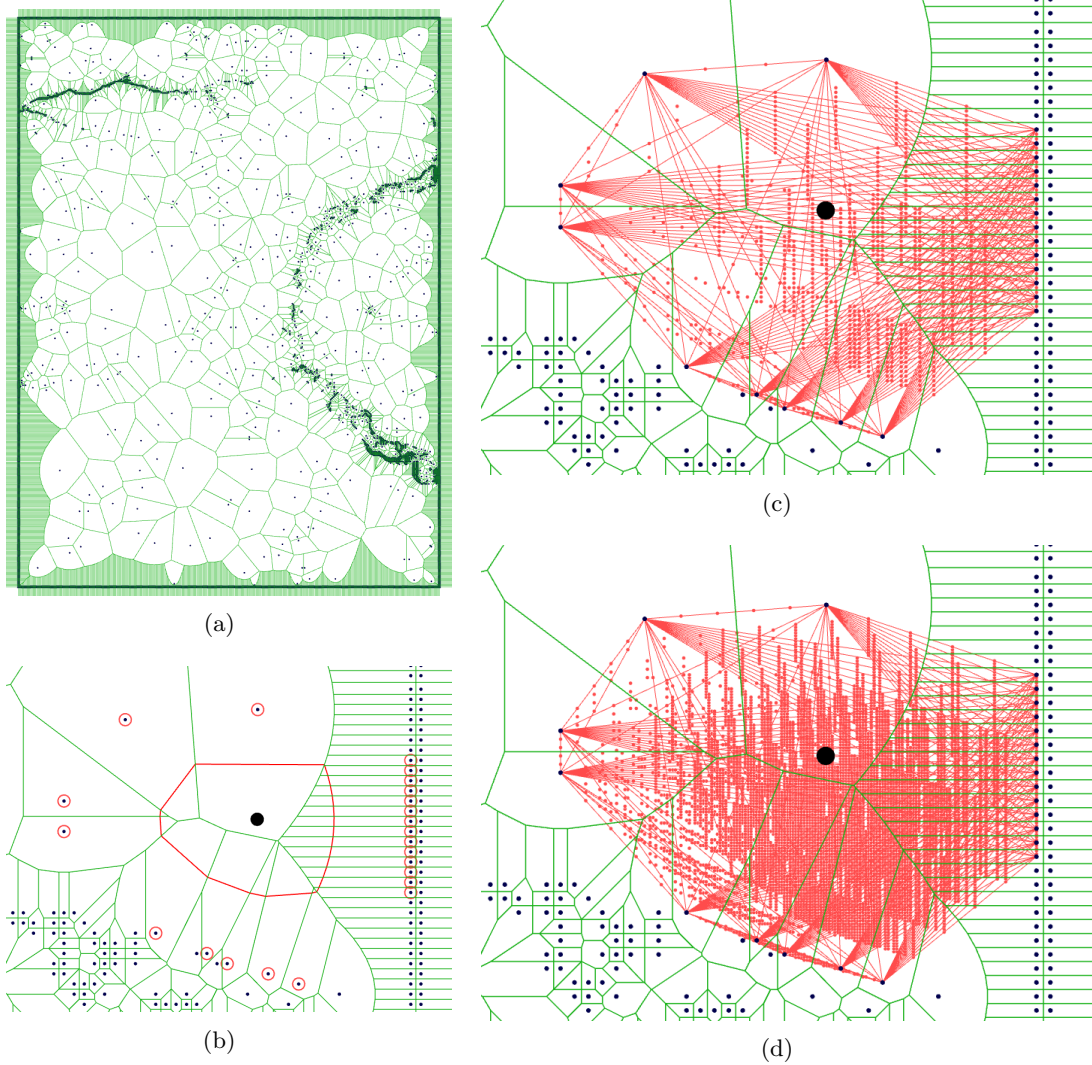


Figure 6.11: (a) Top view of the “Crater Lake” data set, the black dots representing the data sites, and green lines show the Voronoi diagram. Regular sub-structures are clearly visible. (b) Closeup of a virtual Voronoi tile for the fat dot, with its natural neighbors encircled. (c) Projected Bézier simplex  $b^3$  with its control points. (d) Projected Bézier simplex  $b^5$  with its control points.

If the interpolant is to be evaluated over a regular grid, as it is done for visualization, an obvious idea for improvement is to precompute the coefficients and store them for every set of natural neighbors. While possible in general, the computational complexity of a space-efficient indexing scheme for arbitrary neighborhoods renders a preprocessing approach infeasible.

## 6.6 Conclusion

Two fundamental problems in local derivative generation and estimation are the choice of neighborhood and the choice of a weighing function that smoothly decouples the neighborhood from the whole data set.

We have solved the first problem for the ad-hoc derivative generation based on fitting the Taylor approximation of a function. By choosing a  $d$ -times iterated natural neighborhood to fit Taylor terms up to order  $d$ , the fitting becomes an over-constrained least squares problem and therefore numerically stable.

Then, we presented a method that solves both problems by iteratively fitting derivatives up to order  $d$  to a the natural neighborhood at which derivatives are present up to order  $d - 1$ . This is can be done iteratively, starting from values at the data sites. The neighborhood structure, together with the matrices used in the fitting of higher order derivatives, guarantees the derivative fit to be an over-constrained least squares problems. However, this method is not capable of polynomial reproduction.

The methods have been applied to a synthetic data set, for which it was shown that the choice of neighborhood plays an important role in derivative generation, and a larger neighborhood generally provides better results. In the experimental verification, the first method stood up to the conjecture that the choice of iterated natural neighborhoods provides a solid basis for least squares fitting of higher order Taylor terms. In the numerical assessment, it was experimentally verified that both values and gradients converge to a smooth reference data set if Farin's  $C^1$  or Hiyoshi's  $C^2$  interpolant is applied to derivative data generated by any of the presented methods.

Finally, we discovered a severe complexity issue in Hiyoshi's  $C^2$ -smooth natural neighbor scattered data interpolation scheme. This limiting result applies to all methods that eventually employ high-dimensional, high-order Bézier simplices.

# 7 Discrete Harmonic Functions from Local Coordinates

In this chapter we present observations about the computation of discrete harmonic functions based on local coordinates in triangulations of arbitrary dimension, but restrict our presentation to the planar case. This work has been published in [BHFU07].

First we compare different local coordinate schemes by computing discrete harmonic functions as the solution to the discrete Laplace equation subject to boundary conditions sampled from a known harmonic function, and assess the resulting numerical approximation quality.

We then present our main observation: for discretizations given by continuously deforming point sets, natural neighbor coordinates are the only ones yielding a continuous dependency between the deformation and the harmonic function.

This chapter is based on terms and concepts introduced in Section 2.1 about linear algebra, in Section 2.4 about graphs, in Section 2.3.4 about the Delaunay triangulation, and on barycentric and natural neighbor coordinates introduced in Sections 2.1.2 and 3.2.5.

## 7.1 Background

Each triangulation of a point cloud is also a discretization of the continuous domain given by the convex hull. To compute an approximate solution to the Laplace equation – a discrete harmonic function – the Laplace operator is commonly approximated by a weighted sum of differences in neighborhoods of a triangulation. Given a concrete point cloud, two choices determine the nature of the approximation: the triangulation and the weights.

A cautious note: although we approximate the Laplace operator, these approximations are no valid Laplacian approximations in general, since most of them do not possess a meaningful norm. In the computation of discrete harmonic functions, however, this norm is irrelevant.

The remainder of this chapter investigates several choices for weights in the special setting where the triangulation is a Delaunay triangulation. Before we give the definition of discrete harmonic functions and some well-known results, we discuss some related work on the importance of harmonic functions.

### 7.1.1 Related Work

Harmonic functions are defined as solutions of the most basic second order partial differential equation (PDE), the *Laplace equation*. The associated scientific field of potential theory, concerned with solutions to Laplace's equation, has applications in electromagnetism, astronomy, fluid dynamics, and thermodynamics, where harmonic functions are used to describe the behavior of electric, gravitational, fluid, and heat flow potentials. Any computational assessment of such problems requires discrete approximations, usually resulting in finite element approximations that compute discrete harmonic functions.

In the seemingly disconnected field of *graph theory*, concerned with discrete structures made from vertices and edges [Die05, Bol98, GR01], an equivalent notion of Laplacian and discrete harmonic function exists. For example, in electrical engineering, the equilibria in the flow of currents in electrical networks correspond to discrete harmonic functions on the graph representation of the electrical network. Here, the problem is discrete by nature and has no spatial interpretation. An extensive investigation of the relation between the continuous and the discrete Laplacian operator, and as part of it, of continuous and discrete harmonic functions, can be found in Fleischer's Ph.D. thesis [Fle07]. Further recent results on the discretization of the Laplacian has been part of Hirani's Ph.D. thesis in [Hir03].

In robotics, discrete harmonic functions are used to compute potential fields between obstacles in path planning, as Connolly and Grupen surveyed in [CG92].

The extension of the Laplace operator to manifolds is called the Laplace-Beltrami operator, which incorporates the local parameterization of the manifold under concrete embeddings. Approximations of the Laplace-Beltrami operator have been used extensively for mesh processing in computer graphics and geometric modeling, which is surveyed in Sorkine [Sor06]. Interestingly, the proper choice of edge weights in the linear approximation of the Laplace-Beltrami operator has been of central interest in Laplacian based mesh processing over a long time. Wardetzky et al. [WMKG07] showed that there is no approximation of the Laplace operator on an arbitrary, piecewise linear manifold that satisfies all the properties the mesh processing community has been looking for. However, Fisher et al. discovered in [FSBS06] the connection between the Delaunay triangulation in the Euclidean plane and an equivalent structure, the intrinsic Delaunay triangulation on a piecewise linear manifold, which allows for a meaningful and adequate definition of the Laplace operator. Some fundamental results on the nature of discretization of the Laplace-Beltrami operator can be found in the work of Polthier in [Pol05] and Bobenko in [BS07].

Examples for applications of the Laplacian in mesh processing are the work on signal mesh processing by Taubin [Tau95], mesh optimization by Sorkine et al. [NISA06], or mesh editing by Alexa [Ale06], to name a few. We specifically point out the work of Dong et al. in [DKG05], where discrete harmonic functions were used for quadrilateral remeshing of piecewise linear surfaces.

In mechanical engineering, the Laplace discretization plays a crucial role for many fundamental problems. Recent advances in mesh-free methods have, among others, built upon natural neighbor interpolation, like Sukumar and Moran in [SM99]. In this context, it is

interesting to investigate the connection between Laplacian approximations derived from meshes and those based on natural neighbor coordinates.

### 7.1.2 Harmonic Functions and Their Discretization

A *harmonic function* on a domain  $\Omega$  is the solution to the *Laplace equation*

$$\Delta f|_{\Omega} = 0,$$

where  $\Delta = \nabla^2$  is the *Laplace operator* and  $\Delta f(\mathbf{x})$  is called *Laplacian of  $f$  at  $\mathbf{x}$* .

The remainder of this chapter will focus on planar domains  $\Omega \subset \mathbb{R}^2$ , mainly because this facilitates the display of the function graphs as height-field triangulations, and makes available a variety of planar polygonal barycentric coordinates to compare to. This is no limitation since the considered concepts are independent of the dimension of the underlying space. For a function  $(u, v) \in \Omega \mapsto f(u, v) \in \mathbb{R}$ , the Laplacian  $\Delta f$  has the explicit representation

$$\Delta f = \nabla^2 f = \frac{\partial^2}{\partial u^2} f + \frac{\partial^2}{\partial v^2} f.$$

For illustrative purposes, consider the following interpretation of the relation between the discrete and continuous Laplacian based on divided differences, similar to Example 2.5.11 in [Fle07]. In the univariate case with discrete function values at an equally spaced set of ordinates,  $f_i = f(u_0 + h \cdot i)$ ,  $i \in \mathbb{Z}$  the second derivative is approximated by the second divided difference

$$\frac{\partial^2}{\partial u^2} f_i \approx \Delta f_i = \frac{f_{i-1} - 2f_i + f_{i+1}}{h^2}.$$

In the gridded, two-dimensional case with  $f_{i,j} = f(u_0 + h \cdot i, v_0 + h \cdot j)$ , the Laplacian is approximated by

$$\Delta f_{i,j} = \frac{f_{i-1,j} + f_{i+1,j} - 4f_{i,j} + f_{i,j-1} + f_{i,j+1}}{h^2}.$$

The terms above contain the function value and its four neighbors in each canonical grid direction,  $N_{i,j} = \{(i-1, j), (i+1, j), (i, j-1), (i, j+1)\}$ , and allow us to write

$$\Delta f_{i,j} = \frac{1}{h^2} \sum_{(k,l) \in N_{i,j}} (f_{k,l} - f_{i,j}).$$

This is an intuitive explanation for the linear Laplacian approximation in general grids  $f_i = f(\mathbf{x}_i)$ ,  $\mathbf{x}_i \in X \subset \mathbb{R}^2$  that is given as

$$\Delta f_i = \sum_{j \in N_i} \lambda_{ij} (f_j - f_i),$$

where  $N_i$  are the neighbors of  $i$  in a not further specified grid, and  $\lambda_{ij}$  are weights that generalize the discretization step-size to non-uniform distances to neighboring vertices.

Next we introduce some basic results and definitions as found in [GR01].

**Definition 7.1 (Discrete Laplacian)** Let  $G = (V, E, \Lambda)$  be a weighted, connected, undirected graph with positive weights  $0 \leq \lambda_{ij} \in \Lambda$ , and  $f : V \rightarrow \mathbb{R}$  a function on  $V$  with  $f_i := f(v_i)$ . Then,

$$\Delta f_i = \sum_{e_{ij} \in E} \lambda_{ij} (f_j - f_i)$$

is called the graph Laplacian of  $f$  on  $G$  at vertex  $v_i$ .

A definition of discrete harmonic functions can now be given in correspondence to the continuous setting.

**Definition 7.2 (Discrete Harmonic Function)** Let  $G = (V, E, \Lambda)$  be a weighted, connected, undirected graph with positive edge weights, and  $f : V \rightarrow \mathbb{R}$  a function on  $V$  with  $f_i := f(v_i)$ . Let

$$H = \{v_i \in V : \Delta f_i = 0\}, \quad \text{and} \quad P = V \setminus H.$$

The function  $f$  is said to be a discrete harmonic function with poles  $P$ . Furthermore,  $f$  is said to be harmonic at  $v \in H$ .

Several results are known for this abstract definition, the one important for the remainder of this chapter is given below.

**Theorem 7.3** Let  $G = (V, E, \Lambda)$  a weighted, undirected graph with  $\lambda_{ij} \geq 0$ . For every set  $P = V \setminus H$ , and every function  $f|_P : P \rightarrow \mathbb{R}$ , there is a unique function  $f|_H : H \rightarrow \mathbb{R}$  such that  $f|_V : V \rightarrow \mathbb{R}$  is a discrete harmonic function with poles  $P$  on  $G$ .

The proof for this theorem is based on the positive definiteness arising from the positive edge weights and can be found in [Lov00].

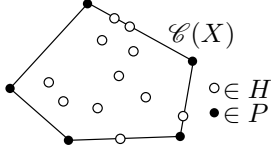
## 7.2 Discrete Harmonic Functions from Local Coordinates

Assuming an arbitrary triangulation, there are several polygonal barycentric coordinate schemes that provide edge weights which reflect the spatial setting in the sense that they represent a Laplacian approximation for which the coordinate functions of the vertices are the solution to the Laplace equation – which matches the corresponding property in the continuous case.

If the triangulation is a Delaunay triangulation, the weights can as well be computed using natural neighbor coordinates.

### 7.2.1 Discrete Harmonic Functions over Triangulations

Now we transfer the general definition of discrete harmonic functions onto the spatial setting in which  $V \equiv \mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^2$ , and present the steps involved in the computation of the solution for the discrete Laplace equation. With the later application


 Figure 7.1: Illustration of  $H$  and  $P$  in a point set with respect to  $\mathcal{C}(\mathbf{X})$ .

of natural neighbor coordinates in mind, we restrict our considerations to the Delaunay triangulation of  $\mathbf{X}$ , whose edges  $E_{\text{DEL}}$  become edges in the corresponding planar straight line graph

$$G := (\mathbf{X}, E_{\text{DEL}}).$$

The later application of generalized barycentric coordinates motivates the association of  $H$  with interior points, i.e., those contained within the convex hull of their neighbors, and  $P$  with boundary points,

$$H := \{\mathbf{x} \in \mathbf{X} : \mathbf{x} \in \mathcal{C}(N(\mathbf{x}))\}, \quad P := \mathbf{X} \setminus H. \quad (7.1)$$

Note that there can be points *on* the convex hull that are not in  $P$  as indicated in Figure 7.1.

For now assume that there are some weights  $\lambda_{ij} \geq 0$  assigned to every directed edge in  $G$ , and  $\lambda_{i1} + \dots + \lambda_{in} = 1$ . Recall that continuous harmonic functions on a domain  $\Omega$  are the unique solution to the Laplace equation subject to Dirichlet boundary conditions  $b : \partial\Omega \rightarrow \mathbb{R}$ ,

$$\Delta f = 0, \quad \text{subject to } f|_{\partial\Omega} = b.$$

Analogously,  $f|_H$  is the unique discrete harmonic extension of  $f|_P$  to  $f$  on  $G$ , i.e.,  $f|_H$  is the solution of the Laplace equation with boundary conditions  $f|_P$ ,

$$\begin{aligned} \Delta f_i &= 0, & \text{for } \mathbf{x}_i \in H \\ f_i &= b(\mathbf{x}_i), & \text{for } \mathbf{x}_i \in P \end{aligned}$$

We may without loss of generality assume  $\mathbf{X}$  enumerated such that

$$f_X := \underbrace{(f_1, \dots, f_m)}_{=: f_H}, \underbrace{(f_{m+1}, \dots, f_n)}_{=: f_P} \in \mathbb{R}^n, \quad \mathbf{x}_1, \dots, \mathbf{x}_m \in H, \quad \mathbf{x}_{m+1}, \dots, \mathbf{x}_n \in P,$$

represents the column vector of discrete function values, and is partitioned into the unknown harmonic values  $f_H$  and the boundary values  $f_P$ . The Laplace operator acting on the whole function can now be written as

$$\Delta f_H = f_H - S f_X = [(\mathbf{I} - S^H) S^P] \begin{bmatrix} f_H \\ f_P \end{bmatrix} = (\mathbf{I} - S^H) f_H + S^P f_P,$$

where  $S \in \mathbb{R}^{m \times n}$  with  $s_{ij} = \lambda_{ij}$  if  $\mathbf{x}_i \in H$  and  $\mathbf{x}_j \in \mathbf{X}$ , and  $s_{ij} = 0$  otherwise, and  $S = [S^H S^P]$  is the partition into  $S^H \in \mathbb{R}^{m \times m}$  and  $S^P \in \mathbb{R}^{m \times (n-m)}$ . Setting  $\Delta f_H = \mathbf{0}$  and solving for  $f_H$  yields

$$f_H = \underbrace{(\mathbf{I} - S^H)^{-1} S^P}_{=: M} f_P.$$

Obviously,  $M$  is only defined if  $(\mathbf{I} - S^H)$  is invertible, which can be shown as follows.

**Corollary 7.4** *The matrix  $M$  is regular.*

**Proof 7.5** *Because  $0 \leq \lambda_{ij}$  and  $\sum_j \lambda_{ij} = 1$ , row sums in  $S^H$  are equal to one if  $\mathbf{x}_i \in H$  and between zero and one if  $\mathbf{x}_i \in P$ . Therefore,  $\mathbf{I} - S^H$  is weakly diagonally dominant. Because the non-zero structure of  $S^H$  is equivalent to the graph connection matrix of the interior vertices of the triangulation, which is connected by definition, this matrix is also irreducible. With a result on the regularity of irreducible, weakly diagonally dominant matrices from [Var62], the claim follows.*

□

**Remark 7.6** *In general, a Laplacian discretization does not satisfy our assumption  $\sum_j \lambda_{ij} = 1$ . However, the unit matrix in the thread above can be replaced by the diagonal matrix  $D$  with entries  $d_{ii} = \sum_j \lambda_{ij}$ , which does not invalidate the arguments.*

**Remark 7.7** *Obviously, the most expensive part in the computation of a discrete harmonic function is the solution of a sparse linear system. Although the nature of the matrix  $M$  calls for iterative solvers which take advantage of fast matrix-vector multiplications that are possible in case of sparse matrices, we found that direct solvers on sparse matrices like SuperLU [Li05] or TAUCS [TCR03] often perform better, and in general at least as good as the iterative approaches.*

## 7.2.2 Laplacian Discretizations Based on Local Coordinates

We have fixed the triangulation to be the Delaunay triangulation, and showed how to compute the solution to the Laplace equation subject to Dirichlet boundary conditions under the assumption that  $0 \leq \lambda_{ij}$ . We now turn our attention to several choices of  $\lambda_{ij}$ .

To motivate our choice of generalized barycentric coordinates as approximations for the discrete Laplace operator, we consider the Laplacian of the coordinate functions,  $c^{[k]}(\mathbf{x}) := x_k$ . Because they are linear in  $\mathbf{x}$ , the Laplace equation  $\Delta c^{[k]} = 0$  holds. In the discrete case, this property corresponds for  $\mathbf{x}_i \in H$  to

$$\begin{aligned} 0 &= \Delta c^{[k]}(\mathbf{x}_i) \\ &= \sum_{j \in N_i} \lambda_{ij} (c^{[k]}(\mathbf{x}_j) - c^{[k]}(\mathbf{x}_i)) \quad k = 1, \dots, n \\ \iff 0 &= \sum_{j \in N_i} \lambda_{ij} (\mathbf{x}_j - \mathbf{x}_i), \end{aligned}$$

which is the definition of homogeneous barycentric coordinates as given in Section 2.1.2, justifying our choice of convex, generalized barycentric coordinates in the following.

In a planar triangulation, the neighborhood  $N_i$  of an inner vertex always comprises a star-shaped polygon with  $\mathbf{x}_i$  contained in its kernel, see Section 2.3.1. We therefore consider



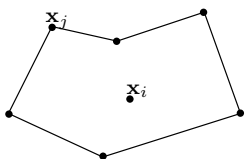
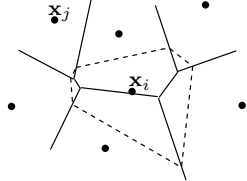
|   |                        |  |                        |
|---|------------------------|--|------------------------|
|  |                        |  |                        |
| cotangent coordinates   | $\lambda^{\text{COT}}$ | Laplace coordinates  | $\lambda^{\text{LAP}}$ |
| Wachspress coordinates  | $\lambda^{\text{WAC}}$ | Sibson coordinates   | $\lambda^{\text{SIB}}$ |
| mean value coordinates  | $\lambda^{\text{MVC}}$ | Hiyoshi coordinates  | $\lambda^{\text{HIY}}$ |

Table 7.1: Coordinates used in the assessment.

the polygonal generalized barycentric coordinate methods listed in the left column of Table 7.1 that were introduced in Section 3.1.5. Inside a single, convex polygon, all the above local coordinates are convex, and rational functions of the position of the polygon vertices. For  $\lambda^{\text{COT}}$  and  $\lambda^{\text{MVC}}$ , this stays true inside the kernel of non-convex polygons as well.

In addition to polygonal barycentric coordinates, the deliberate choice of the Delaunay triangulation makes natural neighbor coordinates from Section 3.2.5 applicable, listed in the right column of Table 7.1. In contrast to polygonal barycentric coordinates, natural neighbor coordinates are defined independent from any *explicit* adjacency on  $\mathbf{X}$ , based on the *implicit* adjacency defined by the Voronoi diagram. The duality between Delaunay triangulation and Voronoi diagram mentioned in Corollary 2.3 guarantees that the natural neighbors of any point  $\mathbf{x}_i \in \mathbf{X}$  are contained in the one-ring neighborhood  $N(\mathbf{x}_i)$  of  $\mathbf{x}_i$  in the Delaunay triangulation. Furthermore, as a consequence of Definition 3.1, those points in  $N(\mathbf{x}_i)$  that are not natural neighbors will have a zero coordinate associated.

Note also that the delicate definition of interior and boundary points in (7.1) honors the fact that natural neighbor coordinates can be extended to the boundary of the convex hull by a limit argument, where they reduce to the piecewise linear interpolant.

### 7.2.3 Experimental Comparison

Next we discuss experimental results on the approximation quality of discrete harmonic functions based on one of the Laplacian discretizations presented in the previous section.

As basis for comparison we consider the discrete approximation of

$$g(x, y) = \frac{1}{3}x^3 - xy^2$$

over the domain  $\Omega = [0.3, 1.3] \times [0.1, 1.1]$ , whose graph is shown in Figure 7.2(a).

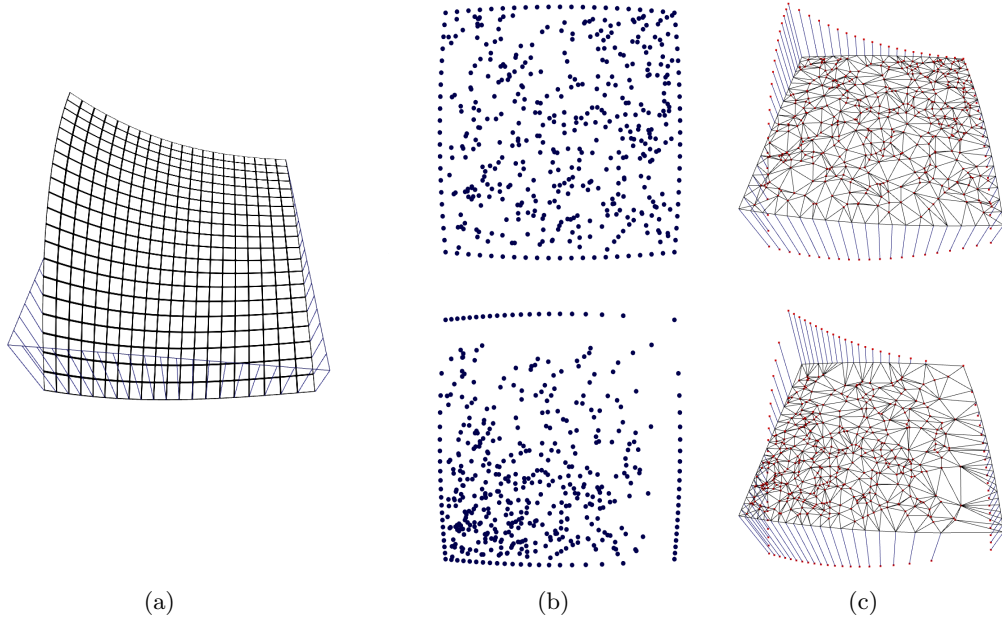


Figure 7.2: (a) Graph of the function  $g(x, y) = \frac{1}{3}x^3 - xy^2$ . (b) Point distribution. (c) Delaunay triangulation and values at the boundary. Homogeneous distribution shown in upper row, inhomogeneous distribution in lower row.

In the following we denote the approximate solutions to  $g$  by  $\tilde{f}^\bullet$ , where  $\bullet = \text{COT}, \text{WAC}, \text{MVC}, \text{LAP}, \text{SIB}$  or  $\text{HIY}$  according to the barycentric coordinates upon which the Laplacian approximation is based. To illustrate the effect of a Laplacian discretization that does not arise from barycentric coordinates but still tries to capture the spatial relation between the points, our comparison includes inverse distance coefficients,

$$\hat{\lambda}_{ij}^{\text{INV}} = \|\mathbf{x}_i - \mathbf{x}_j\|^{-1}.$$

We choose two distributions of  $|\mathbf{X}| = 480$  points for the discretization, one homogeneously distributed shown in Figure 7.2(b) top row, one with a denser distribution in the lower left shown in Figure 7.2(b) bottom row. This distinction helps examine how the approximation quality is influenced by the homogeneity of the sample distribution.

The Dirichlet boundary conditions are sampled from  $g$  at the points on the boundary, where the slight curving visible in Figure 7.2(b,c) is intentional to stabilize the classification of boundary vertices of the triangulation.

Table 7.2 shows the analysis of the per-point errors and indicates that Laplace- resp. cotangent coordinates yield the lowest error, while the others are worse by orders of magnitude. Furthermore, it seems that in the inhomogeneous setting, this discrepancy is less prominent for natural neighbor coordinates.

Figure 7.3 shows the visual comparison of  $\tilde{f}^{\text{COT}}$  and  $\tilde{f}^{\text{INV}}$ . The results obtained using any of the local coordinates in the Laplacian discretization are visually indistinguishable from the exact solution sampled from  $g$  (Figure 7.3(a) and (c)). As Figure 7.3(b) and (d)

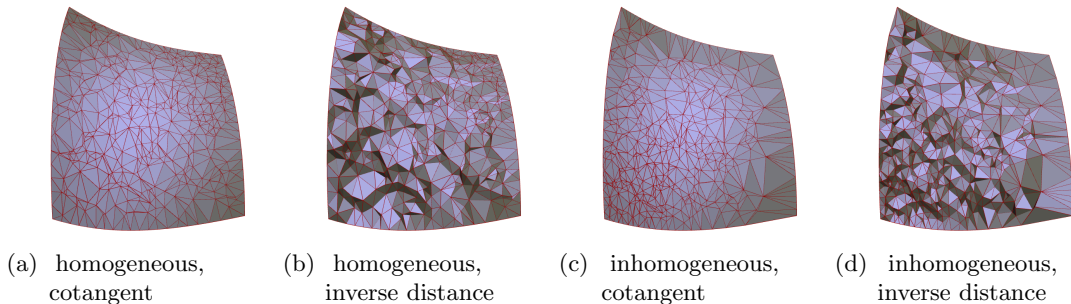


Figure 7.3: Delaunay triangulation of  $\mathbf{X}$  in 2.5D, the  $z$ -axis representing the values of  $f_i$  for the solution of the Laplace equation based on (a),(c) cotangent coordinates (visually indistinguishable from the other barycentric coordinate methods), and (b),(d) based on inverse distance weights. The triangles are flat shaded to enhance the noisy, bad result rendered by the application of inverse distance weights.

| Coordinates used for approximation |                          | Homogeneous point distribution |         | Inhomogeneous point distribution |         |
|------------------------------------|--------------------------|--------------------------------|---------|----------------------------------|---------|
|                                    |                          | RMS                            | MAX     | RMS                              | MAX     |
| Laplace coordinates                | $\tilde{f}^{\text{LAP}}$ | 7.08e-8                        | 1.08e-3 | 3.22e-7                          | 3.88e-3 |
| Sibson coordinates                 | $\tilde{f}^{\text{SIB}}$ | 2.53e-6                        | 3.65e-3 | 1.92e-6                          | 4.20e-3 |
| Hiyoshi coordinates                | $\tilde{f}^{\text{HIY}}$ | 5.78e-6                        | 5.44e-3 | 3.57e-6                          | 4.66e-3 |
| Cotangent coordinates              | $\tilde{f}^{\text{COT}}$ | 7.08e-8                        | 1.08e-3 | 3.22e-7                          | 3.88e-3 |
| Mean value coordinates             | $\tilde{f}^{\text{MVC}}$ | 5.60e-6                        | 4.85e-3 | 7.37e-6                          | 7.04e-3 |
| Wachspress coordinates             | $\tilde{f}^{\text{WAC}}$ | 2.53e-5                        | 1.10e-2 | 2.09e-5                          | 1.18e-2 |
| Inverse distance weights           | $\tilde{f}^{\text{INV}}$ | 5.43e-4                        | 9.32e-2 | 8.76e-4                          | 8.09e-2 |

Table 7.2: Comparison of root mean square (RMS) error and maximum absolute deviation (MAX) for the per-point error  $|\tilde{f}(\mathbf{x}_i) - g(\mathbf{x}_i)|$ .

show, there are considerable distortions for Laplacian approximations under which the coordinate functions are not harmonic, such as for  $\lambda^{\text{INV}}$ .

## 7.2.4 Dynamic Aspects

We are now interested in how discrete harmonic functions based on the different Laplacian discretizations behave if the coordinates of the positions of the point set vary continuously. The motivation for this is the difference between a tessellation of a point set and its Voronoi diagram, as illustrated in Figure 7.5. Figure 7.6(a) shows the Delaunay triangulation of  $\mathbf{X}$  for a simple setting with  $|H| = 4$ ,  $|P| = 10$ , which we deliberately choose to contain three sets of co-circular points. We assume the origin in the center of the data set and shear the points in the direction indicated by the arrows, thus causing the ambiguous diagonals to flip between two possible positions as indicated by the dashed lines. Note that such flips occur in any triangulation if the deformation of the point set is big enough. Because the alternating boundary values shown in Figure 7.6(d) are horizontally symmetric, the discrete harmonic function should be symmetric as well.



Figure 7.4: Swirling deformation applied to the point cloud.

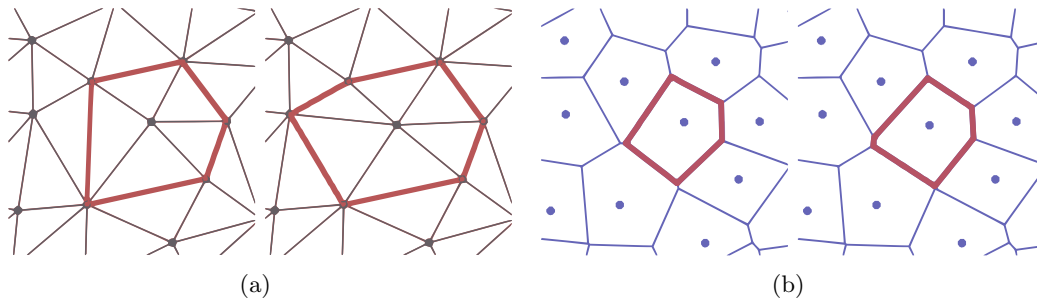


Figure 7.5: (a) Delaunay triangulation. (b) Voronoi diagram. In the same set of points, the middle point moves a differential amount between left and right setting. Note how the significant topological changes in (a) are not present in (b).

Figure 7.6 shows the results of computing the discrete harmonic function on the point set under a shearing deformation, just before and after the diagonal flip. The results for  $\tilde{f}^{\text{COT}}$ ,  $\tilde{f}^{\text{LAP}}$ ,  $\tilde{f}^{\text{SIB}}$ , and  $\tilde{f}^{\text{HIY}}$  are visually indistinguishable, which is why we only show one representative picture in Figure 7.6(c). First, notice how both  $\tilde{f}^{\text{WAC}}$  in Figure 7.6(d) and  $\tilde{f}^{\text{MVC}}$  in Figure 7.6(e) result in an asymmetric harmonic function in spite of the symmetry in the point set and the values. As long as the triangulation stays the same, the harmonic function for all approaches continuously follows the deformation. As a result of the diagonal flip, however, the polygonal coordinates show a discontinuous change in the harmonic function, which is not the case for natural neighbor coordinates.

### 7.3 Conclusion

We have compared different discretizations of the Laplacian, applied to the computation of discrete harmonic functions from prescribed boundary values. To this end, we used the Delaunay triangulation to provide the connectivity, and generalized barycentric coordinates in the one-ring neighborhood to provide the edge weights. The generalized barycentric coordinates can be separated into polygonal barycentric coordinates and natural neighbor coordinates.

First, we assessed the approximation quality by comparing values sampled from a known harmonic function to those computed using either of the considered Laplacian approximations from boundary values that were sampled from the known function. Although

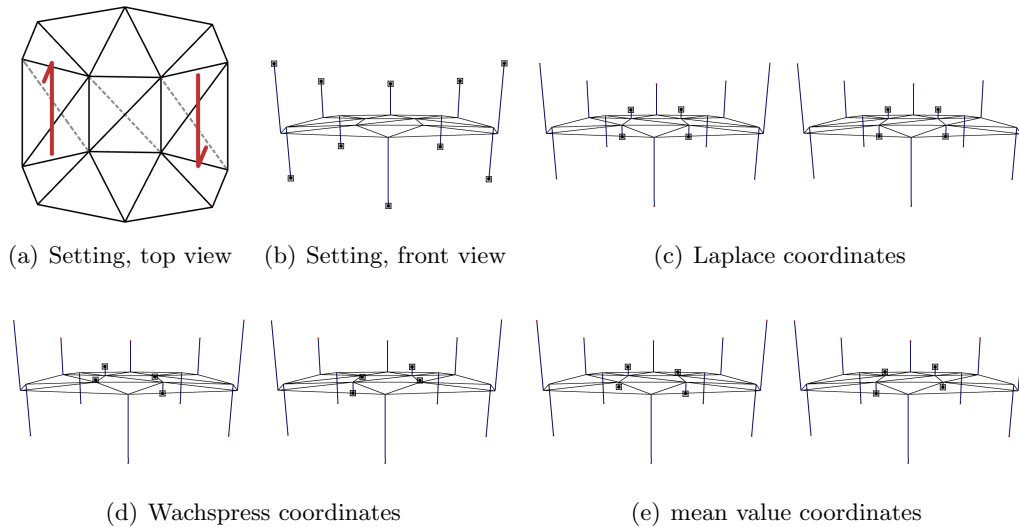


Figure 7.6: (a) Shear around the center of the data set is indicated by the arrows, the dashed lines depicting the two possible diagonal Delaunay triangulations. (b) Boundary condition values are shown as thick dots, the interior four vertices are to be computed. (c)-(e) The thick dots indicate the computed discrete harmonic function values at the vertices of the triangulation. The left and right pictures show the values for the two different diagonal directions. Note the jumps in (d), (e) which does not occur in (c).

all approaches based on generalized barycentric coordinates provide acceptable results, the lowest approximation error by several orders of magnitude resulted from approximations based on Laplace/cotangent coordinates, which are identical in case of Delaunay triangulations.

Second, we used a point set with fix boundary values but continuously changing positions to analyze the effect of triangulation changes on the discrete harmonic function. We have shown that Laplacian approximations based on polygons yield discontinuous changes in the harmonic function under modifications in the triangulation.

Because cotangent coordinates, when computed in the Delaunay triangulation, are identical to Laplace coordinates, they technically belong to the class of natural neighbor coordinates. The natural neighbor coordinates  $\lambda^{\text{LAP}}$ ,  $\lambda^{\text{SIB}}$ , and  $\lambda^{\text{HIY}}$  always admit harmonic functions that continuously change with the positions of the point set, making them especially suited for application on variable point sets. Based on the nature of the problem setting, we conjecture that the Laplacian approximation inherits the  $C^1$  continuity of Sibson coordinates and the  $C^2$  continuity of Hiyoshi coordinates. It would be interesting to investigate possible applications in the context of mechanical engineering, where mesh-free methods heavily rely on the absence of any explicit mesh structure.

We are confident that within an appropriate framework these results also apply to the manifold case. One promising method is that of [Flö03b] which provides a setting for manifolds that is similar to the one we used here. Another option lies in the generalization of natural neighbor coordinates based on the intrinsic Delaunay triangulation.



# 8 Natural Neighbor Extrapolation

Among locally supported scattered data schemes, natural neighbor interpolation has some unique features that makes it interesting for a range of applications. However, its restriction to the convex hull of the data sites is a limitation that has not yet been satisfyingly overcome. Evaluating an interpolant outside the convex hull is called extrapolation and usually amounts to educated guessing. While global scattered data interpolation and approximation such as radial basis functions or tensor product spline fitting naturally allow extrapolation to some extent, local methods do not. In this setting, we discuss some aspects of scattered data extrapolation in general, and compare existing methods.

The key contribution of this chapter is a framework for smooth local extrapolation of data defined over point sets that seamlessly blends with classical natural neighbor interpolation schemes. Each application dictates its own notion of acceptable behavior outside the convex hull, which might be constant, or asymptotic towards a polynomial or exponential falloff. Our framework offers such control over the extrapolant away from the convex hull. Parts of this chapter have been published in [BFHU08].

## 8.1 Introduction

It is often implicitly assumed that interpolation is restricted to the “intuitive interior” of the data sites, which is usually their convex hull. Extrapolation in this context means to find a function that also extends to the “intuitive exterior” of the data sites, i.e., to interpolate over a domain that extends past the convex hull. Extrapolation can be achieved in multiple ways. An interpolant might be globally defined, like radial basis function (RBF) or inverse distance weighted (IDW) interpolants, in which case its evaluation outside the convex hull amounts to extrapolation. Interpolation schemes with limited domain can be extended by constructing a function that smoothly joins the interpolant at the boundary of its domain.

While extrapolation in itself is an ill-posed problem, two different objectives can be identified when dealing with it: The first is to construct pleasant-looking functions (surfaces) that leave the transition between the intuitive interior and exterior unnoticed. Applications for this are the visualization of unstructured data over a rectangular domain, like weather forecast data or digital elevation models in geoinformation systems and architecture. The second objective is to construct a function that agrees with application specific expectations that encode knowledge about the application domain. This is typically given in the context of open boundary simulations, oil exploration [MS99, Zor04], or biosciences [MTS<sup>+</sup>04], where properties of the underlying physical model play a role in the interpolation method. Another possible application emerges in the modeling of binder surfaces

for stamping, which must be extended beyond the surface to be stamped. To facilitate the classification of extrapolation methods, we propose to adapt a set of criteria that has previously been introduced by Alfeld in a technical report on triangular extrapolation methods [Alf84].

We investigated several new ways to extend natural neighbor interpolation past the convex hull of the data sites, which we present here. The final result is a framework that alleviates some of the issues present in current extrapolation methods. Based on dynamically inserted new data sites, the ghost points, we are able to extend any natural neighbor interpolant over an arbitrary data set to all of space, while preserving desirable properties like smoothness or the continuous dependence on the coordinates of the data sites.

We start with a brief summary of relevant scattered data interpolation methods and presents previously proposed approaches for the extrapolation of local scattered data interpolants in Section 8.2. A taxonomy of scattered data extrapolation approaches is developed and applied to the presented methods in Section 8.3, including the methods developed later in this chapter. Following that, we will describe the different ideas we have pursued and comment on their extrapolation performance in Sections 8.4-8.6. We present the key contribution of this chapter, natural neighbor extrapolation with dynamic ghost points, in Section 8.7. A thorough visual comparison of some selected extrapolation methods in challenging situations is given in Section 8.8. This chapter is concluded in Section 8.9.

## 8.2 Related Work

Extrapolation is strongly linked to interpolation in that, either, interpolation methods with restricted definition domains are extended or, interpolation methods naturally cover a domain that is larger than the convex hull of the data sites. We therefore briefly summarize relevant scattered data interpolation methods in Section 8.2.1 and extrapolation methods in Section 8.2.2.

For the rest of this chapter, we drop explicit summation bounds for ease of notation and assume that indices address the feasible, finite range.

### 8.2.1 Scattered Data Interpolation

The vast field of scattered data interpolation has been around since the mid-seventies. Surveys on scattered data interpolation in general by Franke and Nielson can be found in [Fra79, Fra82, FN91], on tessellation-based interpolation by Zeilfelder and Seidel in [ZS02], and a textbook on RBF approaches by Wendland in [Wen04]. We consider the scattered data interpolation problem for a set of data sites  $\mathbf{X} = \{\mathbf{x}_i\}_i$ ,  $\mathbf{x}_i \in \mathbb{R}^n$ , and data  $Z = \{z_i\}_i$ , where in this context  $z_i(\mathbf{x})$  is a Taylor polynomial of given degree  $k \in \{0, 1, 2\}$  carrying the partial derivative information for data site  $\mathbf{x}_i$ , such that  $z_i(\mathbf{x} - \mathbf{x}_i)$  locally approximates an unknown function represented by the data. Given this input, an interpolation method  $I$  provides an  $n$ -variate, real-valued function  $f = I(\mathbf{X}, Z) \in C^d$ ,  $d \in \mathbb{N} \cup \{\infty\}$ , that satisfies



the interpolation property for derivatives up to order  $k$ ,

$$\left. \frac{\partial^{|\mathbf{j}|} f}{\partial \mathbf{x}^{\mathbf{j}}} \right|_{\mathbf{x}=\mathbf{x}_i} = \left. \frac{\partial^{|\mathbf{j}|} z_i}{\partial \mathbf{x}^{\mathbf{j}}} \right|_{\mathbf{x}=\mathbf{0}}, \quad |\mathbf{j}| \leq k,$$

where  $\mathbf{j} = (j_1, \dots, j_n) \in \mathbb{N}_0^n$  with  $|\mathbf{j}| = j_1 + \dots + j_n$ . If  $f$  is defined over the domain  $\Omega \subset \mathbb{R}^n$ , then we call  $D(I) := \Omega$  the *definition domain* of  $I$ . We denote the convex hull of  $\mathbf{X}$  by  $\mathcal{C}(\mathbf{X})$ , and the data sites on the boundary of  $\mathcal{C}(\mathbf{X})$  by  $\mathbf{X}^{\mathcal{C}}$ . In the sequel, we focus on bivariate interpolants.

Most scattered data interpolation schemes require some sort of preprocessing that is performed once before being able to evaluate the interpolant  $I$  at arbitrary positions in  $D(I)$ . The complexity of an interpolation scheme is consequently split into the complexity of preprocessing and the complexity of evaluating the interpolant.

A common preprocessing step is the generation of derivative information for  $k > 0$ , which is often not part of the input to scattered data interpolation. Although many schemes explicitly define their own estimation procedure, it is in general decoupled from the interpolation. Appropriate methods for derivative generation can be found in [Sib81, Nie83, Aki84, Ste84, Alf85], or in Chapter 6.

The essential step common to all interpolation methods is the computation of a value for a query position, which is characterized by the number of data sites involved in the computation. Global schemes are more expensive in that they process all input data in each evaluation, but generally allow for a higher smoothness, while local schemes have lower computational complexity at the expense of only a limited degree of continuity.

## Global Schemes

The most prominent global interpolation scheme is the radial basis function (RBF) approach, which produces smooth interpolants of high quality and has been discussed in Section 3.1.3. In our implementation, we applied to each basis function  $\varphi_i$  a scaling factor that is a multiple of the distance of  $\mathbf{x}_i$  to its furthest direct neighbor in the Voronoi diagram of  $\mathbf{X}$ . We will consider the RBF method with following basis functions:

- $I^{\text{RBF}^{\text{T}}}$ ,  $\phi(r) = r^2 \log r$  (thin plate splines), global,
- $I^{\text{RBF}^{\text{G}}}$ ,  $\phi(r) = \exp(-\alpha r^2)$ ,  $\alpha \in \mathbb{R}^+$  (Gaussian), quasi-local,
- $I^{\text{RBF}^{\text{W}}}$ ,  $\phi(r) = (1-r)_+^4 (4r-1)$ , where  $(x)_+^4$  is the truncated fourth degree monomial (Wendland's compact polynomial), local.

Another approach to global interpolation is given by inverse distance weighted (IDW) schemes, which we discussed in Section 3.1.1. We will consider Shepard's method modified in two ways. First, blending quadratic Taylor polynomials at the data sites with global support. Second, blending quadratic Taylor polynomials and using a modified weighting function to localize the support. However, by dropping the global support of the weight functions, the interpolant is no longer defined in all  $\mathbb{R}^n$ . The considered IDW methods are:

- $I^{\text{IDW}^{\text{Q}}}$ : quadratic Shepard, blending quadratic Taylor polynomials, global,

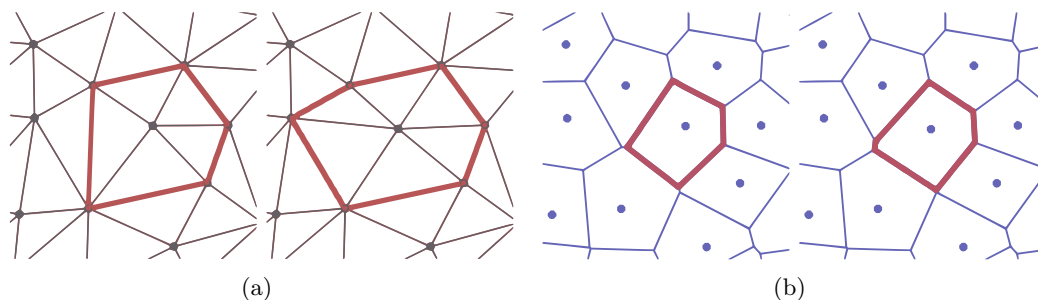


Figure 8.1: (a) A slight perturbation in a triangulation can lead to topological changes. (b) The Voronoi diagram of a point set continuously depends on the coordinates of the points.

- $I^{\text{IDWMQ}}$ : modified quadratic Shepard, as proposed in [FN80], local.

### Local Schemes

Interpolation methods that operate on tessellations of the data set domain are also known as finite element schemes; they were discussed in Section 3.1.4. All tessellation-based approaches share one shortcoming: the interpolants do not depend continuously on the coordinates of the data sites. For any particular tessellation method, there always exists an ambiguous configuration of points that indicates a change in the topology of the tessellation, and for which the constructed interpolant changes discontinuously. An example of such a configuration for the Delaunay triangulation and its counterpart in the Voronoi diagram is given in Figure 8.1.

### Natural Neighbor Interpolation

In contrast to finite element schemes, natural neighbor interpolation is tied to the Voronoi diagram of the data sites and as such also shows a continuous dependency on the coordinates of the data sites. Of the natural neighbor interpolants introduced in Section 3.2, the methods developed in this chapter use the  $C^0$  interpolant based on Sibson coordinates (cf. Section 3.2.5.3), its  $C^1$  generalization of Farin (cf. Section 3.2.7.2), and the  $C^2$  interpolant of Hiyoshi (cf. Section 3.2.7.3).

Furthermore, we consider the partition of unity method on Delaunay circumcircles proposed by Brown (cf. Section 3.2.5.6). This method yields Sibson coordinates for a special choice of rational, non-positive blending functions. We have adopted Brown's idea for extrapolation in Section 8.6.

We will consider the following natural neighbor interpolants in the later discussion of extrapolation.

- $I^{\text{NNS}}$ : Sibson's  $C^0$  interpolant as proposed in [Sib80],
- $I^{\text{NNF}}$ : Farin's  $C^1$  interpolant as proposed in [Far90],
- $I^{\text{NNH}}$ : Hiyoshi's  $C^2$  interpolant as proposed in [HS04].
- $I^{\text{BRO}}$ : Brown's  $C^0$  interpolant as proposed in [Bro97].

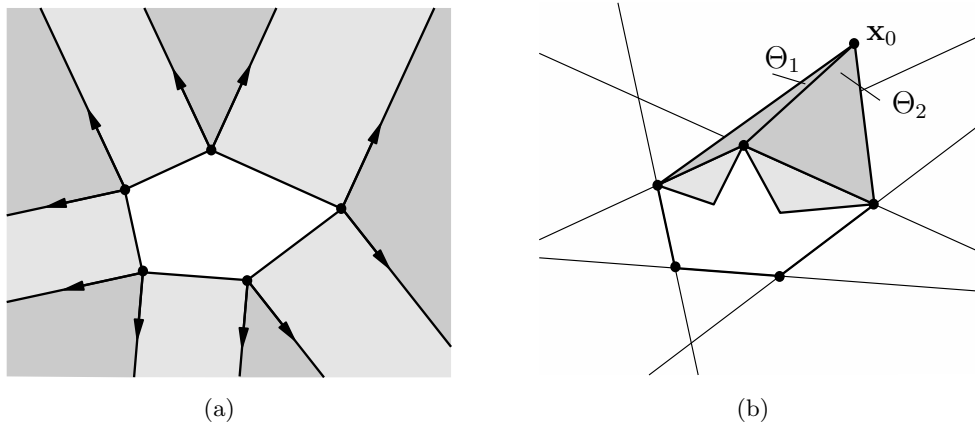


Figure 8.2: (a) Partition of the complement of the convex hull into cones and half-open prisms (wedges and half-open rectangles in 2D), as used in  $E^{\text{AKI}}$  and  $E^{\text{FRA}}$ . (b) To evaluate at  $\mathbf{x}_0$  using  $E^{\text{ALFB}}$ , the functions from the light gray triangles adjacent to the convex hull are extrapolated and mixed based on the angles  $\Theta_1$  and  $\Theta_2$ .

## 8.2.2 Scattered Data Extrapolation

Though the meaning of extrapolation varies from application to application, it essentially refers to the extension of a given interpolant from the “intuitive interior,” to the “intuitive exterior”. Thus, for a given interpolation method  $I$ , defined over the domain  $D(I)$ , we denote by  $E(I)$  the result of applying an extrapolation method  $E$  to extend the interpolant to the *extended domain*  $D(E; I)$ . For practical reasons, we ignore subtle differences in the definition of  $D(I)$  for different interpolation schemes  $I$  and assume interpolation in general to refer to  $\mathcal{C}(\mathbf{X})$ , which allows us to include RBF and IDW methods in a later comparison. Consequently, globally defined interpolants  $I$  are interpreted as the result of applying a dummy extrapolation method  $E^{\text{ID}}(I) = \text{Id}(I)$  that extends  $I$  from  $\mathcal{C}(\mathbf{X})$  to its original domain  $D(I)$ .

The common approaches for scattered data extrapolation, namely ghost point methods, convex hull extension methods, external blending methods, and global interpolation methods are discussed in the following sections.

### Ghost Point Methods

Ghost Point Methods generate new data sites  $\mathbf{X}^G \subset \mathbb{R}^n$ , called ghost points, such that  $\mathcal{C}(\mathbf{X}^G)$  covers an extended domain, over which available interpolants can be applied.

A method to improve piecewise linear interpolation over Delaunay triangulations has been proposed by Lasser and Stüttgen in [LS96] who essentially place ghost points along a rectangle enclosing the data set. Similar methods addressing the extension of Delaunay triangulations past their convex hull have been discussed in [OBSC00], Section 6.3, where it was concluded that the quality of the resulting triangulation depends on the placement and number of “imaginary points”.

A similar conclusion about the number of ghost points was drawn by Alfeld in [Alf84]. He

proposed to construct an interpolant over an extended domain by introducing ghost points leading to a new triangulation of the data sites and ghost points. A smooth triangular interpolant is then constructed over the extended domain as the solution of a global energy minimization, e.g., the minimization of the thin-plate energy, based on the method from [Alf85]. We will refer to this extrapolation approach as  $E^{\text{ALFT}}$ .

In Section 8.7, we introduce a novel ghost point framework for natural neighbor extrapolation. In [Owe93], Owen applied a technique similar to our proposed approach to provide limited extrapolation for Sibson’s  $C^0$  interpolant. By placing the data set in an appropriately sized bounding window and clipping all Voronoi tiles against it, he computes affine weights that allow for interpolation. This method has also been implemented in [Ei08].

### Convex Hull Extension Methods

Convex Hull Extension Methods partition the complement of the convex hull into unbounded regions consisting of cones extending from convex hull vertices and half-open prisms extending from convex hull facets, as shown in Figure 8.2(a). In each region, a function is defined that smoothly joins its neighbors.

One such method was proposed by Akima in [Aki78], which operates solely on the corresponding interpolant proposed in the same publication. In each wedge region, the corresponding Taylor polynomial at the wedge vertex is extrapolated. In the local coordinate system of each rectangular region, a polynomial is constructed based on the special structure of Akima’s triangular interpolation scheme. The result is a  $C^1$  function over  $\mathbb{R}^2$ , and we refer to Akima’s method as  $E^{\text{AKI}}$ .

Franke’s transfinite extension, introduced in [Fra79], is another approach operating on the extension of the convex hull. It can be applied to any interpolant for which Taylor polynomials at the convex hull vertices and normal derivatives along the convex hull edges exist. Franke treats wedge regions like Akima. To evaluate the function at a point  $\mathbf{x}_0$  in a rectangular region,  $\mathbf{x}_0$  is projected to the convex hull edge and the univariate Taylor polynomial determined by the normal derivatives is extrapolated to compute a function value at  $\mathbf{x}_0$ . We refer to this method as  $E^{\text{FRA}}$ .

We finally mention the  $C^{-1}$ -continuous nearest neighbor interpolant which also covers all of  $\mathbb{R}^n$  and which is constant over each Voronoi tile.

### External Blending Methods

External Blending Methods extend particular functions of a piecewise interpolant to overlapping regions in the complement of the convex hull and blend them in a Lagrangian interpolation manner. The relation between convex hull extension and external blending methods is analogous to the relation between FEM interpolants which are constructed to smoothly join local functions along the piecewise domain boundaries, and PoU methods which combine overlapping, local interpolants using smooth blending functions.

Alfeld proposed in [Alf84] a method to extrapolate tessellation-based interpolation schemes past the convex hull. From a point  $\mathbf{x}_0$  outside the convex hull of the data sites, a set of

convex hull edges, and consequently the adjacent elements  $\Omega_i$  of the tessellation, are visible. As shown in Figure 8.2(b), the lines pointing from  $\mathbf{x}_0$  to the vertices of a visible convex hull edge form an angle  $\Theta_i$  at  $\mathbf{x}_0$ . The local function  $\varphi_i$  over each element  $\Omega_i$  can usually be evaluated at  $\mathbf{x}_0$ , thus extrapolating the local function. After the functions of all visible elements have been extrapolated, the function values are mixed using ratios proportional to some power of  $\Theta_i$ . This way, Alfeld is able to smoothly extrapolate any tessellation-based interpolant past the convex hull of the data sites. We will refer to this method as  $E^{\text{ALFB}}$ .

Brown proposed a local coordinate system defined over Delaunay triangulations in [Bro97], for which he gave a construction past the convex hull of the data sites. We introduced the interpolation method in Section 3.2.5.6. For a point contained in the circumcircles of a set of Delaunay triangles  $\{\Omega_i\}_i$ , the barycentric coordinates with respect to each triangle  $\Omega_i$  are computed, and blended in a PoU fashion using smooth, positive blending functions over the corresponding circumcircles. This method naturally extends the coordinates to the interior of circumcircles that reach outside the convex hull. For points even further outside, he proposed to blend the barycentric coordinates with respect to all boundary-adjacent triangles in a Lagrangian interpolation manner based on the distance to the corresponding circumcircles. The resulting generalized barycentric coordinates are then used to build a simple  $C^0$  scattered data interpolant with linear precision. This method will be referred to as  $E^{\text{BRO}}$ .

In this chapter, several new external blending methods are presented. Section 8.4 introduces a slightly modified version of Alfeld's external blending method  $E^{\text{ALFB}}$ , which we will refer to as extrapolation based on weighted triangular exterior coordinates,  $E^{\text{WTE}}$ . Section 8.5 discusses an idea to use Watson's construction for interpolation. Due to problems of this method, we do not consider it for comparison. The method presented in Section 8.6 utilizes notions from projective geometry to enhance Brown's method. It will be referred to as extrapolation based on projective exterior coordinates,  $E^{\text{PEX}}$ .

The summary of local extrapolation methods considered in the sequel is

- $E^{\text{AKI}}$ : Akima's discrete extension [Aki78], a convex hull extension method,
- $E^{\text{FRA}}$ : Franke's transfinite extension [Fra79], a convex hull extension method,
- $E^{\text{ALFB}}$ : Alfeld's blending of visible edges [Alf84], an external blending method,
- $E^{\text{ALFT}}$ : Alfeld's global triangular thinplate minimization [Alf85], which can be interpreted as a ghost point method in which the ghost points globally influence the interpolant.
- $E^{\text{BRO}}$ : Extension of Brown's coordinates in an unstructured point set [Bro97].
- $E^{\text{WTE}}$ : Weighted triangular exterior coordinates, Section 8.4.
- $E^{\text{PEX}}$ : Projective exterior coordinates, Section 8.6.

## Global Methods

Most global interpolation methods naturally extend to all  $\mathbb{R}^n$ . In particular, this entails the RBF and IDW methods introduced in Section 8.2.1. Extrapolation of these inter-

polants requires no modification, and for sake of consistency we denote by  $E^{\text{ID}}(I) = \text{Id}(I)$  the extrapolation method applied to one of these schemes. Away from  $\mathcal{C}(\mathbf{X})$ , RBFs with local and quasi-local support asymptotically approach the supporting polynomial, while global ones generally diverge dramatically. It must be noted that for IDW interpolants  $I$  with compactly supported weight functions,  $D(E^{\text{ID}}, I)$  is finite.

### 8.3 Taxonomy of Scattered Data Extrapolation

To allow an individual comparison of existing extrapolation methods, we adopt and extend the classification of extrapolation approaches introduced by Alfeld in [Alf84], and propose the following criteria, where the first six correspond to Alfeld's. The results of this classification applied to a number of approaches introduced so far is then given in Table 8.3.

**Smoothness** How smooth is  $E(I)|_{D(I)}$ , and how smooth is  $E(I)|_{D(E;I)}$ ?

**Structure** Is the  $E(I)$  of the same structure everywhere, i.e., piecewise polynomial of a certain degree, or are interior and exterior different? A similar structure simplifies the analysis of the method.

**Reproduction Power** Is the class of functions reproduced exactly by  $I$  the same as that reproduced by  $E(I)$ ? All interpolants and extrapolation approaches we considered here possess some sort of polynomial precision, and we use  $P^d$  in the table to indicate reproduction of polynomials of degree  $d$ .

**Finiteness** Is the extended domain  $D(E; I)$  finite or does it cover all  $\mathbb{R}^n$ ?

**Blending Extrapolation** Is  $E(I)|_{D(I)}$  identical to  $I|_{D(I)}$ ? If an extrapolation method simply continues the interpolant where it ceases to be defined, it is not considered blending. Otherwise, it augments the interpolant in a way to improve the joint between  $E(I)|_{D(I)}$  and  $E(I)|_{D(E;I)\setminus D(I)}$ .

**Generality** How large is the class of interpolants  $I$  to which an extrapolation approach  $E$  can be applied? An approach is considered general if it can be applied to more than a single, specific interpolation scheme.

**Similar Results from Similar Input (sim/sim)** Do small perturbations of the input lead to small changes of the output? This property is already important for interpolation alone, and is an indicator for robustness of a particular method.

**Size of Support** How many data sites need to be processed to evaluate the interpolant? Local support (marked L in the table) corresponds to a small, bounded number of data sites in the vicinity of the evaluation position. Convex hull support (marked C in the table) refers to the data sites on the boundary of the convex hull and possibly interior data sites nearby. Global support (marked G in the table) refers to all data sites.

Alfeld also proposed to take the quality of a scheme into account, but because of its subjective flavor we omit this criterion. We note that Brown's method is no competitor for smooth scattered data interpolation, for it was introduced as a local coordinate system in the Delaunay triangulation, and interpolation was added as an afterthought.

| Method            |   | smoothness<br>(in/out)            | structure<br>(in/out) | same rep.<br>(in/out) <sup>(a)</sup> | domain <sup>(b)</sup>       | blending | general          | sim.sim.<br>(in/out) | size of<br>support |
|-------------------|---|-----------------------------------|-----------------------|--------------------------------------|-----------------------------|----------|------------------|----------------------|--------------------|
| Ideal Technique   |   | $C^\infty$                        | same                  | y                                    | $\mathbb{R}^n$              | /        | y                | y                    | small              |
| Triangular        | Akima ( $E^{\text{AKI}}$ ) [Aki78]      | $C^1$                             | diff                  | n                                    | $\mathbb{R}^n$              | n        | n                | n/y                  | L                  |
|                   | Franke ( $E^{\text{FRA}}$ ) [Fra79]     | as $I/C^0$                        | diff                  | n                                    | $\mathbb{R}^n$              | n        | y <sup>(c)</sup> | n                    | L                  |
|                   | Alfeld ( $E^{\text{ALFB}}$ ) [Alf84]    | as $I$                            | diff                  | y                                    | $\mathbb{R}^n$              | n        | y <sup>(d)</sup> | n                    | C                  |
|                   | Brown ( $E^{\text{BRO}}$ ) [Bro97]      | $C^0$                             | diff                  | y ( $P^1$ )                          | $\mathbb{R}^n$              | n        | n                | n                    | C                  |
|                   | Author ( $E^{\text{WTE}}$ ) Section 8.4 | as $I/C^{-1}$                     | diff                  | n                                    | $\mathbb{R}^n$              | n        | y                | n                    | C                  |
|                   | Author ( $E^{\text{PEX}}$ ) Section 8.6 | as $I/C^{-1}$                     | diff                  | n                                    | $\mathbb{R}^n$              | n        | n                | y                    | C                  |
| Ghost<br>Points   | - stat. assigned                        | as $I$                            | same                  | n ( $P^d/P^{d-1}$ )                  | $\mathcal{C}(\mathbf{X}^G)$ | y        | y <sup>(e)</sup> | y                    | L                  |
|                   | - dyn. assigned                         | as $I$                            | diff                  | n ( $P^d/P^{d-1}$ )                  | $\mathbb{R}^n$              | y        | y <sup>(e)</sup> | y                    | C                  |
|                   | - stat. dismissed                       | $C^0$ at $\mathbf{X}^\mathcal{C}$ | diff                  | n ( $P^d/P^0$ )                      | $\mathcal{C}(\mathbf{X}^G)$ | y        | y <sup>(f)</sup> | y                    | L                  |
|                   | - dyn. dismissed                        | $C^0$ at $\mathbf{X}^\mathcal{C}$ | diff                  | n ( $P^d/P^0$ )                      | $\mathbb{R}^n$              | y        | y <sup>(f)</sup> | y                    | C                  |
| Global<br>Interp. | Duchon ( $I^{\text{RBF}^T}$ ) [Duc77]   | $C^\infty$                        | same                  | y ( $P^d$ )                          | $\mathbb{R}^n$              | /        | /                | y                    | G                  |
|                   | Wendland ( $I^{\text{RBF}^W}$ ) [Wen04] | $C^d$                             | diff                  | y ( $P^d$ )                          | $\mathbb{R}^n$              | /        | /                | y                    | L                  |
|                   | Shepard ( $I^{\text{IDW}^Q}$ ) [She68]  | $C^\infty$                        | same                  | y ( $P^0$ )                          | $\mathbb{R}^n$              | /        | y <sup>(g)</sup> | y                    | G                  |
|                   | Nielson ( $I^{\text{IDW}^M^Q}$ ) [FN80] | $C^d$                             | same                  | y ( $P^0$ )                          | finite                      | /        | y <sup>(g)</sup> | y                    | L                  |
|                   | Alfeld ( $I^{\text{ALF}^T}$ ) [Alf85]   | as $I$                            | same                  | y                                    | finite                      | y        | y <sup>(h)</sup> | n                    | as $I$             |

Single entries in columns one and three apply to both interior and exterior, different values are separated by a slash. A sole slash indicates that the criterion does not apply.

(a) We assume that required derivatives are exact.  
(b) We assume  $\mathbb{R}^n$  if extension to higher dimensions is straightforward.  
(c) Cross-boundary derivatives must be available.  
(d) Local functions over  $\mathcal{C}$ -elements must be defined over all  $\mathbb{R}^n$ .  
(e) Applies to any method that can deal with point-based data.  
(f) Applies to any method utilizing affine weights that are entirely determined by the position of the data sites.  
(g) The quadratic approximations at the data sites can be replaced by arbitrary local/global approximations.  
(h) The interpolation scheme must be applicable to incomplete input data.

Table 8.1: Classification of extrapolation methods.

The comparison in Table 8.3 includes the methods  $E^{\text{WTE}}$  from Section 8.4 and  $E^{\text{PEX}}$  from Section 8.6 as well as the ghost point methods introduced in Section 8.7 for completeness and reference. The reader might want to proceed there before closer inspection of the table.

The “similar results from similar input” criterion implicitly assumes that the perturbation of data sites leaves the set of vertices on the convex hull,  $\mathbf{X}^\mathcal{C}$ , unchanged. Otherwise, constructions based on convex hull vertices show discontinuous changes under the perturbations. Among the ghost point methods, the DDC approach is not affected by changes in  $\mathbf{X}^\mathcal{C}$  because the construction of ghost points is independent of the concrete structure of  $\mathbf{X}^\mathcal{C}$ . Furthermore, RBF and IDW schemes are naturally unaffected by changes in  $\mathbf{X}^\mathcal{C}$ .

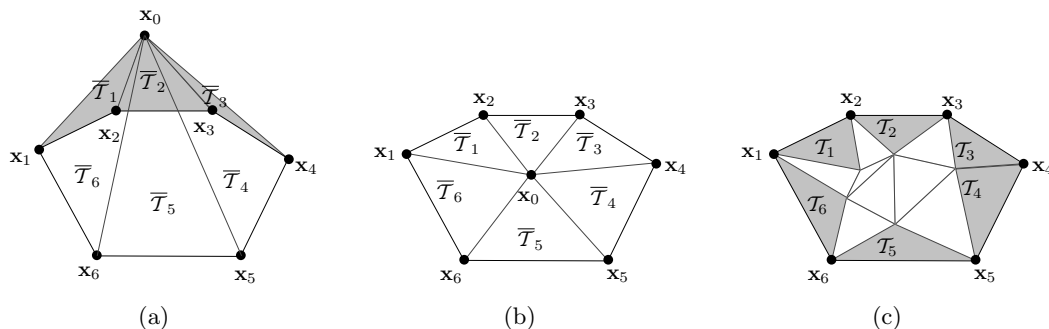


Figure 8.3: The construction used in weighted triangular exterior coordinates. (a) Triangles  $\bar{T}_1, \dots, \bar{T}_6$ , formed by the query position with the convex hull edges. Only  $\bar{T}_1, \bar{T}_2$ , and  $\bar{T}_3$  have positive areas. (b) All triangles  $\bar{T}_1, \dots, \bar{T}_6$  have negative areas. (c) The triangles adjacent to the convex hull.

## 8.4 Weighted Triangular Exterior Coordinates

We describe a convex hull extension method based on weighted triangular exterior coordinates,  $E^{\text{WTE}}$ , that continuously extends the piecewise barycentric coordinates in boundary triangles of some triangulation. The extended local coordinates are combined using affine weights derived from areas of triangles in a support construction, resulting in generalized, non-convex barycentric coordinates  $\lambda^{\text{WTE}}$  which are defined over all  $\mathbb{R}^2$ .

The local coordinates from the interior are continuously extrapolated as local coordinates  $\lambda^{\text{WTE}}$  if the data set has no collinear simplicial facets on its convex hull, and is discontinuous otherwise. This method is a variation of the “Blending by Visible Edges” method proposed by Alfeld for planar scattered data in [Alf84]. His construction does not explicitly provide coordinates outside, but is identical to ours up to the usage of angles instead of areas.

The following presentation assumes a planar setting, but a generalization to higher dimensions is straightforward by considering simplices instead of triangles.

### 8.4.1 Method Description

We assume piecewise linear interpolation over a triangulation  $T$  of the data sites  $\mathbf{X} \subset \mathbb{R}^2$ , composed of triangles  $\{T_i\}_i$  with  $T_i = \Delta(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$  for some  $\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c \in \mathbf{X}$ . We use  $\Delta(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$  to denote the counterclockwise oriented triangle formed by an ordered sequence of points.

Inside the convex hull, the triangle vertex data defines a unique linear function in each triangle. Outside the convex hull, no triangles are defined that could give rise to an interpolating function. In the following, we describe how to evaluate an interpolating function at a point  $\mathbf{x}_0$  at an arbitrary position, with some restrictions on the geometry of the convex hull.

If  $T_i$  is a triangle having an edge on the convex hull, and  $g$  the unique linear function



interpolating the vertex data of  $T_i$ , then  $g$  extrapolates the interpolant in the vicinity of  $T_i$  outside the convex hull. By blending several nearby functions  $g_i$  using an affine combination with coefficients that are smooth with respect to the query position  $\mathbf{x}_0$ , we can define a smooth function outside the convex hull.

Let without loss of generality  $\mathbf{x}_1, \dots, \mathbf{x}_m \in \mathbf{X}$  be the clockwise oriented convex hull vertices, i.e., the counterclockwise boundary of  $\mathbb{R}^2 \setminus \mathcal{C}(\mathbf{X})$ , where the index arithmetic is modulo  $m$ , and summation over  $i$  always addresses all elements  $i = 1, \dots, m$ . We form triangles  $\bar{T}_i := \Delta(\mathbf{x}_0, \mathbf{x}_i, \mathbf{x}_{i+1})$  as depicted in Figure 8.3(a) and (b), and assume  $T_i$  to share edge  $(\mathbf{x}_i, \mathbf{x}_{i+1})$  with  $\bar{T}_i$  as depicted in Figure 8.3(c). By  $A_i$  we denote the signed area of  $\bar{T}_i$ , where  $A_i$  is positive for vertices with counterclockwise orientation, and by  $A_i^+ := \max(0, A_i)$  the restriction of  $A_i$  to positive values. If  $\mathbf{x}_0 \in \mathcal{C}(\mathbf{X})$ , all  $A_i^+$  are zero. The resulting coefficients are

$$\alpha_i := A_i^+ / \sum_j A_j^+, \quad (8.1)$$

where we define  $\alpha_i := 0$  if both numerator and denominator are zero. With these affine weights, the interpolating function outside the convex hull is given by

$$\sum_i \alpha_i g_i(\mathbf{x}_0). \quad (8.2)$$

Instead of evaluating an interpolant, we can compute generalized barycentric coordinates of  $\mathbf{x}_0$  with respect to the vertices of the triangles  $T_i$ . If the barycentric coordinates  $\lambda_j^i(\mathbf{x}_0)$  with respect to  $T_i$  are defined by

$$\mathbf{x} = \sum_{\mathbf{x}_j \in T_i} \lambda_j^i(\mathbf{x}) \mathbf{x}_j,$$

the local coordinates are derived by rearranging,

$$\begin{aligned} \mathbf{x}_0 &= \sum_i \alpha_i \mathbf{x}_0 \\ &= \sum_i \alpha_i \sum_{\mathbf{x}_j \in T_i} \lambda_j^i(\mathbf{x}_0) \mathbf{x}_j \\ &= \sum_{\mathbf{x}_j \in T_i} \left( \sum_i \alpha_i \lambda_j^i(\mathbf{x}_0) \right) \mathbf{x}_j \\ &=: \sum_{\mathbf{x}_j \in T_i} \lambda_j^{\text{WTE}} \mathbf{x}_j. \end{aligned}$$

Because  $\boldsymbol{\lambda}^{\text{WTE}}$  comprises continuous, local coordinates, we can conclude that the interpolating function

$$\sum_j \lambda_j^{\text{WTE}}(\mathbf{x}_0) z_j$$

has linear precision for  $z_j \in \mathbb{R}$ . Further, derivatives at the data sites can be interpolated by Sibson's, Farin's or Hiyoshi's method without any modification.

### 8.4.2 Alfeld's Extrapolation

The method introduced by Alfeld in [Alf84] is structurally identical to ours when based on piecewise linear interpolation over some triangulation of the data set. It differs from ours in that weights are derived from the angles at the query position, where  $\Theta_i(\mathbf{x}_0)$  denotes the signed angle at vertex  $\mathbf{x}_0$  of the triangle  $T_i$  with its sign equal to that of the area  $A_i$ , and  $\Theta_i^+(\mathbf{x}_0) := \max(0, \Theta_i(\mathbf{x}_0))$  as above (cf. Figure 8.2(b)). Then, the extrapolated function defined by Alfeld's method  $E^{\text{ALF}}$  is

$$\sum_i \Theta_i^+(\mathbf{x}_0) g_i(\mathbf{x}_0) / \sum_i \Theta_i^+(\mathbf{x}_0),$$

where  $g_i(\mathbf{x}_0)$  is again the linear function interpolating the data of the corresponding convex hull triangle.

### 8.4.3 Increasing the Continuity Away from the Convex Hull

In the definition of  $\alpha_i$  in (8.1) the denominator is piecewise linear with a derivative discontinuity at zero. By adding an exponent  $\beta$  such that

$$\alpha_{i,\beta} := \frac{(A_i^+)^{\beta}}{\sum_j (A_j^+)^{\beta}},$$

the affine coefficients can be made  $C^{\beta-1}$ -continuous away from the convex hull - reasoning that applies to Alfeld's construction as well. Extrapolation using both  $E^{\text{WTE}}$  and  $E^{\text{ALFB}}$  is illustrated in Figure 8.4 for  $\beta = 1, 2, 3$ .

### 8.4.4 Restrictions and Shortcomings

Obviously,  $A_i$  is a linear function of  $\mathbf{x}_0$ , and  $\alpha_i$  is piecewise rational. Whenever  $\mathbf{x}_0$  lies on a line supporting a convex hull edge, the corresponding exterior triangle has zero area. Problems arise when all areas, and thus the denominator in (8.1), converge toward zero, leading to poles and possibly discontinuities of  $\alpha_i$ .

We now examine the case when all non-zero areas converge toward zero, which can lead to the artifact visible in Figure 8.5(c). Let without loss of generality  $A_1^+, \dots, A_m^+$  denote all non-zero areas, with  $A_i^+ \rightarrow 0$ ,  $i = 1, \dots, m$ , as  $\mathbf{x}_0 \rightarrow \mathbf{q}$  for some  $\mathbf{q}$  on the convex hull. Then,

$$\lim_{\mathbf{x}_0 \rightarrow \mathbf{q}} \alpha_i = \lim_{\mathbf{x}_0 \rightarrow \mathbf{q}} \frac{A_i^+}{\sum_j A_j^+} = \lim_{\mathbf{x}_0 \rightarrow \mathbf{q}} \frac{\frac{d}{d\mathbf{n}_i} |_{\mathbf{q}} A_i^+}{\sum_j \frac{d}{d\mathbf{n}_i} |_{\mathbf{q}} A_j^+}$$

by the linearity of the derivative. Since  $dA_i^+/d\mathbf{n}_i|_{\mathbf{q}}$ , where  $\mathbf{n}_i$  is the outward normal vector at  $\mathbf{q}$ , is well defined and negative, we get, by l'Hôpital's rule for one-sided limits, that every coefficient converges to a value  $\alpha_i \geq 0$ .

If two or more line segments, say  $(\mathbf{x}_1, \mathbf{x}_2)$  and  $(\mathbf{x}_2, \mathbf{x}_3)$  on the convex hull are collinear, the

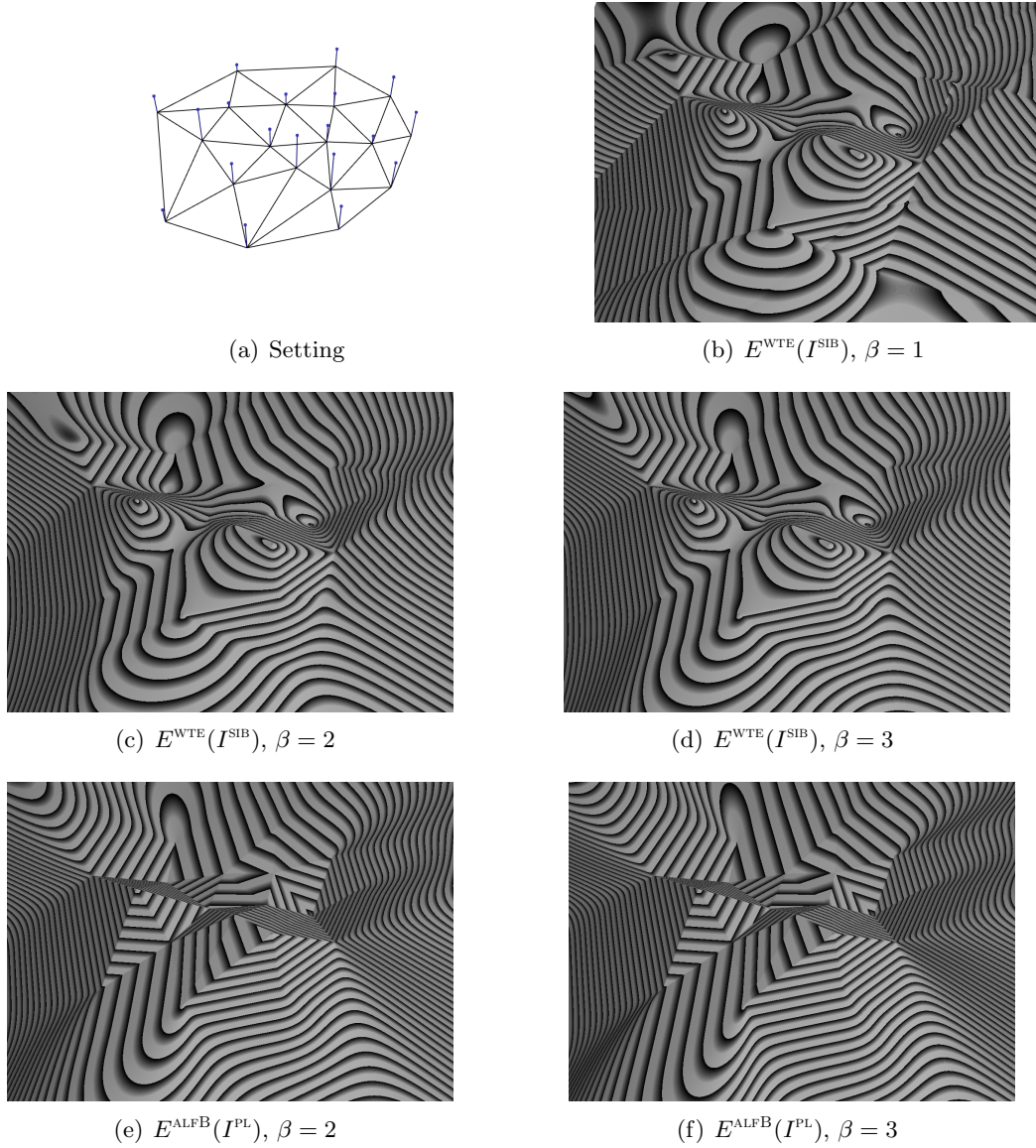


Figure 8.4: (a) A very well-behaved data set with homogeneously sized triangles along the convex hull. (b)-(d)  $E^{\text{WTE}}(I^{\text{SIB}})$  with exponents of  $\beta = 1$ ,  $\beta = 2$ , and  $\beta = 3$ , resp. (e)-(f)  $E^{\text{ALFB}}(I^{\text{PL}})$  with exponents of  $\beta = 2$  and  $\beta = 3$ , resp.

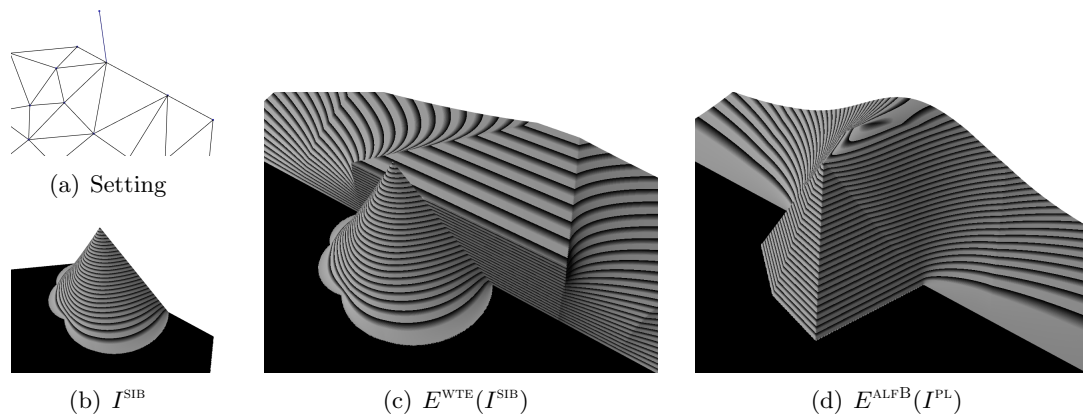


Figure 8.5: (a) Deliberately chosen degenerate setting: the four data sites in the top-right of the images are collinear with conflicting data values. (b) Sibson interpolation inside the convex hull. (c)  $E^{\text{WTE}}(I^{\text{SIB}})$  with an exponent of  $\beta = 1$ . (d)  $E^{\text{ALFB}}(I^{\text{PL}})$  with an exponent of  $\beta = 1$ .

corresponding areas  $A_1^+$  and  $A_2^+$  both converge to zero as  $\mathbf{x}_0$  approaches any point  $\mathbf{q}$  on these line segments. Because  $A_1^+$  and  $A_2^+$  are linear functions of the same perpendicular distance to the line segments, the limit of  $\alpha_1$  and  $\alpha_2$  is constant and positive along these segments. In the limit from outside,  $E^{\text{WTE}}$  therefore gives a single linear function over both edges, which conflicts with the two different linear functions on these boundary edges in the interior. This discontinuous behavior can be a severe drawback in case of structured or partially structured data sets.

We note that the alternative approach by Alfeld in [Alf84] uses the angles of the triangles at the query position, and does not have this problem: as the query position approaches an edge, the corresponding angle approaches  $\pi$ , while the remaining weights approach zero. This can be verified in Figure 8.5(d).

The application of  $E^{\text{WTE}}$  results in general in a function that is only  $C^0$  across the convex hull of the data set if interpolation other than piecewise linear is applied inside the convex hull. E.g., the gradient of natural neighbor interpolation is in general different from that of the piecewise linear interpolant, to which  $E^{\text{WTE}}$  interpolates on the convex hull.

Another undesired effects is related to the “similar output from similar input” criterion stated in Section 8.3, which can be verified for Alfeld’s construction in Section 8.8.3.

### 8.4.5 Implementation Notes

The triangles involved in the computation of  $\lambda^{\text{WTE}}$  can be found by traversing the convex hull and checking for positive volumes of the triangles  $\bar{T}_i$ . Given a half-edge data structure, this is easily accomplished.

In higher dimensions, the problem with collinear facets is equivalent to the two-dimensional case.

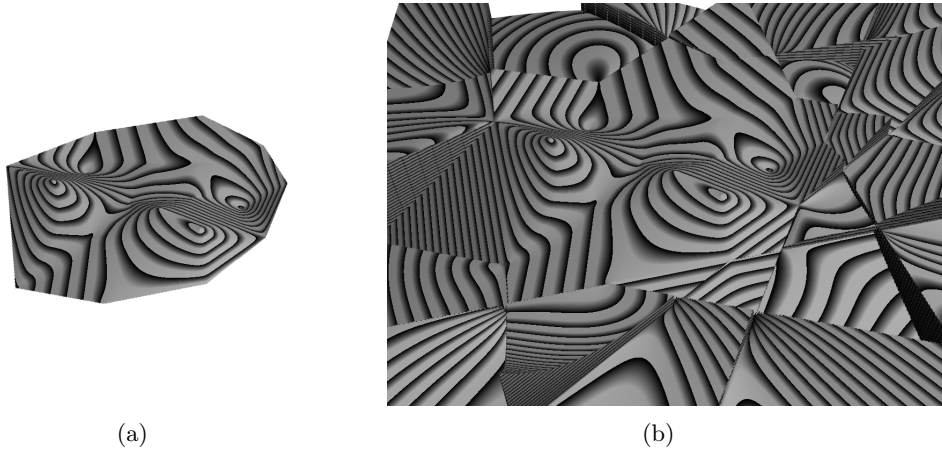


Figure 8.6: (a) Isocontour visualization for Sibson’s interpolant. (b) The same data set extrapolated by evaluating Watson’s construction based on arbitrarily chosen sets of interior neighbors.

## 8.5 Extrapolating Watson’s Construction

As described in Section 3.2.5.4, Watson proposed to compute the areas involved in the definition of Sibson coordinates by a signed triangle decomposition (cf. [Wat92], p. 81). Besides its straightforward application in higher dimensions, this method gives an explicit construction of the rational function that Sibson coordinates take inside the regions of constant neighborhood, which enables us to argue for a construction of external barycentric coordinates.

Despite the numerical issues pointed out previously, Watson’s observation suggests an interesting line of thought. Everywhere inside the region of constant neighborhood, as shown in Figure 3.9(b), the sequence of neighbors and the traversal order remains the same. By keeping this computation sequence fixed, but changing  $\mathbf{x}_0$ , we are efficiently evaluating the rational function  $\lambda^{\text{SIB}}(\mathbf{x}_0)$  outside the region of constant neighborhood.

As long as the query position is inside the convex hull, the neighborhood is well-defined, and the areas of constant neighborhood join continuously along their bounding circular arcs. If we find a way to keep the set of neighbors fixed as the query position leaves the convex hull, such that a vertex only leaves or enters the set of neighbor if it is zero, we continue the smooth, rational function from inside to the outside.

Unfortunately, we did not find a way to define a neighborhood that depends on a position outside the convex hull, agrees with the natural neighborhood on the convex hull and changes in a compatible manner as the position traverses around the data set. The shortcomings observed in our experiments are depicted in Figure 8.6.

## 8.6 Projective Exterior Domain Coordinates

The next approach utilizes ideas from projective geometry in an effort to form a unifying basis in which natural neighbor based algorithms can be applied inside as well as outside the convex hull.

The main feature of projective geometry is the possibility to have points at infinity. We add points at infinity to extend the data set past its convex hull and define some meaningful notion of circumcircle for them. Then, we apply Brown's construction of non-convex coordinates in point clouds, which combines triangular barycentric coordinates from natural neighbors using blending functions over the circumcircles. The coordinates we thereby get are no longer barycentric coordinates with respect to points, but in an affine basis involving the perpendicular of a convex hull edge, which prevents direct further use for interpolation. We can, however, transform them into affine coefficients for vertices to construct a smooth interpolant.

This approach has the same limitation as the one presented in Section 8.4, i.e., there must not be collinear points on the convex hull of  $\mathbf{X}$ .

### 8.6.1 Extrapolation with Brown Coordinates

Brown already devised a construction that allows the computation of barycentric coordinates outside the convex hull, which are continuous everywhere but at some isolated points (cf. Section 3.2.5.6).

If  $\mathbf{x}_i$  is contained in the circumcircle of some triangles, his construction produces generalized barycentric coordinates even outside the convex hull of the data sites. This already extends the domain of definition for the coordinates to the union of circumcircles,

$$\Omega^{\text{ext}} = \bigcup_{T_i \in T} \mathcal{C}(\circ(T_i)),$$

where  $\mathcal{C}(\circ(T_i))$  is the interior of the circumcircle of  $T_i$ . The boundary of the circumcircles of these triangles form the boundary of the extended domain of definition.

Outside  $\Omega^{\text{ext}}$ , Brown proposes to interpolate the barycentric coordinates with respect to the circumcircles of triangles adjacent to the convex hull of  $\mathbf{X}$ , which we assume without loss of generality to be  $T_1, \dots, T_k$ . With the blending functions now defined as

$$\psi_i(\mathbf{x}) = \prod_{j \in \{1, \dots, k\} \setminus \{i\}} (\|\mathbf{x} - \mathbf{c}_j\|^2 - r_j^2), \quad (8.3)$$

which is *Lagrangian interpolation* where the polynomials are taken over the distances to the circumcircles.

Brown already observed discontinuities of the coordinate functions at the boundary of  $\Omega^{\text{ext}}$  if two circumcircles intersect in a point that is not a vertex. Such a situation is displayed in Figure 8.9(c). In case his construction is applied for interpolation, he proposes to add another data site at the position of the discontinuity and assign it the mean value of the

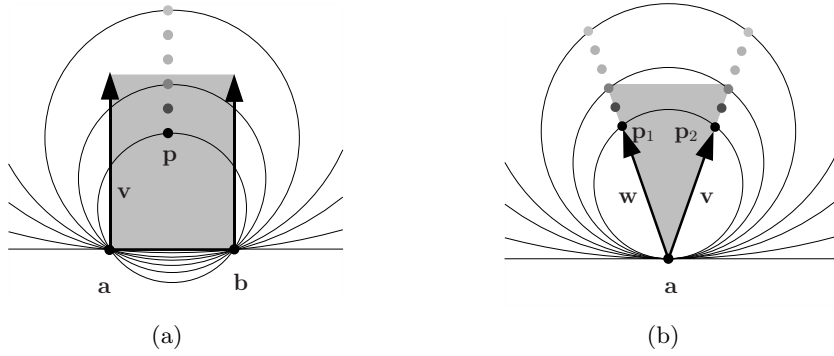


Figure 8.7: Illustration of the reasoning behind our definition of the circumcircle of an infinite triangle. The points  $\mathbf{p}$ ,  $\mathbf{p}_1$ , and  $\mathbf{p}_2$  help describe the limit process for circumcircles as they converge to the points at infinity corresponding to  $\mathbf{v}$  and  $\mathbf{w}$ . (a) As  $\mathbf{p}$  moves toward the infinite point represented by  $\mathbf{v}$ , the circumcircle becomes the upper half-plane. (b) As  $\mathbf{p}_1$  and  $\mathbf{p}_2$  move equally towards the infinite points represented by  $\mathbf{v}$  and  $\mathbf{w}$ , the circumcircle becomes the half-plane orthogonal to the angle bisector between  $\mathbf{v}$  and  $\mathbf{w}$ .

two disagreeing linear functions there.

Away from the  $\Omega^{\text{ext}}$ , the function has a very smooth shape thanks to the boundary-global polynomial interpolation.

**Remark 8.1** *It seems that Brown accidentally omitted in the definition of weights for the exterior blending in (8.3) to add a power similar to that in (8.4), so that we changed it in our implementation to*

$$\psi_i(\mathbf{x}) = \prod_{j \in \{1, \dots, k\} \setminus \{i\}} (\|\mathbf{x} - \mathbf{c}_j\|^2 - r_j^2).$$

## 8.6.2 Extension of Circumcircles to Points at Infinity

In the following, we borrow concepts from projective geometry, but skip a rigorous application of projective geometry for the sake of presentation's clarity.

We will consider points at infinity, which correspond to vectors, denoted by  $\mathbf{v}_i$ , and regular points, denoted by  $\mathbf{x}_i$ , and use them to define *unbounded triangles* in the following.

### 8.6.2.1 Unbounded Triangles and Circumcircles

In Section 2.3.2, we introduced the *cone* spanned by a set of vectors as  $\text{cone}(\mathbf{v}_1, \dots, \mathbf{v}_m) = \{\sum_i \alpha_i \mathbf{v}_i : \alpha_i \in \mathbb{R}, \alpha_i \geq 0\}$ .

**Two points and a vector:** Consider two points  $\mathbf{a}, \mathbf{b} \in \mathbb{R}^2$  and a vector  $\mathbf{v} \in \mathbb{R}^2$  orthogonal to the segment  $\overline{\mathbf{ab}}$  as in Figure 8.7(a). These form a degenerate triangle

$$\sqcup(\mathbf{a}, \mathbf{b}, \mathbf{v}) := \overline{\mathbf{ab}} \oplus \text{cone}(\mathbf{v}),$$

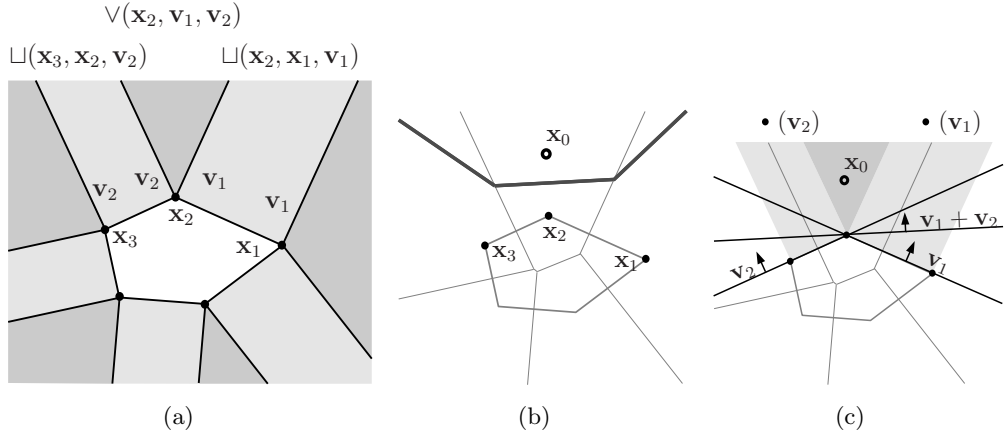


Figure 8.8: (a) A convex polygon and the infinite triangles formed by the vertices and the edge normals, the ones adjacent to  $\mathbf{x}_2$  explicitly labeled. (b) Natural neighbors of  $\mathbf{x}_0$ . (c) Triangles in conflict with  $\mathbf{x}_0$ , whose vertices are a superset of the natural neighbors of  $\mathbf{x}_0$ .

corresponding to the open, rectangular polygon bounded by the rays  $\mathbf{a} + \alpha \mathbf{v}$  and  $\mathbf{b} + \alpha \mathbf{v}$ ,  $\alpha \geq 0$ , and the segment  $\overline{\mathbf{ab}}$ . We define the *unbounded circumcircle*  $\mathcal{O}^{\sqcup}(\mathbf{a}, \mathbf{b}, \mathbf{v})$  as the half-space  $\{ \mathbf{x} : (\mathbf{x} - \mathbf{a})^T \mathbf{v} \geq 0 \}$ .

**A point and two vectors:** Now, consider a point  $\mathbf{a} \in \mathbb{R}^2$  and two unit vectors  $\mathbf{v}, \mathbf{w} \in \mathbb{R}^2$ , as in Figure 8.7(b). The corresponding degenerate triangle is

$$\mathcal{V}(\mathbf{a}, \mathbf{v}, \mathbf{w}) := \mathbf{a} \oplus \text{cone}(\mathbf{v}, \mathbf{w}).$$

bounded by the rays  $\mathbf{a} + \alpha \mathbf{v}$  and  $\mathbf{a} + \alpha \mathbf{w}$ ,  $\alpha \geq 0$ . We define the unbounded circumcircle  $\mathcal{O}^{\vee}(\mathbf{a}, \mathbf{v}, \mathbf{w})$  as the half-space  $\{ \mathbf{x} : (\mathbf{x} - \mathbf{a})^T (\mathbf{v} + \mathbf{w}) \geq 0 \}$ .

### 8.6.2.2 Unbounded Natural Neighbors

We partition the complement of the convex hull,  $\mathbb{R}^2 \setminus \mathcal{C}(\mathbf{X})$ , into a set of unbounded triangles. If  $\mathbf{x}_1, \dots, \mathbf{x}_k$  is the clockwise sequence of boundary vertices, where we assume a circular index  $k + 1 = 1$ , and  $\mathbf{v}_1, \dots, \mathbf{v}_k$  are the corresponding sequence of outward-pointing normals of the edges  $\overline{\mathbf{x}_1 \mathbf{x}_2}, \dots, \overline{\mathbf{x}_k \mathbf{x}_1}$ , then

$$T^\infty = \bigcup_{i=1, \dots, k} \{ \mathcal{U}(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{v}_i) \} \cup \{ \mathcal{V}(\mathbf{x}_i, \mathbf{v}_{i+1}, \mathbf{v}_i) \}$$

is a partition of  $\mathbb{R}^2 \setminus \mathcal{C}(\mathbf{X})$  into convex, open polyhedra.

In Section 2.3.4 it was mentioned that a triangle of a Delaunay triangulation of a point set  $\mathbf{X}$  is in conflict with a point  $\mathbf{x} \notin \mathbf{X}$  if its circumcircle contains that point. Furthermore, the vertices of all triangles in conflict with  $\mathbf{x} \in \mathcal{C}(\mathbf{X})$  form the set its natural neighbors.

We extend this to points outside the convex hull by calling an unbounded triangle *in conflict* with a point  $\mathbf{x}$  if it is in the corresponding unbounded circumcircle. From Figure 8.8 is easy to verify that the vertices of all unbounded triangles in conflict is composed of the



natural neighbors of  $\mathbf{x}$  plus the set of edge normals from which the unbounded triangles are built.

### 8.6.2.3 Local Coordinates in Unbounded Triangles

In an unbounded triangle made of two points  $\mathbf{a}$ ,  $\mathbf{b}$ , and the vector  $\mathbf{v}$ , an affine basis of  $\mathbb{R}^2$  is given by  $\mathbf{a}; \mathbf{b} - \mathbf{a}, \mathbf{v}$ . Every point has the unique representation

$$\mathbf{x} = \mathbf{a} + \lambda_1(\mathbf{b} - \mathbf{a}) + \lambda_2\mathbf{v} = (1 - \lambda_1)\mathbf{a} + \lambda_1\mathbf{b} + \lambda_2\mathbf{v}$$

where  $1 - \lambda_1, \lambda_1$  are affine coefficients, but  $\lambda_2$  is not.

In an unbounded triangle made of one point  $\mathbf{a}$ , and two linearly independent vectors  $\mathbf{v}$ ,  $\mathbf{w}$ , an affine basis of  $\mathbb{R}^2$  is given by  $\mathbf{a}; \mathbf{v}, \mathbf{w}$ , where every point trivially has the unique representation

$$\mathbf{x} = \mathbf{a} + \lambda_1\mathbf{v} + \lambda_2\mathbf{w}.$$

### 8.6.3 Extrapolation Using Brown's Approach on Unbounded Circumcircles

With unbounded triangles and circumcircles we can use Brown's construction to build an interpolant that agrees with Sibson's interpolant inside the convex hull and continuously extends outside almost always, shown in Figure 8.9(d). The interpolant shows artifacts in general if there are sliver triangles on the convex hull because it honors the interpolant inside the convex hull, and will be discontinuous for collinear data sites on the convex hull.

The interpolant is evaluated for a point  $\mathbf{x} \in \mathbb{R}^2$  using affine weights  $\boldsymbol{\lambda}$  that agree with Sibson coordinates inside the convex hull but are no barycentric coordinates outside. If  $\mathbf{x} \in \mathcal{C}(\mathbf{X})$ , then we define  $\boldsymbol{\lambda}(\mathbf{x}) := \boldsymbol{\lambda}^{\text{SIB}}(\mathbf{x})$ , which can be computed using weight functions (3.7). We now focus on the construction of the affine weights  $\boldsymbol{\lambda}(\mathbf{x})$  for  $\mathbf{x} \notin \mathcal{C}(\mathbf{X})$ .

Let  $T$  be the set of Delaunay triangles and  $T^\infty$  the set of unbounded triangles of a Delaunay triangulation of  $\mathbf{X}$ . We define  $\mathbf{X}^\infty := \mathbf{X} \cup \{\mathbf{v}_1, \dots, \mathbf{v}_k\} \subset \mathbb{R}^n$  as the union of data sites and points at infinity. Outside the convex hull, we determine the set of triangles in conflict with  $\mathbf{x}$ , say  $T_1, \dots, T_k \in T \cup T^\infty$ , where we omit all unbounded triangles formed by a point and two vectors. For  $\mathbf{x} \in \Omega^{\text{ext}}$ , i.e., the union of circumcircles, there are triangles  $T_i \in T$  in conflict, and we define  $\psi_i$  like in (3.7), thus extending the natural neighbor coordinates to  $\Omega^{\text{ext}} \setminus \mathcal{C}(\mathbf{X})$ . Note that  $\psi^{\text{SIB}}$  is negative outside the convex hull.

In every remaining triangle  $\sqcup(\mathbf{a}, \mathbf{b}, \mathbf{v}) \in T^\infty$ , we compute local coordinates  $\lambda_a, \lambda_b, \lambda_v$  for  $\mathbf{x}$  as proposed in Section 8.6.2.3. For interpolation purposes, we need to assign values to the points at infinity. Considering that  $\mathbf{v}$  has been constructed from  $\mathbf{a}$  and  $\mathbf{b}$  by

$$\mathbf{v} = \begin{bmatrix} b_2 - a_2 \\ a_1 - b_1 \end{bmatrix} = (\mathbf{b} - \mathbf{a})^\perp,$$

we assume an even contribution of  $\mathbf{a}$  and  $\mathbf{b}$  to the point at infinity represented by  $\mathbf{v}$ . We can move the contribution of  $\mathbf{v}$  into the affine weights for the vertices, given by

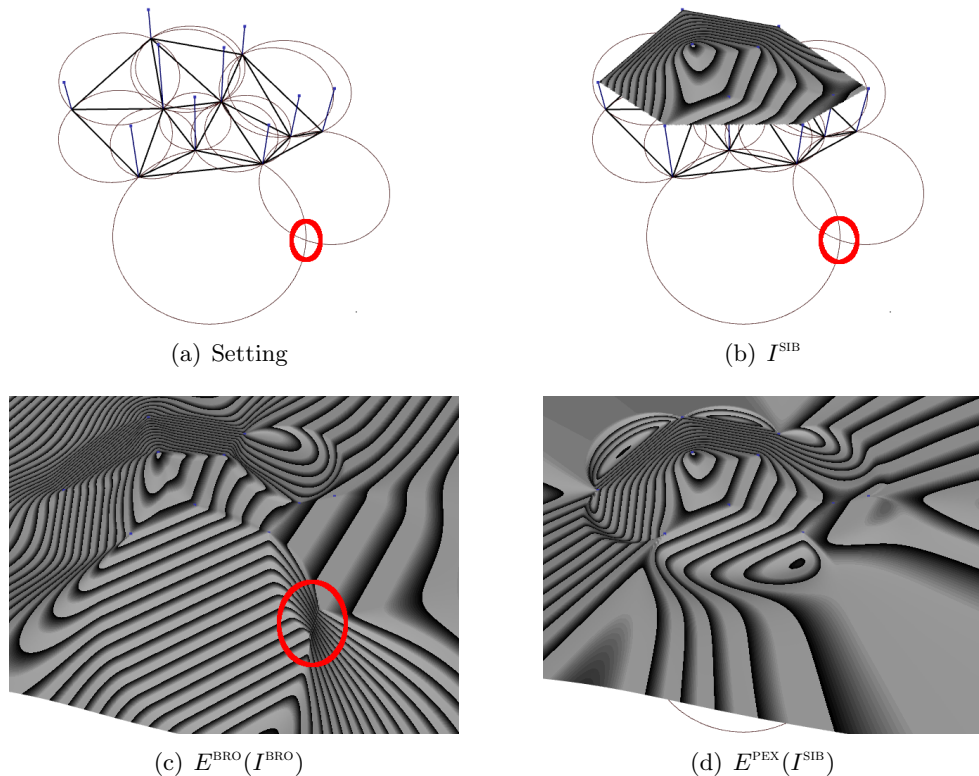


Figure 8.9: Comparison of Brown's method and extrapolation based on projective exterior coordinates. The circle marks the region where a discontinuity arises in Brown's approach is not present with projective exterior coordinates. (a) Data set. (b) Sibson's interpolant. (c) Brown's extrapolation method. (d) Extrapolation based on projective exterior coordinates.

$$\alpha_a := (\lambda_a + \lambda_v/2)/(\lambda_a + \lambda_b + \lambda_v) \text{ and } \alpha_b := (\lambda_b + \lambda_v/2)/(\lambda_a + \lambda_b + \lambda_v).$$

Since we assume unit vectors,  $\lambda_v$  is the distance of  $\mathbf{x}$  to the circumcircle, which is the line segment supporting  $\overline{\mathbf{ab}}$ . Then, the blending function for a user control power  $\beta$  associated with the affine weights  $\alpha_a$  and  $\alpha_b$  is

$$\psi_i(\mathbf{x}) = (\lambda_v)^\beta,$$

where we choose  $\beta = 3$  to yield  $C^2$  continuous coordinates outside the convex hull.

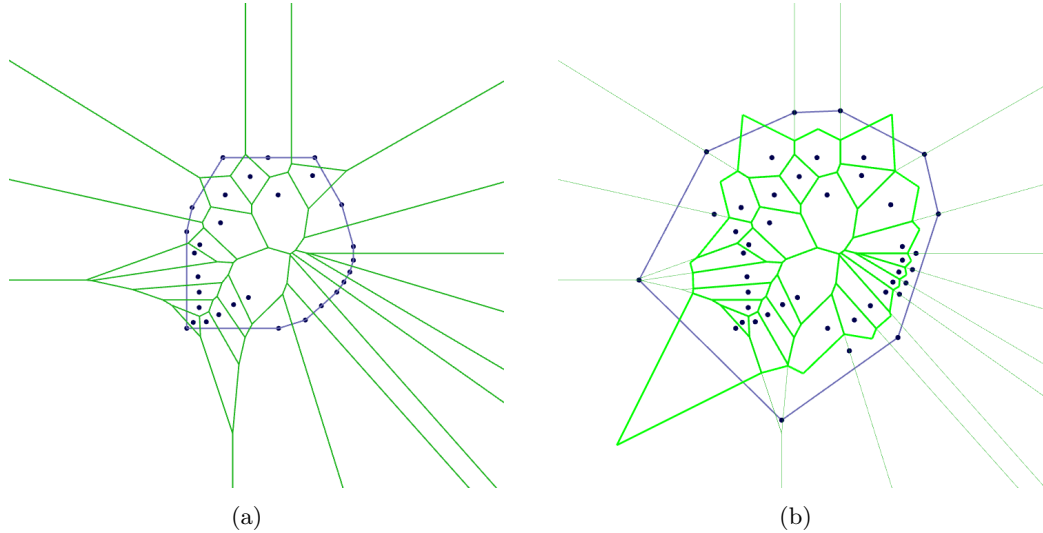


Figure 8.10: (a) Sites  $\mathbf{X}$  with their convex hull and the Voronoi diagram drawn as solid lines, showing unbounded Voronoi tiles for vertices on the convex hull. (b) The same setting with some ad-hoc choice of ghost points  $\mathbf{X}^G$  added, both extending the convex hull and bounding the previously unbounded Voronoi tiles.

## 8.7 Ghost Points for Natural Neighbor Interpolation

In this section we present our ghost point-based modification of natural neighbor interpolants. We first discuss the general idea in Section 8.7.1, then introduce the concepts of dismissed and assigned ghost points in Section 8.7.2, and present concrete placement strategies in Section 8.7.3 and Section 8.7.4.

Our method concentrates on discrete data, i.e., given at points. While it is possible to extend it to transfinite data, we refrain from discussion of such methods in this paper.

### 8.7.1 Ghost Point Idea

The ghost point concept entails the modification / enrichment of the data set such that an interpolation method that is initially defined only in a limited domain can now be applied on a larger domain. An example of ghost points along with their influence on the Voronoi diagram of a set of sites is shown in Figure 8.10. Interpolation by the ghost point method is comprised of the following steps.

1. Construct ghost points  $\mathbf{X}^G \subset \mathbb{R}^2$  from  $\mathbf{X}$ , where  $\mathbf{X}^G \cap \mathbf{X} = \emptyset$ .
  2. Compute any natural neighbor coordinates  $\lambda^G$  for  $\mathbf{x}_0$  with respect to  $\mathbf{X} \cup \mathbf{X}^G$ .
- 3a. *Dismissed Ghostpoints* (Option 1)
1. Dismiss the coefficients from  $\lambda^G$  corresponding to ghost points and renormalize, yielding affine coefficients  $\alpha$ .
  2. Combine values from  $Z$  using  $\alpha$ , or blend the extrapolated Taylor polynomials of the corresponding points.

3b. *Assigned Ghostpoints* (Option 2)

1. Assign values and derivatives to the points in  $\mathbf{X}^G$ .
2. Proceed with an arbitrary natural neighbor interpolation scheme.

Crucial are the placement of the ghost points in Step 1, and the assignment of values to the ghost points in Step 3b.

We now describe the main issues of extending natural neighbor interpolants using ghost points and relate them to the rationale behind our ghost point framework.

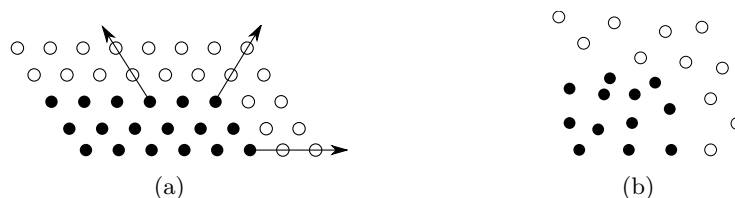


Figure 8.11: (a) Data sites (solid) are used to place ghost points (hollow) in a structured setting. (b) Arbitrary ghost point choice in an unstructured setting.

**Rigid Invariance** The placement of ghost points should be invariant under rigid transformations. In a structured setting as shown in Figure 8.11(a), where data is distributed over a grid, ghost points can trivially be generated by extending the grid. Since such an extension is not available in scattered data as shown in Figure 8.11(b), we anchor the ghost point construction at the vertices and edges of the convex hull, which makes the construction invariant under rigid transformations.

**Finiteness** Any concrete choice of ghost points leads to an extended domain that is again finite. To allow evaluation of the interpolant at an arbitrary position outside the convex hull, the ghost point construction must either recursively continue until the position is covered, or the placement of ghost points must be dynamic in that it takes the actual evaluation position into account. In this paper, we only focus on the second approach of dynamic ghost points.

One major property of natural neighbor interpolation is its continuous dependence on the coordinates of the data sites. By making the ghost point coordinates depend smoothly on the position of the evaluation position, we maintain the continuity of the interpolant even in its extended domain.

**Artifact Removal** Tessellation-based interpolants suffer severe artifacts in case of slightly concave data site distributions at the boundary because of long, skinny triangles or polygons, as shown in Figure 8.27(a) and (b). Natural neighbor interpolants have similar problems due to the linear precision on the convex hull, shown in Figure 8.27(c), where the following observation is useful. The “perfect” data site distribution for natural neighbor interpolation is completely homogeneous, which roughly means the same density of neighbors in every direction. The corresponding “degenerate” case occurs on the convex hull, where the outside completely lacks neighbors. The seamless transition between

these two extrema is a core advantage of natural neighbor interpolation that makes it cope so well with very inhomogeneous site distributions. The distance from the convex hull at which ghost points are placed plays a crucial role in overcoming artifacts. We place ghost points such that the original natural neighbor interpolant is augmented near the boundary of the convex hull, where the local data site density should be considered in the computation of an offset distance, and try to provide ghost points that mimic the perfect setting.

The boundary artifacts have in part been overcome by Cueto et al. in [CDG00, CCD02], who applied density-scaled  $\alpha$ -shapes that allow the restriction of the domain to a concave shape in which undesired triangles are omitted.

**Reproduction Power** While ghost point positions determine the domain over which an extended interpolant is defined, the interpolant itself also depends on the values at the ghost points. In how far the reproduction power of the interpolant is preserved by the ghost point framework depends on the generated values. To this end, we propose two solutions which we coin “dismissed” and “assigned” ghost points.

### 8.7.2 Assigned vs. Dismissed Ghost Points

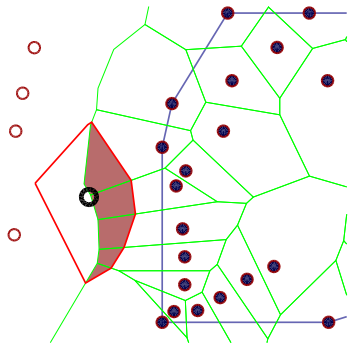


Figure 8.12: Illustration of sub-tiles used in the dismissed ghost point method. The areas of the shaded polygons are used to determine the affine weights by which to mix the values at the corresponding data sites (drawn solid).

By default, ghost points do not carry any data besides their coordinates. Two options exist to deal with this issue when evaluating a natural neighbor interpolant that builds on these data.

The first option, called “dismissed ghost points,” proceeds as follows. If  $\lambda = (\lambda_1, \dots, \lambda_m) \in \mathbb{R}^m$  are natural neighbor coordinates of  $\mathbf{x}$  in

$$\mathbf{x} = \sum_{i=1}^m \lambda_i(\mathbf{x}) \mathbf{x}_i, \quad \mathbf{x}_1, \dots, \mathbf{x}_k \in \mathbf{X}, \mathbf{x}_{k+1}, \dots, \mathbf{x}_m \in \mathbf{X}^G,$$

and  $k > 0$ , then  $\gamma = (\lambda_1, \dots, \lambda_k) / (\lambda_1 + \dots + \lambda_k) \in \mathbb{R}^k$  is a set of affine coefficients that result from ignoring the ghost point contributions. This situation is shown in Figure 8.12. Because they fulfill the Lagrange property  $\gamma_i(\mathbf{x}_j) = \delta_{ij}$ , these affine coefficients can be used for interpolation in  $\psi(\gamma(\mathbf{x}))$ . Due to the loss of the local coordinate property, which is crucial in the smooth constructions of Sibson, Farin, and Hiyoshi, the resulting interpolant is only  $C^0$  at the vertices being natural neighbors of ghost points. Along a ray pointing

away from the data sites, the interpolant asymptotically converges towards a constant function. However, there might be situations where this behavior is sufficient.

Because no values are required at the data points, we have considerable freedom in their placement, our only restriction being invariance under rigid transformations. We have not yet exploited this particular possibility and provide results for the same placement strategies used in assigned ghost points.

The second option, called “assigned ghost points,” lies in generating feasible data at ghost points by extrapolating the Taylor polynomials from the data sites that were used in the construction of the ghost point. This required link between data sites and ghost points imposes some constraints on the construction. The concrete implementation of assigned ghost points depends on the placement strategy; we use two straight-forward ways to propagate the data including derivatives from convex hull vertices to the ghost points.

In particular, if a ghost point  $\mathbf{x}_i^G$  is constructed from the convex hull vertex  $\mathbf{x}_i^\ell$  with the associated Taylor polynomial  $z_i$ , and  $\mathbf{a} = \mathbf{x}_i^G - \mathbf{x}_i^\ell$  is the relative position of  $\mathbf{x}_i^G$  with respect to  $\mathbf{x}_i^\ell$ , then we assign the Taylor expansion  $z_i(\mathbf{a} + \mathbf{x})$  to  $\mathbf{x}_i^G$ . If a ghost point  $\mathbf{x}_i^{Gm}$  is constructed from the convex hull edge  $\overline{\mathbf{x}_i^\ell \mathbf{x}_{i+1}^\ell}$  we proceed as follows. For  $\mathbf{a}_1 = \mathbf{x}_i^{Gm} - \mathbf{x}_i^\ell$  and  $\mathbf{a}_2 = \mathbf{x}_i^{Gm} - \mathbf{x}_{i+1}^\ell$ , the Taylor polynomial at  $\mathbf{x}_i^{Gm}$  is taken to be the average of the Taylor expansions  $z_i(\mathbf{a}_1 + \mathbf{x})$  and  $z_{i+1}(\mathbf{a}_2 + \mathbf{x})$ . For example, if  $z_i(\mathbf{x}) = z_i + \nabla_i \mathbf{x} + \mathbf{x}^T \mathcal{H}_i \mathbf{x} / 2$ , with  $z_i$ ,  $\nabla_i$ , and  $\mathcal{H}_i$  denoting value, gradient, and Hessian, then the Taylor polynomial at the ghost point  $\mathbf{x}_i^G$  is given by

$$\begin{aligned} z_i^G(\mathbf{x}) = z_i(\mathbf{a} + \mathbf{x}) &= \underbrace{z_i(\mathbf{a})}_{=: z_i^G} + \underbrace{(\nabla_i + \mathcal{H}_i \mathbf{a}) \cdot \mathbf{x}}_{=: \nabla_i^G} + \mathbf{x}^T \underbrace{\mathcal{H}_i}_{=: \mathcal{H}_i^G} \cdot \mathbf{x} / 2. \end{aligned}$$

### 8.7.3 Static Ghost Point Placement

A static method places ghost points at fixed positions and allows a limited expansion of the domain. If the required amount of extrapolation is small, this method delivers satisfying results and is computationally less expensive than a dynamic approach. There are fewer restrictions on the placement of ghost points in the static case, and the dynamic ghost point methods discussed in Section 8.7.4 become static if the distance of the evaluation position to the convex hull is always assumed zero.

#### 8.7.3.1 Isosceles Triangles Above Convex Hull Edges

The first static approach is an ad-hoc approach that performs moderately well and allows for a small extension of the convex hull. For every convex hull edge, we offset its midpoint along its normal by a distance equal to the edge length, as illustrated in Figure 8.13.

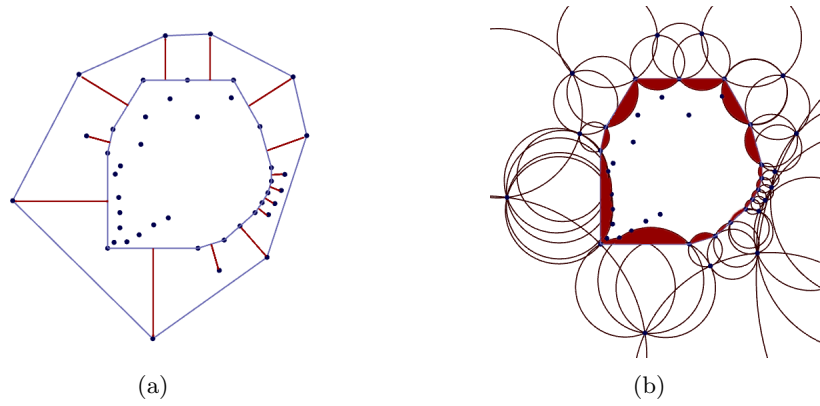


Figure 8.13: (a) Ghost point construction by offsetting the midpoint of each convex hull edge along its normal, the old and the new convex hull are drawn. (b) Blending regions in which the ghost points “pick up” the natural neighbor interpolant and provide a transition to the outside.

Values can be assigned to the ghost points according to the approach presented in Section 8.7.2 by taking the average of the extrapolated Taylor polynomials at the vertices of the edge supporting the ghost point.

### 8.7.3.2 Densely Sampled Enclosing Circle

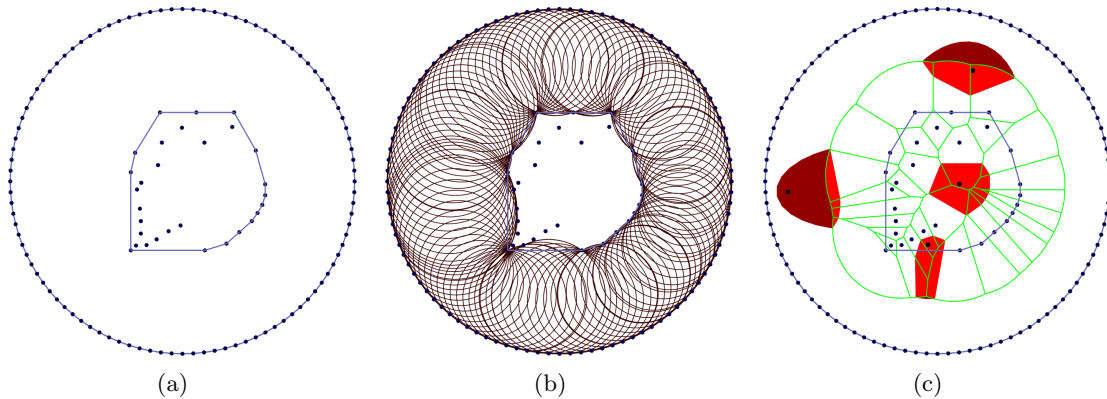


Figure 8.14: (a) Data set and the enclosing ghost points for a circle with diameter two times that of the smallest enclosing circle. (b) Circumcircles induced by the ghost points, notice that for dense point samples, there is almost no transition area. (c) Bounded Voronoi tiles of the original sites, some query positions and their associated virtual tiles. The darker color depicts the area of the virtual tile covering the Voronoi tile of a ghost point. This will be ignored when the dismissed ghost point approach is used.

The second construction determines the smallest enclosing circle of the data set, and places ghost points on a circle with the same center and radius scaled by a factor  $\beta$ . A possible extension of this idea is to use the smallest enclosing ellipse.

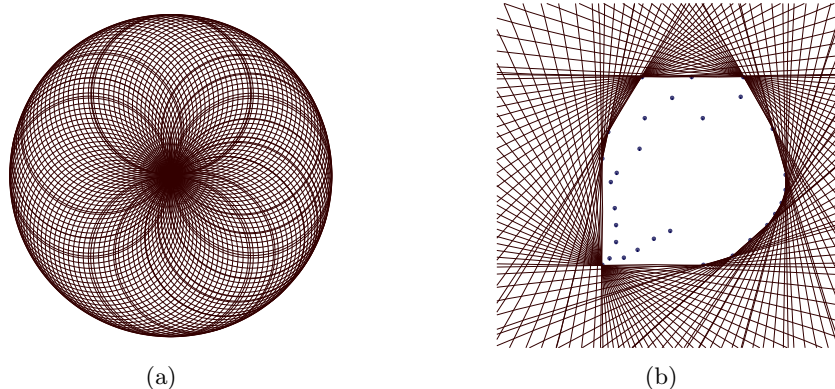


Figure 8.15: (a) Circumcircles induced by ghost points placed on a circle with radius a hundred times that of the smallest enclosing circle. (b) Close-up of the original data. Notice how the circumcircles align with the convex hull edges.

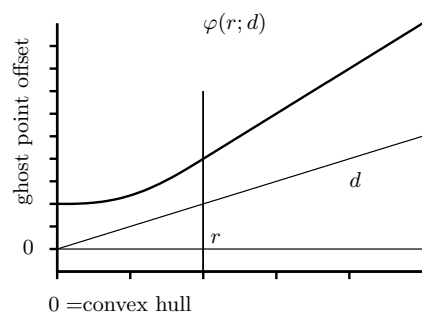


Figure 8.16: The graph of  $\varphi(r; d)$ .

An example for  $\beta = 2.0$  is shown in Figure 8.14. It is apparent that for small scale structures on the convex hull, the blending overlap of the circumcircles is less prominent. This becomes even clearer when looking at Figure 8.15, where ghost points are placed on a circle with  $\beta = 100.0$ . As  $\beta \rightarrow \infty$ , we arrive at a setting similar to the unbounded circumcircles in Section 8.6.3, and the transitions provided by the circumcircles vanish, resulting in a  $C^0$  transition.

Furthermore, it must be noted that there is no obvious way to assign values to the ghost points other than the mean value of the data set. Considering the complexity concerns worded in Section 6.5, it is not feasible to use this construction for extrapolation with Hiyoshi’s interpolant as the number of neighbors is naturally extraordinarily high.

But if used with the dismissed ghost point strategy, this construction delivers reasonable results as we see in the comparisons in Section 8.8.

### 8.7.4 Dynamic Ghost Point Placement

A dynamic ghost point method constructs an adequately expanded convex hull for every evaluation point  $\mathbf{p} \in \mathbb{R}^n$ , with the ghost points’ coordinates continuously depending on that of  $\mathbf{p}$ . Let  $d := d(\mathbf{p}, \mathcal{C}(\mathbf{X})) = \min_{\mathbf{x} \in \mathcal{C}(\mathbf{X})} \|\mathbf{x} - \mathbf{p}\|$  be the distance of  $\mathbf{p}$  to the convex hull of the data sites. We propose to place the ghost points at a distance twice as far



from the convex hull as  $\mathbf{p}$  such that it is always in the middle of some enclosing points. It is important that ghost points are distinct, since otherwise the family of interpolants parameterized by  $d$  is no longer continuous with respect to the evaluation position. We model the link between  $d$  and the coordinates of the ghost points using a monotone, smooth function  $\varphi(d)$  such that  $\varphi(d) \geq 2d$ . In our examples, we choose the piecewise quartic  $C^2$  function  $\varphi$  that blends between an initial, constant distance  $r$  and  $2d$  over the interval  $[0, r]$  as shown in Figure 8.16. The function is given by

$$\varphi(r; d) = \begin{cases} r + 2d^3/r^2 - d^4/r^3 & \text{if } d < r \\ 2d & \text{else.} \end{cases}$$

An extension of  $\varphi$  to higher smoothness is straightforward, and any monotone, smooth function whose slope converges to roughly two is appropriate. The parameter  $r$  depends on the individual setting and will be discussed along with the proposed ghost point approaches.

**Dynamic Convex Hull Offset (CHO)** We generate ghost points  $\mathbf{x}_i^G$  by displacing the convex hull vertices  $\mathbf{x}_i^\ell$  such that the new convex hull edges are parallel to the old convex hull edges at a distance of  $\varphi(r; d)$ , as shown in Figure 8.17(a), and call this the convex hull offset (CHO) strategy. To make the ghost point distribution more homogeneous, we insert additional ghost points  $\mathbf{x}_i^{Gm}$  on the new convex hull such that they project onto the mid-points  $(\mathbf{x}_i^\ell + \mathbf{x}_{i+1}^\ell)/2$  of the old convex hull edges, where index arithmetic is modulo the number of convex hull vertices. To ensure that the evaluation position  $\mathbf{p}$  is contained within the convex hull of the ghost points,  $\varphi(r; d) \geq d$  must hold. Since  $\varphi(r; d) \geq 2d$ , it is furthermore guaranteed that  $\mathbf{p}$  lies well away from the boundary of the convex hull. The advantage of this particular construction is the association between ghost points and data sites, which allows a meaningful assignment of values and derivatives to the ghost points according to Section 8.7.2. The effect of this assignment is shown in Figure 8.18 for the extrapolation of linear Taylor polynomials. When applying this approach to higher dimensions, we suggest to use a ghost point for every  $1, 2, \dots, n-1$ -simplex on the convex hull, i.e., for vertices, edges, and triangles in  $3D$ .

The two major smooth natural neighbor interpolants available, Farin's  $C^1$  and Hiyoshi's  $C^2$  interpolant, have their reproduction power degraded by one as soon as their evaluation involves ghost points, as we explain in the following. Both interpolants internally use Bézier simplices to model the interpolation constraints given by the derivative data. Thanks to the concept of degree elevation by which they determine underconstrained control points, Farin's interpolant has second order precision for first degree Taylor polynomials at the data points, and Hiyoshi's interpolant has third order precision for second degree Taylor polynomials. As described in the paragraph on "Reproduction Power" on page 149, the data at ghost points are extrapolated from the Taylor polynomials at the convex hull data points used in their construction. Consequently, the data at the ghost points does in general not agree with the function reproduced by the interpolant in the interior and the reproduction power of the extrapolated interpolant is degraded by one wherever a ghost point is involved in the evaluation.

A drawback of the CHO strategy results from offsetting all edges by the same amount.

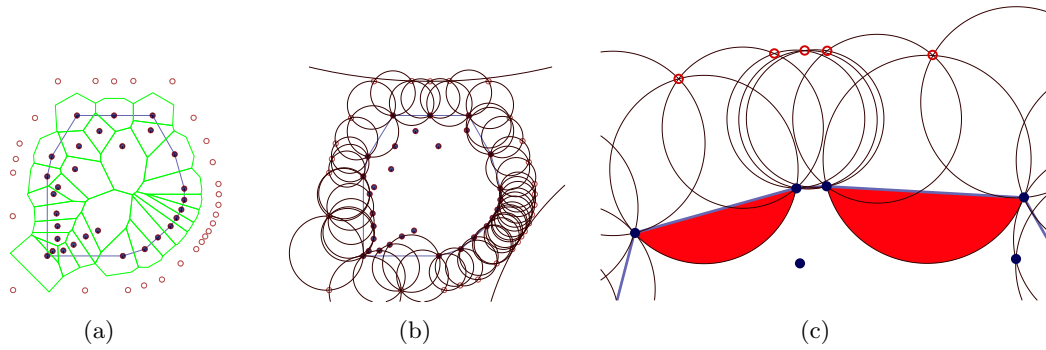


Figure 8.17: Ghost points (hollow) in the dynamic convex hull offset approach. (a) The initial setting for  $d = 0$ . (b) The corresponding circumcircles, providing the blending regions. (c) Convex hull displayed as solid lines with interior below. Only circumcircles involving ghost points are displayed. The shaded blending regions indicate where the original natural neighbor interpolant is augmented to alleviate the convex hull artifacts, the region being much less pronounced at the short edge in the middle.

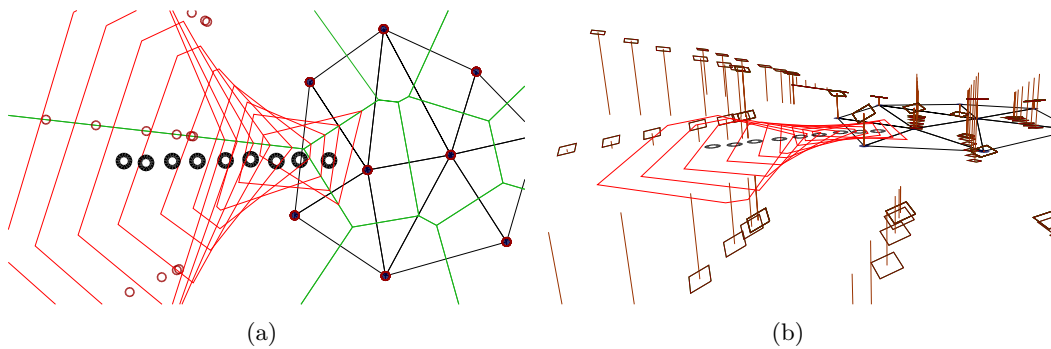


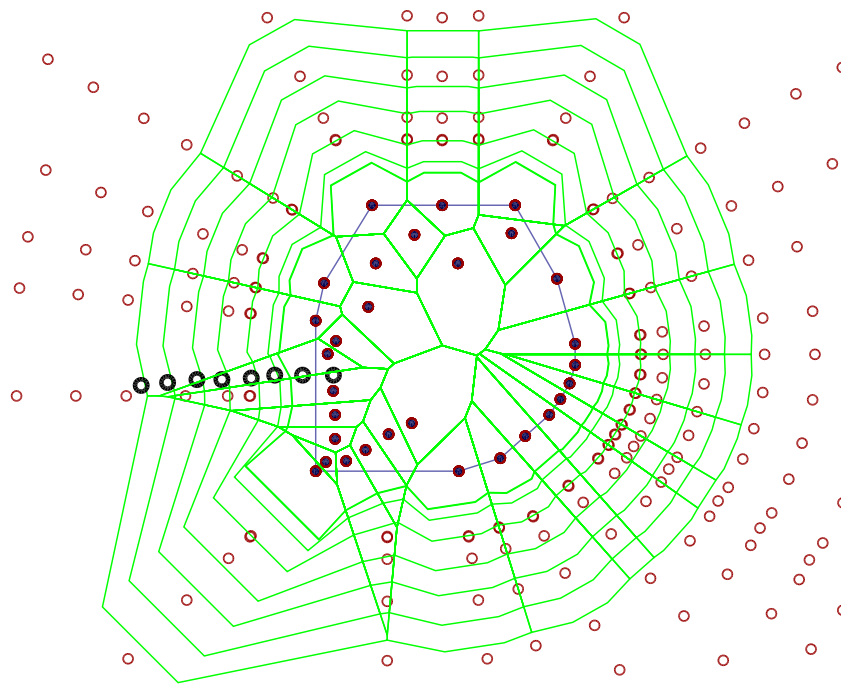
Figure 8.18: (a) The virtual tile evolution for evaluation positions (drawn as thick rings) moving away from the data sites, with ghost points (small rings) moving away as well. (b) Perspective 2.5D view of the setting from (a), the gradients shown as tilted rectangles placed at the data sites.

Dense data requires a small, sparse data a large offset, as shown in Figure 8.17(c). The more the density of data sites varies along the convex hull, the bigger a compromise the choice of the global offset becomes.

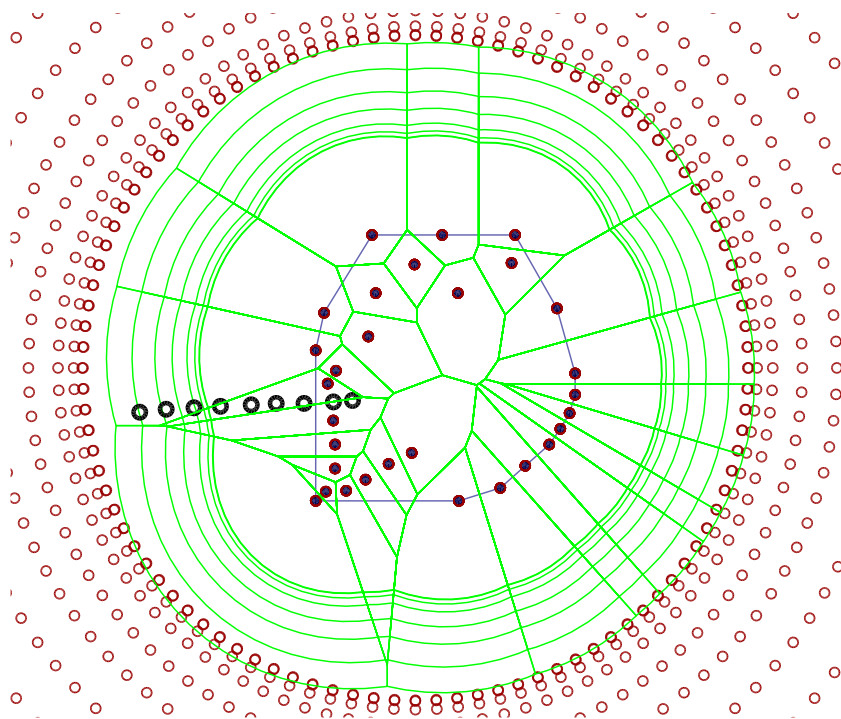
Because ghost points change their position based on the evaluation position, the Delaunay triangulation to facilitate access to the Voronoi diagram suffers frequent updates as well. An evaluation of the interpolant over a grid would benefit from a dynamic update strategy that minimizes the amount of updates, which will be subject of further research. For random access to the interpolant we suggest to construct for every evaluation a small, local Delaunay triangulation from which the classical natural neighbor interpolant can be evaluated. First, all vertices of old simplices in conflict with the evaluation position must be determined. Then, all ghost points depending on any of these vertices must be inserted into the local Delaunay triangulation, which usually accesses convex hull vertices additional to the ones already involved.

**Dynamic Dense Circle Placement (DDC)** In this strategy, we opt to determine the smallest enclosing circle of the data set, double its radius and place evenly distributed ghost points on it, as shown in Figure 8.19(b). The resulting static ghost point method is similar to the “windowing” approach of Owen in [Owe93], where the ghost points play the role of the window boundary. In the dynamic ghost point setting, we choose  $r$  as the radius of the smallest enclosing circle such that the ghost points are smoothly offset starting with a circle of radius  $2r$ .

Because there is no useful association between ghost points and data sites, we can either use the dismissed ghost point approach or assign the mean value of the data at the data sites. Neither of these choices is optimal. The first extrapolates a constant values representing a weighted average of the data at visible convex hull data sites along rays away from the convex hull, but is only  $C^0$ . The second provides a globally smooth interpolant, but provides only a uniform, constant asymptotic behavior outside.



(a)



(b)

Figure 8.19: (a) Development of the Voronoi diagram of  $\mathbf{X}$  (solid dots) and  $\mathbf{X}^G$  (thin, hollow dots) for a sequence of query positions (thick, hollow dots) ranging from inside to outside the convex hull of  $\mathbf{X}$ . (b) Same visualization for the dense circle approach.

## 8.8 Visual Comparison of Extrapolation Methods

In this section we display some selected, representative height field visualizations of several approaches discussed so far. The selection of data sets is such that the main issues are shown.

### 8.8.1 Tame Data Set

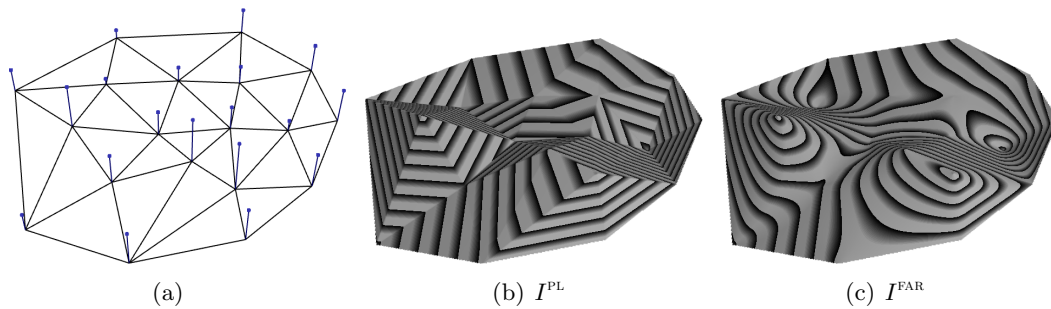


Figure 8.20: (a) A very well behaved data set without any challenging spatial constellations. (b) Piecewise linear interpolation. (c) Farin's  $C^1$  interpolant.

The tame data set, shown in Figure 8.20, contains no challenging configurations and is well handled by all considered extrapolation methods. It allows to get an idea of the function shape produced by the individual extrapolation schemes apart from critical geometric constellations, shown in Figure 8.21.

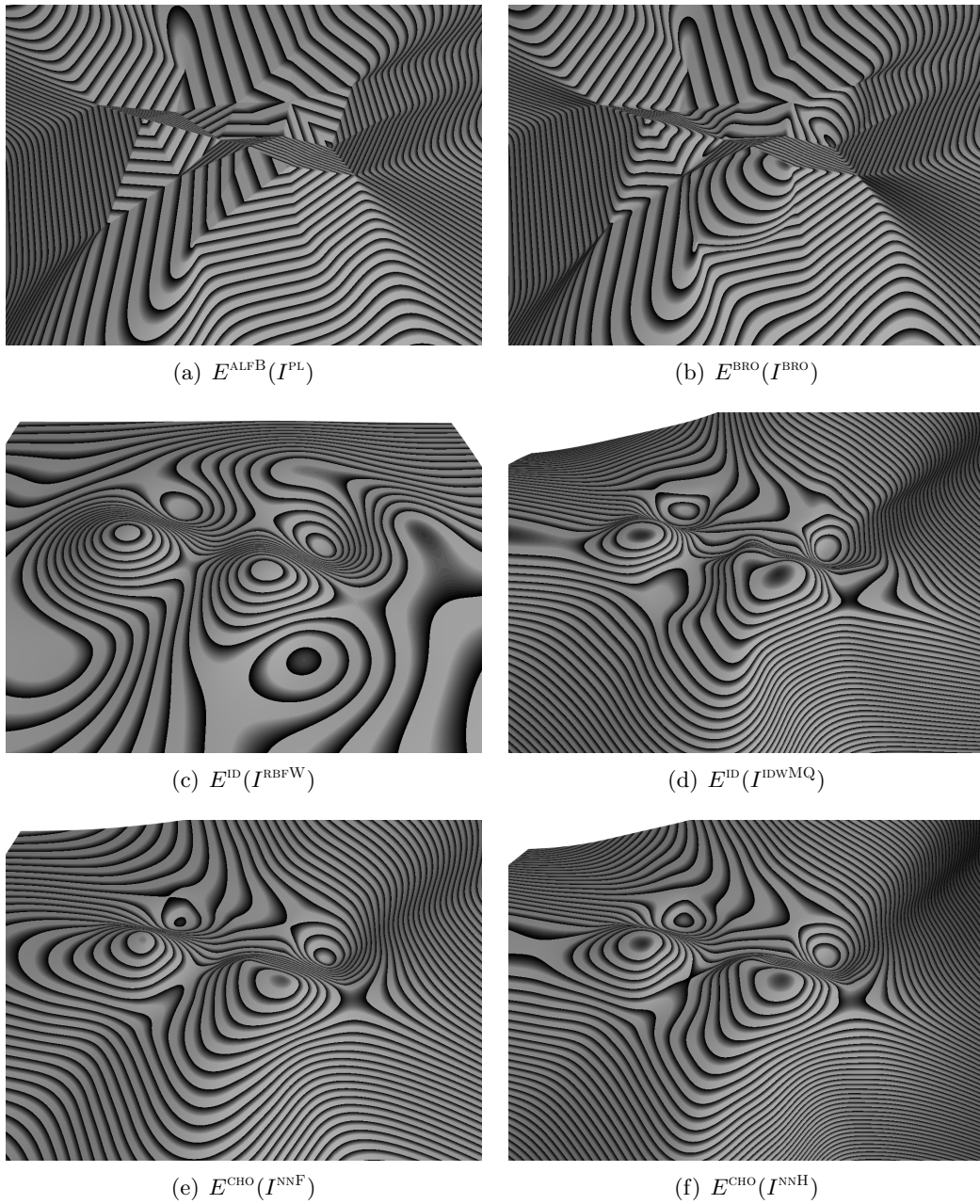


Figure 8.21: The displayed extrapolation schemes all cope well with the tame data set. Gradients have been provided at the boundary vertices.

## 8.8.2 Oscillation Data Set

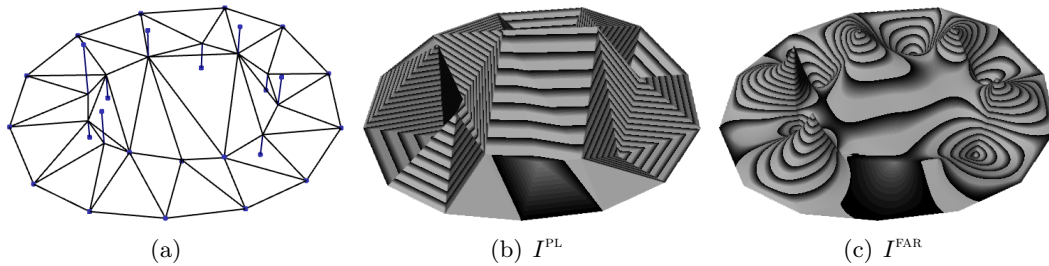


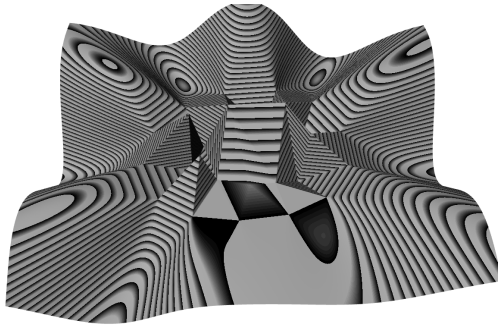
Figure 8.22: (a) A data set in which the data sites next to the boundary take alternating values along the boundary. (b) Piecewise linear interpolation. (c) Farin's  $C^1$  interpolant.

In this data set, shown in Figure 8.22, the values at the data sites next to the boundary alternate along the boundary, which also results in alternating gradients of the interpolant along the boundary for local interpolation schemes. This data set illustrates changes between locally different function shapes.

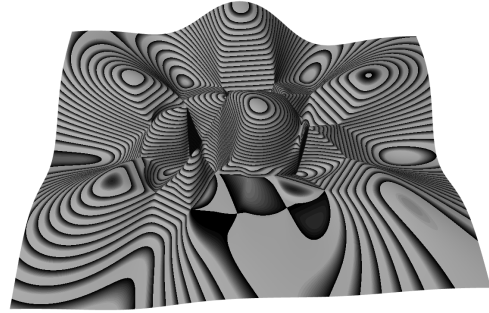
Alfeld's extrapolation picks up the gradient of the piecewise linear interpolant inside and blends them outside, resulting in expected behavior in Figure 8.23(a). Brown's extrapolation also produces a pleasant-looking transition between outside and inside, yet suffers moderate overshoots from Brown's interpolation scheme inside in Figure 8.23(b).

The radial basis function approach produces significant overshoots, while providing a very pleasant transition to the outside in Figure 8.23(c). With inverse distance weighting, the interpolant locally adapts well to the data, while showing some slight wiggles outside in Figure 8.23(d). Although the interpolant has only finite extent, this is not visible in the image due to the limited evaluation domain for the height field.

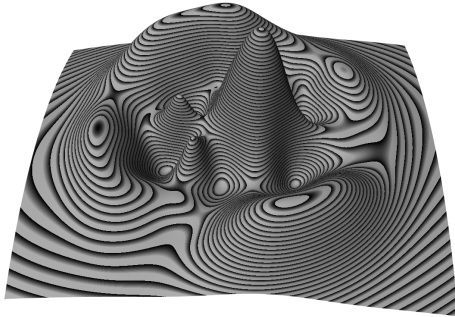
Since in this data set, both values and gradients at the convex hull vertices are zero, the corresponding ghost point extrapolation simply produces local transitions to the constant zero function outside, unaffected by the oscillations inside in Figure 8.23(e) and (f).



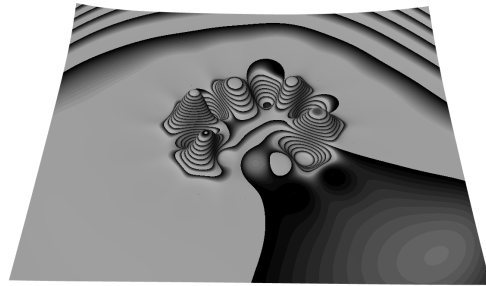
(a)  $E^{\text{ALFB}}(I^{\text{PL}})$



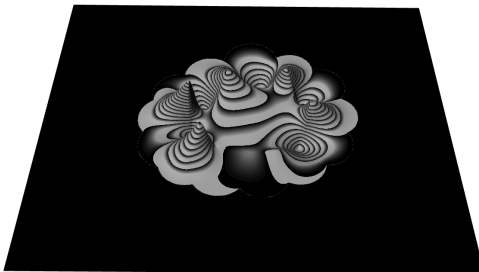
(b)  $E^{\text{BRO}}(I^{\text{BRO}})$



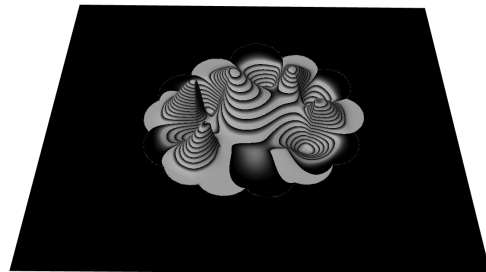
(c)  $E^{\text{ID}}(I^{\text{RBFW}})$



(d)  $E^{\text{ID}}(I^{\text{IDWMQ}})$



(e)  $E^{\text{CHO}}(I^{\text{NNF}})$



(f)  $E^{\text{CHO}}(I^{\text{NNH}})$

Figure 8.23: Extrapolation schemes applied to the oscillating data set. The values and derivative data for convex hull vertices are zero. (a), (b) Local extrapolation based on tessellations. (c), (d) Global interpolants. (e), (f) Natural neighbor extrapolation using ghost points.



## 8.8.3 Flip Data Set

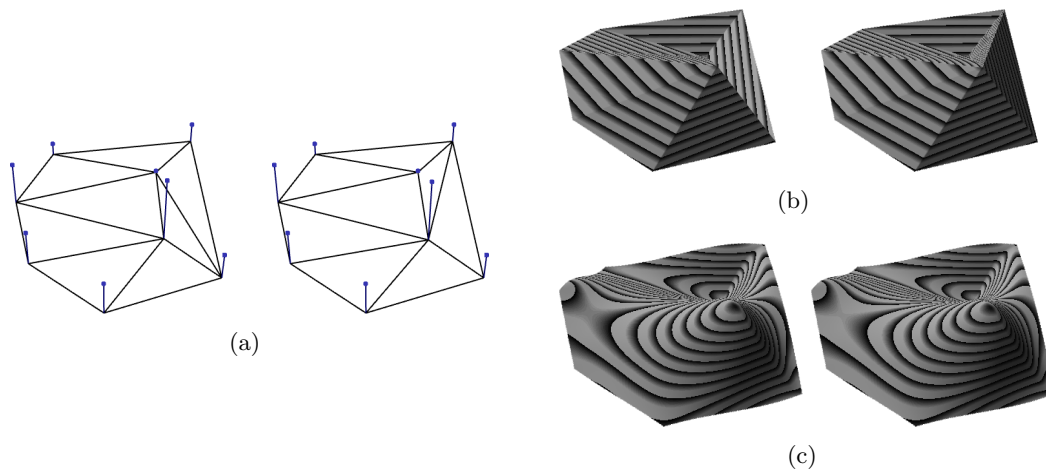
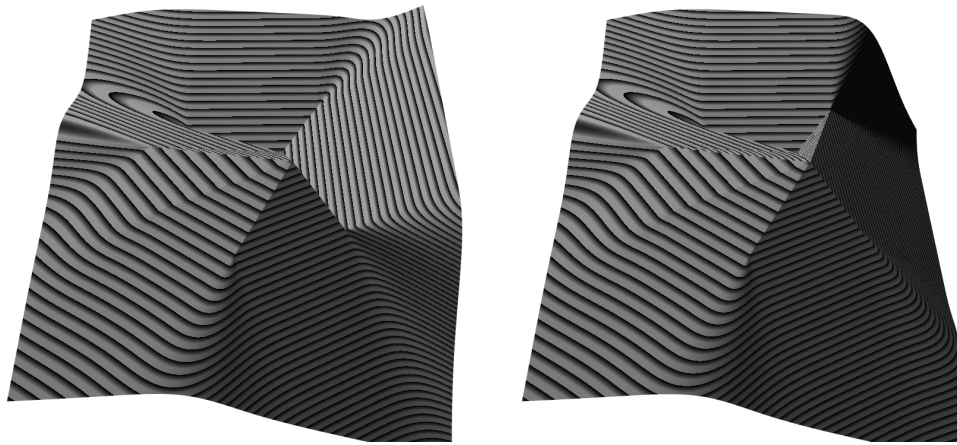
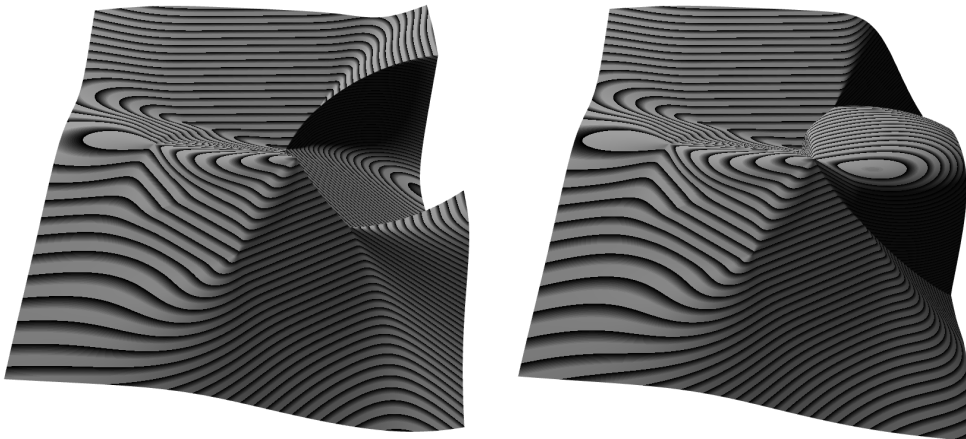


Figure 8.24: (a) Two almost identical data sets, with different tessellations. (b) Linear interpolation in the underlying triangulation, the discontinuous change of the interpolant due to the edge flip is clearly visible. (c) Farin's  $C^1$  interpolant does not have this problem.

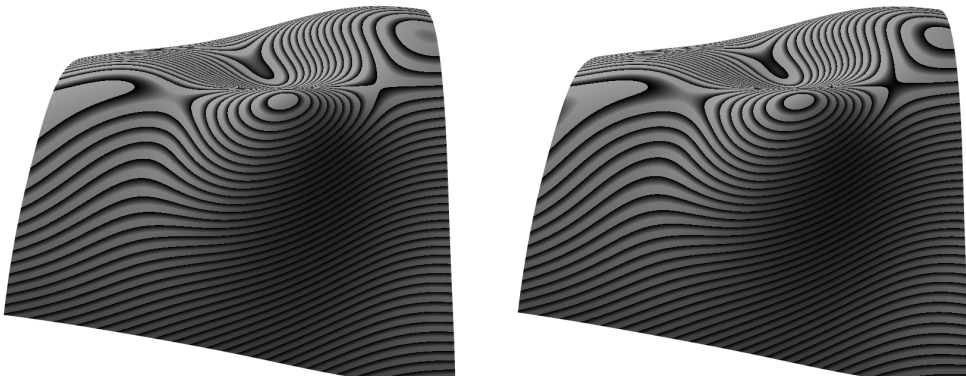
The flip data set, shown in Figure 8.24(a), illustrates the “similar results from similar input” aspect, which is apparently problematic for tessellation-based approaches. Figure 8.24(b) and (c) show interpolants with restriction to the convex hull of the data. The nearly identical data sets were deliberately chosen such that a Delaunay edge flip occurs among them. On the following pages we show representative extrapolation methods applied to the different data sets next to each other in each line. The choice of the Delaunay triangulation is no real restriction since for every tessellation, there are situations at which topological changes occur. We show the extrapolation of tessellation-based interpolants in Figure 8.25(a) and (b), where the difference between left and right images are clearly visible. The global interpolants shown in in Figure 8.25(c) and Figure 8.26(a) as well as the natural neighbor interpolants with assigned ghost points using the CHO strategy shown in Figure 8.26(b) and (c) are not affected by the change in the triangulation.



(a)  $E^{\text{ALFB}}(I^{\text{PL}})$

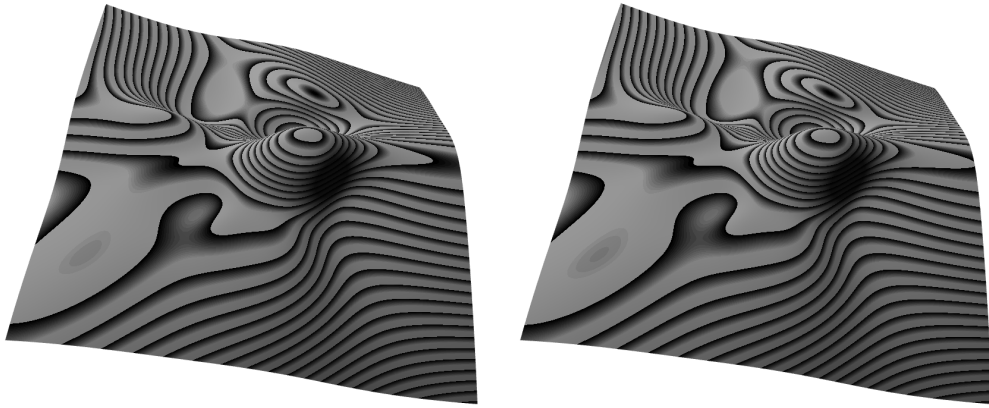


(b)  $E^{\text{BRO}}(I^{\text{BRO}})$

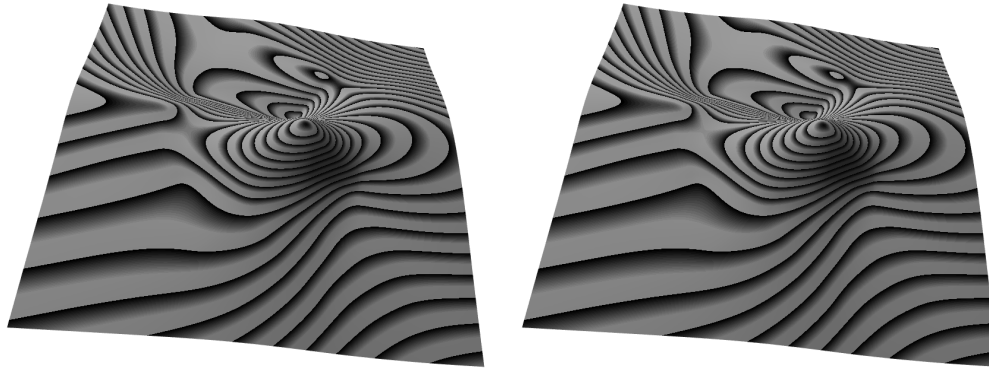


(c)  $E^{\text{ID}}(I^{\text{RBFW}})$

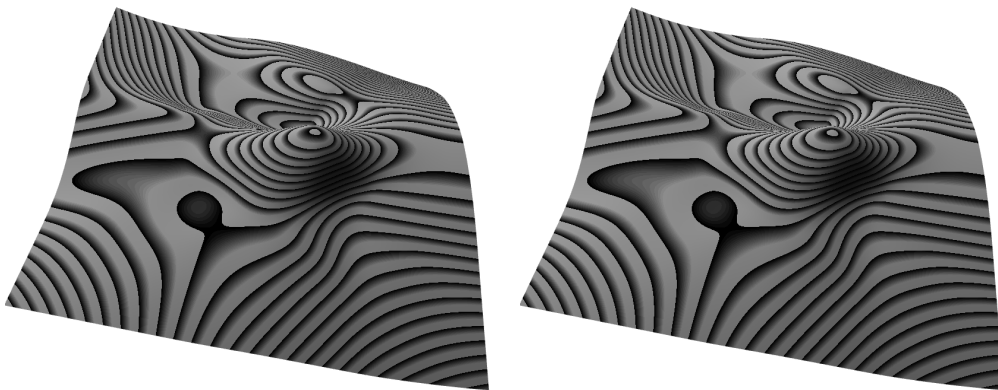
Figure 8.25: (a), (b) The discontinuous change of the interpolant in  $D$  is propagated to  $D^E$ . (c) Wendland's RBF interpolant is independent of any triangulation.



(a)  $E^{\text{ID}}(I^{\text{IDW MQ}})$



(b)  $E^{\text{CHO}}(I^{\text{NNF}})$



(c)  $E^{\text{CHO}}(I^{\text{NNH}})$

Figure 8.26: (a) Like the RBF interpolant in Figure 8.25(c), Nielson's modified quadratic Shepard interpolant is independent of any triangulation. The weight functions extend far enough to cover the displayed domain. (b), (c) Natural neighbor interpolants extrapolated using the CHO ghost point strategy have no issues either.

## 8.8.4 Sliver Data Set

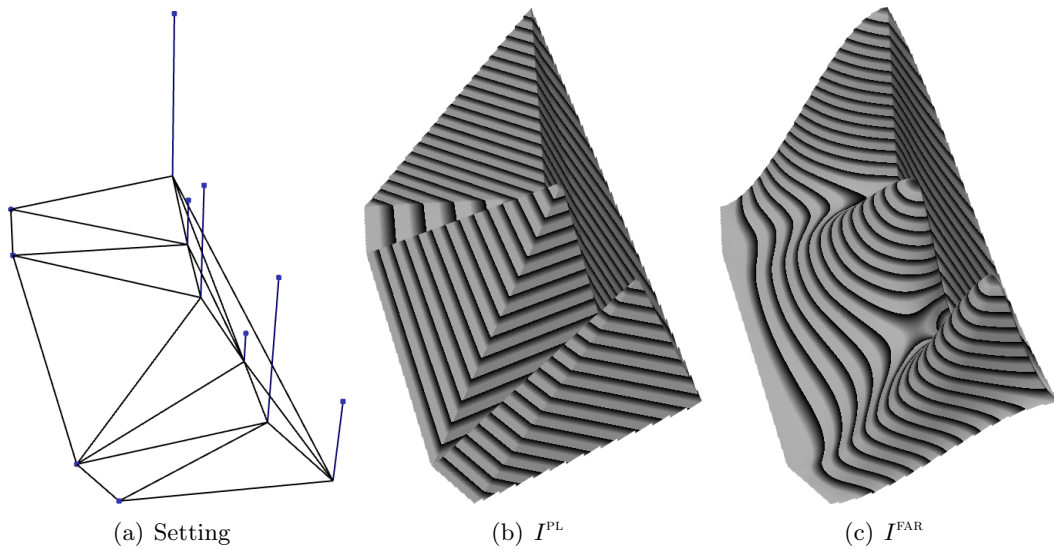


Figure 8.27: Benchmark data set for convex hull artifacts. (a) 2D data set with values indicated by vertical lines. The data distribution is slightly concave near the boundary of the convex hull. (b) Linear interpolation in the underlying triangulation. The observable artifacts are apparent in a similar fashion in any tessellation-based interpolant. (c) Farin's  $C^1$  interpolant. Although natural neighbor based interpolation is generally independent of a particular triangulation, it suffers similar artifacts on the boundary of the convex hull, where the interpolant is solely determined by the data at adjacent convex hull sites.

The sliver data, shown in Figure 8.27, provides a setting in which the convex hull artifacts of tessellation-based and natural neighbor interpolants become apparent. The images show how the tessellation based extrapolation methods simply continue the degenerate local shape, while ghost point methods as well as global interpolants succeed in overcoming the artifacts.

Figure 8.28 shows a range of extrapolation methods applied to the sliver data set. The important difference to notice between convex hull extension schemes in Figure 8.28(a) and (b) and the remaining ones in (c) through (d) is the influence that data near the convex hull has on the shape of the interpolant outside the convex hull. The upper row shows an inadequate dependency, while the lower two rows show intuitively feasible behavior.

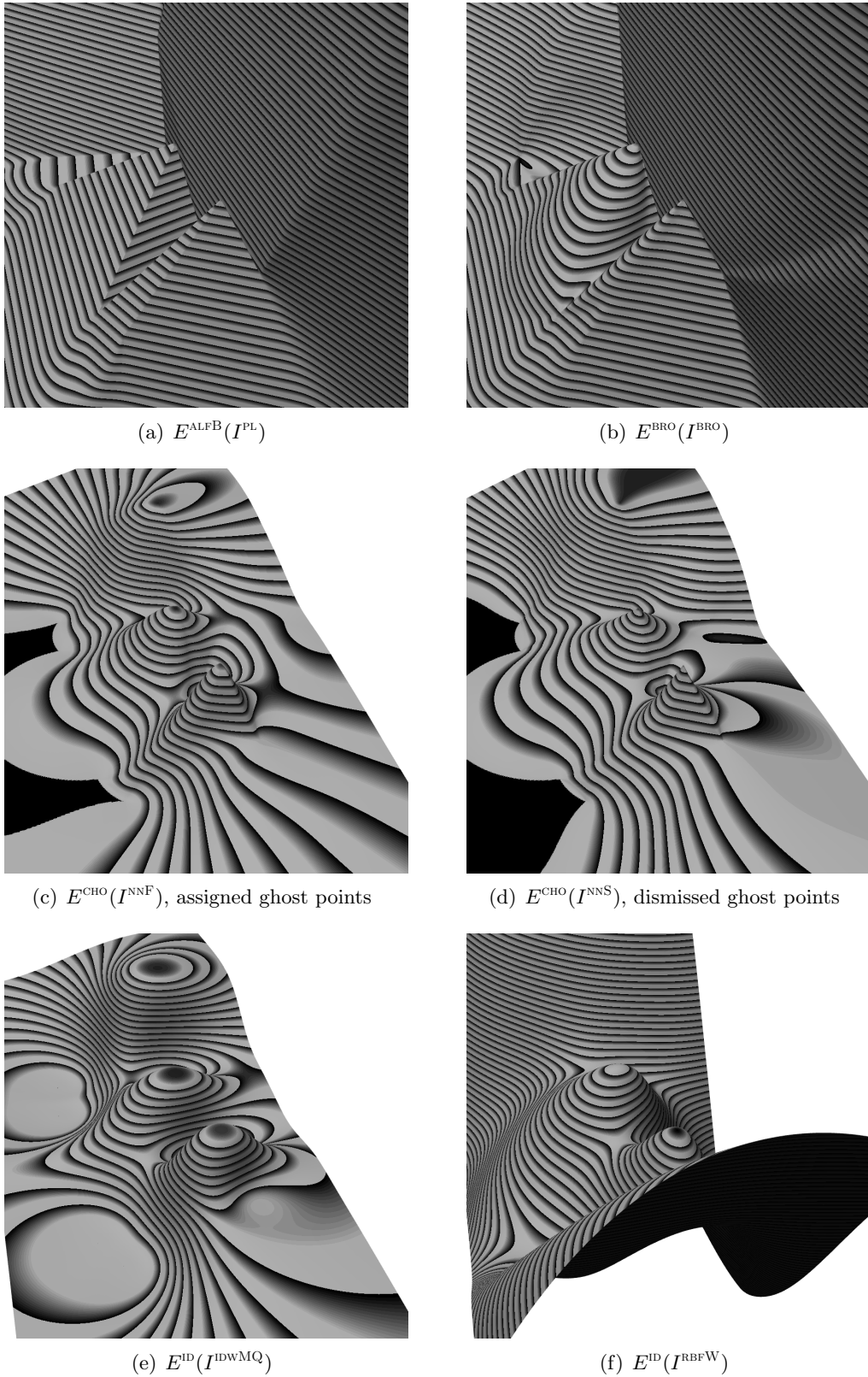


Figure 8.28: (a), (b) The local extrapolation schemes smoothly continue the interpolants where their original domain ends. (c), (d) Ghost point methods remove the artifacts. (e), (f) Global methods, being independent of the tessellation, show no artifacts either.

### 8.8.5 Artifacts and Asymptotic Behavior

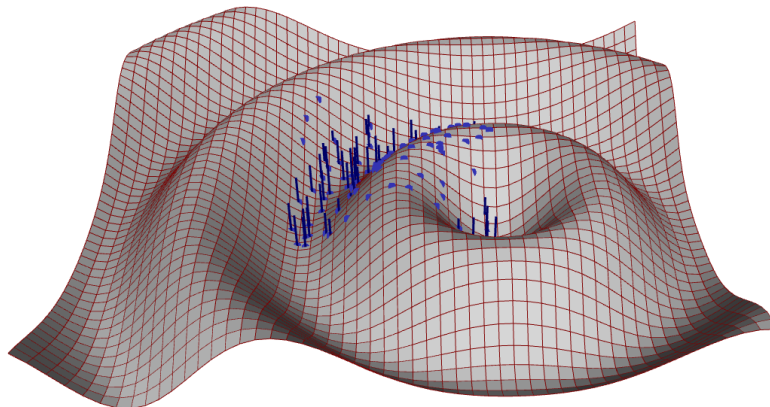


Figure 8.29: The data set including second degree Taylor polynomials of  $\cos \|\mathbf{x}\|$  have been sampled at the indicated positions.

The following illustrations are based on second degree Taylor polynomials sampled from  $\cos \|\mathbf{x}\|$  over a small region as shown in Figure 8.29. The images display the extrapolated height field above the error field, the latter showing the difference between the interpolant and the cosine function, where a plane region indicates low error. Because of the interpolation property, the absence of wriggles proves high absolute accuracy of the interpolant.

The first aspect addressed by this example is the artifact removal performed by the ghost point extension near the convex hull, which is shown in Figure 8.30 and 8.31. Given the non-polynomial nature of the cosine function, all considered interpolants must diverge from the function in the exterior, and we are interested in how gracefully they start to deviate. We note that Figure 8.30(a) and (b) display artifacts near the convex hull, while interpolating nicely inside. Figure 8.30(c) and (d) display the dynamic CHO strategy applied to do  $C^1$  respective  $C^2$  extrapolation. The important detail here is how the interpolant starts to deviate from the function at the boundary of the flat region. Another interesting detail not directly related to extrapolation is how much better the interpolation with  $I^{\text{NNF}}$  looks compared to  $I^{\text{NNH}}$ . We further show results of the DDC strategy in Figure 8.31(a), as well as the result of applying Wendland's RBF interpolant in Figure 8.31(b). RBF generally possess good approximation quality for smooth functions, so it comes at no surprise that these give the visually most pleasing results. The results of inverse distance weighted methods in Figure 8.31(c) and (d) show that extrapolation in case of  $I^{\text{IDWQ}}$  delivers a smooth function outside, yet the interpolated function in the interior is of very low quality.

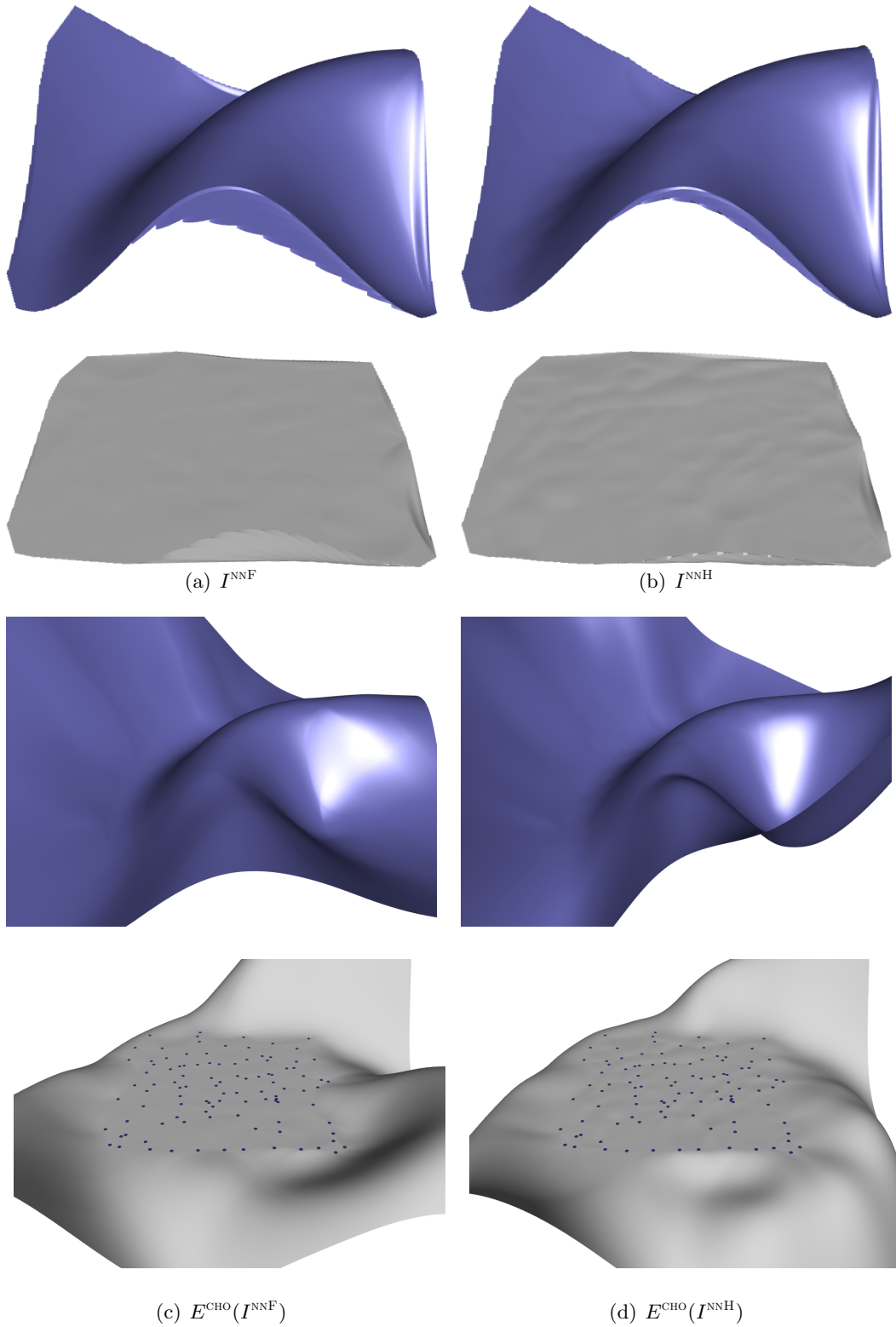


Figure 8.30: (a), (b) Farin's and Hiyoshi's interpolants applied to the data set. The artifacts in the front and on the right side are clearly visible. (c), (d) Both interpolants augmented by the CHO ghost point strategy. An interesting result is the apparently worse performance of the  $C^2$  interpolant in terms of wriggles.



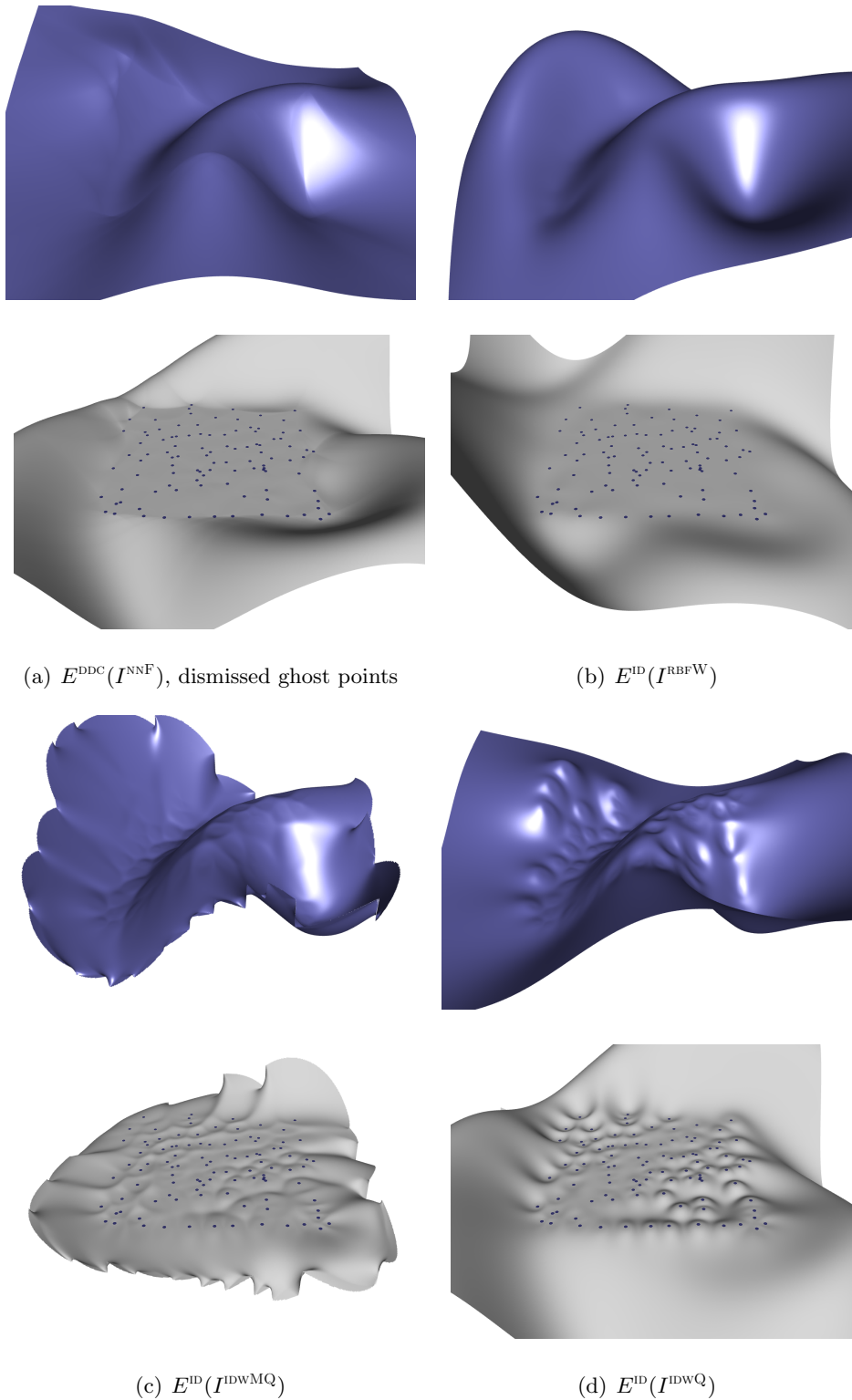


Figure 8.31: (a) The DDC ghost point approach exhibits a rather abrupt change between inside and outside. (b) RBFs perform better than ghost point-augmented natural neighbor interpolants in (a) and Figure 8.30 both in terms of transition between inside and outside, and approximation quality. (c) Nielson's modified quadratic Shepard's interpolant is neither suited for interpolation nor extrapolation. (d) Away from the convex hull, Shepard's interpolant provides a smooth function, yet the approximation quality in the interior is not satisfactory.



The second considered aspect, shown in Figures 8.32 and 8.33, sets ghost point-augmented natural neighbor interpolants apart from RBFs in that they follow the last observed trend at the convex hull rather than the global characteristic of the data set. This particular property gains importance as one usually observes heterogeneous characteristics over large data sets, and extrapolation should honor the local characteristic where it is performed. In particular, compactly supported and quasi-local radial basis functions converge to the globally fitted polynomial. Unbounded RBFs show even worse behavior in that they diverge in an uncontrolled way. Natural neighbor interpolation based on dynamic, assigned ghost points, on the other hand, converge towards a mixture of Taylor polynomials at convex hull vertices that are visible from the evaluation position, and can vary for different parts of the convex hull.

Except for Figure 8.30(d), we omit results for Hiyoshi's  $C^2$  interpolant due to infeasible computation times. In each evaluation with  $n$  natural neighbors, Hiyoshi's interpolant requires the construction of an  $n$ -variate, quintic Bézier simplex from the second degree Taylor polynomials at the natural neighbors, which has complexity  $\mathcal{O}(n^5)$  with a large constant factor. Outside  $\mathcal{C}(\mathbf{X})$ , the natural neighborhood of the evaluation point easily becomes as large as 30 or more points, and the sampling of the height-field for visualization becomes infeasible.

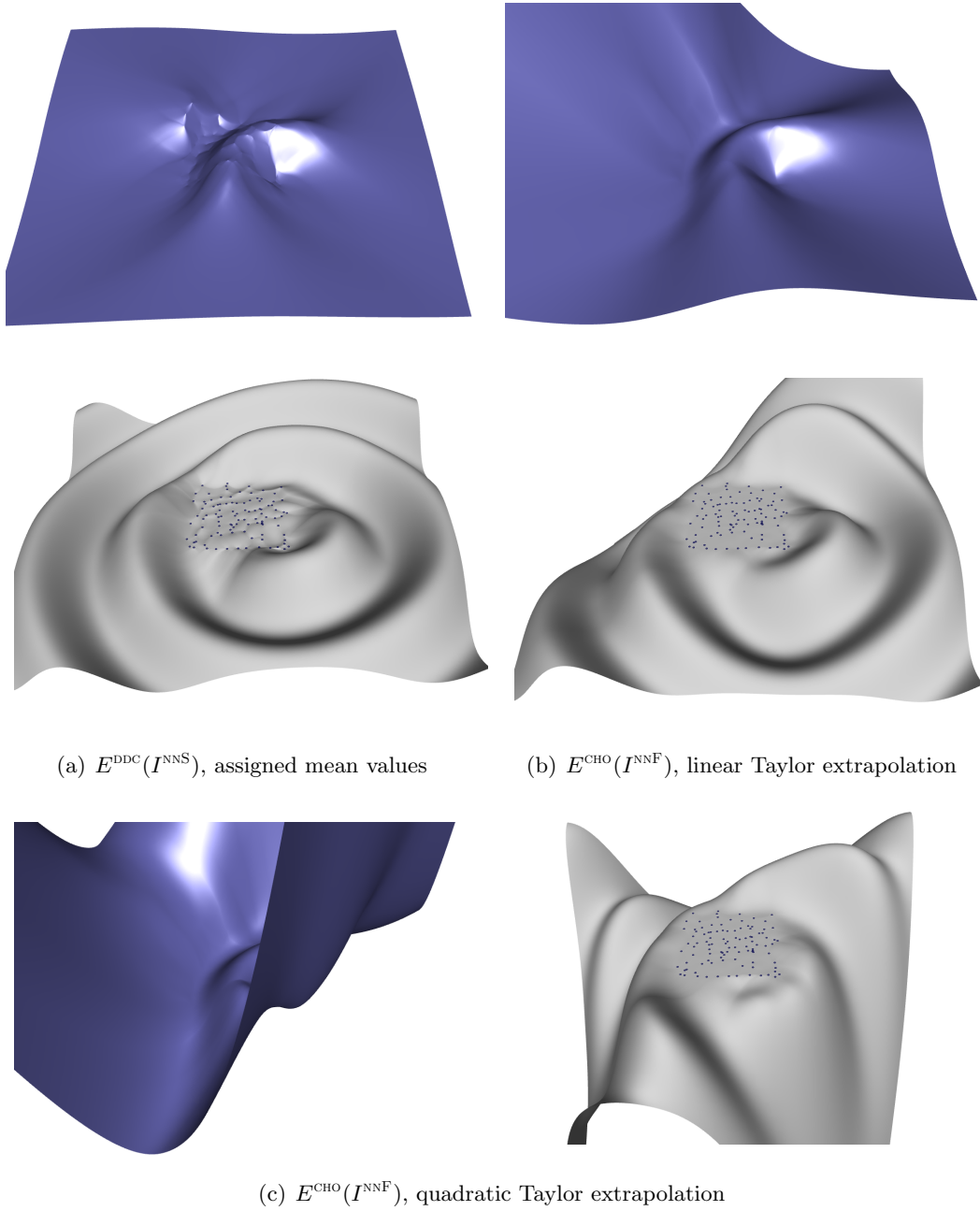


Figure 8.32: Ghost points are assigned (a) the mean value of all boundary data sites, (b) extrapolated linear Taylor polynomials, (c) extrapolated quadratic Taylor polynomials.

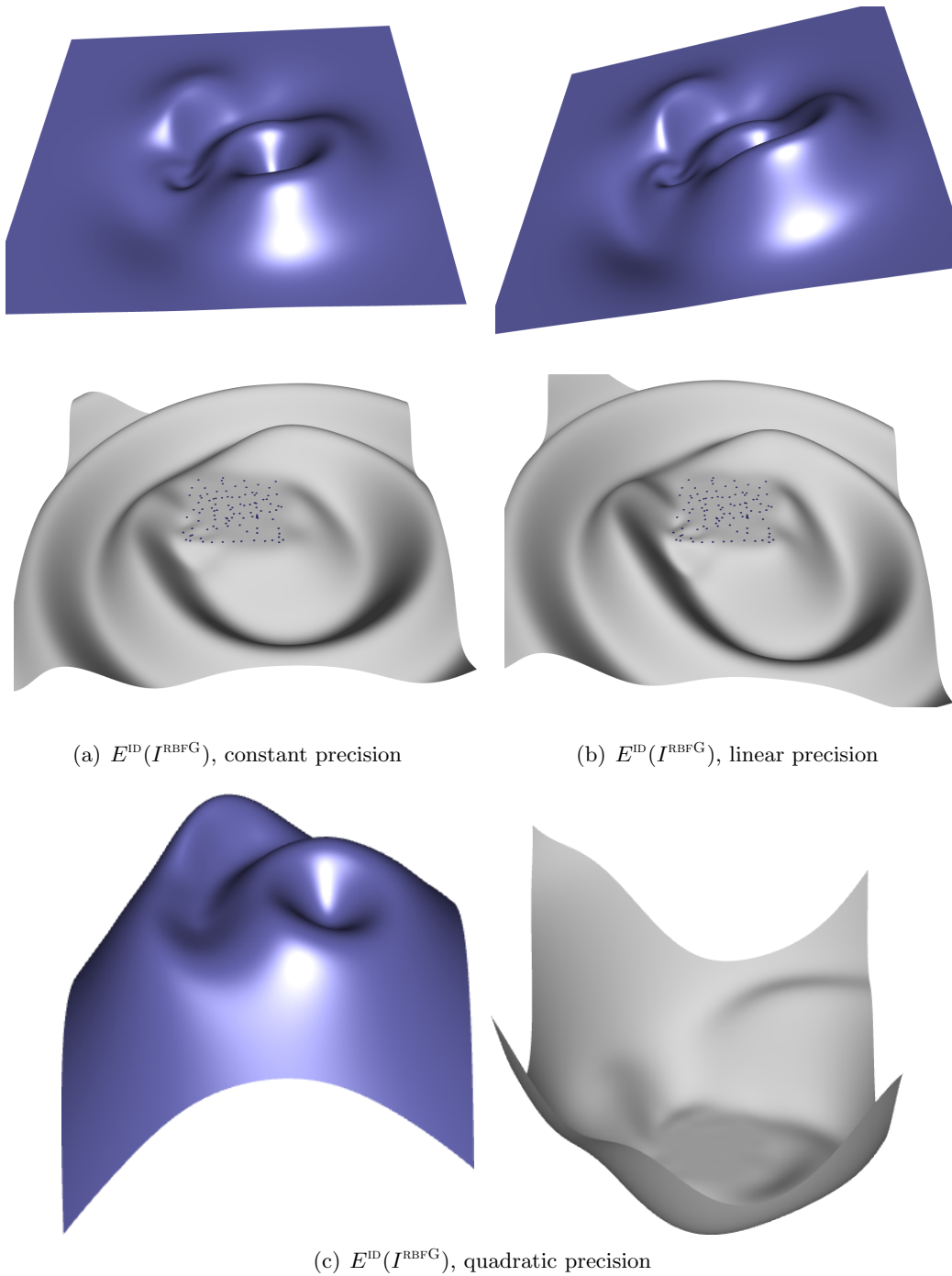


Figure 8.33: The asymptotic behavior of quasi-local radial basis functions based on underlying polynomials of (a) degree zero, (b) degree one, and (c) degree 2.

## **8.9 Conclusion and Future Work**

The main objective of this paper was the presentation of natural neighbor extrapolation based on dynamic ghost points. To provide an adequate context, we first reviewed available extrapolation approaches, which we subjected to a comparative assessment based on a characterization that applies to scattered data extrapolation in general.

Natural neighbor interpolation is by default limited to the convex hull of the data sites, and exhibits undesirable artifacts near the boundary of the convex hull. We presented a framework for the extrapolation of natural neighbor interpolants that extends their domain to all of space, while maintaining all desirable properties of the interpolant. By augmenting the original data set with dynamically constructed ghost points, we overcome the boundary artifacts inherent to natural neighbor interpolation. Furthermore, we are able to extrapolate local trends defined by the derivative data at the data sites in a local fashion, which is considered an advantage over RBF interpolants.

It is apparent that extrapolation without any further constraints is an ill-posed problem. For a set of feasible assumptions, RBFs and the proposed ghost point construction can clearly be identified as the approaches providing results of highest quality. RBF interpolants are global in nature, but possess the better approximation quality in general, while natural neighbor interpolants based on ghost points provide local scattered data interpolation that better adapts to local characteristics of the data set.

One essential requirement for scattered data interpolation that this paper pointed out was the continuous dependency of an interpolant with respect to the input data, which adds to the reliability of the interpolation scheme. Natural neighbor interpolants fulfill this requirement as long as the data sites are disjoint and the set of convex hull vertices does not change. This latter constraint poses a limitation that should be overcome, and which will be investigated in future work.

# 9 Adaptive Delaunay Tessellation

In this chapter we give a rigorous definition and prove fundamental properties of a recently introduced mesh generation technique, called Adaptive Delaunay Tessellation.

This chapter builds upon the structures introduced in Section 2.3.4 on the Delaunay triangulation, Section 2.3.5 on the Voronoi diagram, and Section 2.3.1 on non-convex polygons in 2D. It furthermore uses notions introduced Section 2.4 on graphs.

## 9.1 Background

In [CSB<sup>+</sup>08], a novel tessellation technique, called *Adaptive Delaunay Tessellation* (ADT), was introduced in the context of computational mechanics as a tool to support nodal integration schemes in the Finite Element Method. While focusing on potential applications, the former presentation lacked rigorous proofs of the claimed geometric properties of the ADT. This chapter gives pending proofs for the three main claims, uniqueness, connectedness, and coverage of the Voronoi tiles by adjacent ADT tiles.

We first introduce some terms and notions used throughout the rest of this chapter in Section 9.3. Uniqueness and connectedness of the ADT are stated and proved in Section 9.4.1 and Section 9.4.2. Section 9.4.3 introduces an alternative characterization of the ADT as an interesting result that also simplifies the proof of the last treated property in Section 9.4.4 on the coverage of the Voronoi tile of a vertex by ADT elements adjacent to it.

## 9.2 Introduction

One building block in finite element analysis is the background tessellation of a spatial domain, where requirements vary with the applications at hand. The corresponding field of mesh generation traditionally focuses on the generation of both vertex positions and connectivity, starting from a description of the domain boundary. Vertices are placed heuristically to adaptively sample local features and maintain a good shape of the resulting elements, although the discussion about the optimal shape is not settled even for triangles [She02]. The tessellation can be done in ad-hoc constructions [Loh96], iteratively [TAD07], or based on guidance fields [BH96, ADA07] to name a few. In the rare occasion of a predetermined set of vertices, the choice of methods is generally limited to the constrained Delaunay triangulation or other, data-dependent, triangulations [HD06].

In [CSB<sup>+</sup>08], a novel tessellation technique, called *Adaptive Delaunay Tessellation* (ADT), was introduced in the context of computational mechanics as a tool to support Voronoi-

based nodal integration schemes in the Finite Element Method. Its main contribution is a simplified access to the element structure that supports nodal integration schemes based on a robust and unique transformation of the Delaunay triangulation. While focusing on the applications in linear elasticity, the presentation in [CSB<sup>+</sup>08] lacks rigorous proofs for the geometric properties of the ADT which are uniqueness of the ADT, connectedness of the ADT, and coverage of the Voronoi tiles by adjacent ADT tiles. These properties are essential for the computation of the nodal integration scheme. The pending proofs for these main claims are given in this chapter.

We start with a motivation for the ADT and introduce some notation used in the rest of this chapter in Section 9.3. Uniqueness and connectedness of the ADT are stated and proved in Sections 9.4.1 and 9.4.2. Section 9.4.3 introduces an alternative characterization of the ADT to simplify the proof of the last important property in Section 9.4.4 on the coverage of the Voronoi tile of a vertex by ADT elements adjacent to it. We discuss the ADT in Section 9.5 and conclude in Section 9.6.

### 9.2.1 Linear Elasticity and Nodal Integration in the Finite Element Method

The *Finite Element Method* encompasses a wide range of applications to model complex continuous phenomena based on a description composed of many small parts, called finite elements, which allow a structurally simpler definition of the phenomenon. In the field of computational mechanics, which is the main target of the ADT, one of the most basic and well-studied problems is that of linear elasticity, for which we sketch the key aspects in a simplified form.

The Finite Element Method typically requires the evaluation of the integral

$$\int_{\Omega} \nabla \psi \cdot f$$

for the so-called *test function*  $\psi$ , and some function  $f$ . In general,  $\psi$  is an interpolant that is defined piecewise over the elements  $\Omega_i$  of the domain. This allows the decomposition of the global integral into a sum of local integrals,

$$\sum_i \int_{\Omega_i} \nabla \psi \cdot f.$$

This is the heart of the Finite Element Method and allows the numerical assessment of the problem by concentrating on its local reformulation for each  $\Omega_i$  and the associated local interpolant  $\psi_i = \psi|_{\Omega_i}$ . Disadvantages of the element-based approach are the requirement of derivatives of the test function  $\psi$ , and numerical problems if the elements  $\Omega_i$  are badly shaped, as discussed in [She02].

One remedy for these problems is to turn to meshless methods, for which the discretization focuses on vertices rather than elements. A meshless approach based on natural neighbor coordinates has been proposed in [SMSB01]. Another meshless technique is *nodal integration*, as introduced by Chen et al. in [CWYY01]. Here, an alternative tessellation of the domain is considered, where the vertices  $\mathbf{v}_j$  are enclosed by polygons  $\mathcal{T}_j$ , with  $\Omega = \bigcup_j \mathcal{T}_j$

and the above integral becomes

$$\sum_j \int_{\mathcal{T}_j} \nabla \psi \cdot f,$$

assigning a part of the integral to each vertex  $\mathbf{v}_j$  rather than to an element  $\Omega_i$ . The canonical choice for this tessellation is the Voronoi diagram of the vertices  $\mathbf{v}_j$  of the background tessellation, restricted to  $\Omega$ . The application of Stokes' theorem allows to write

$$\int_{\mathcal{T}_j} \nabla \psi \cdot f = \int_{\partial \mathcal{T}_j} \mathbf{n} \cdot \psi \cdot f,$$

where  $\mathbf{n}$  is the outward pointing unit vector. By turning the area integral into a path integral, no derivatives of  $\psi$  are required, but now the local integration domains  $\mathcal{T}_i$  do no longer correspond to the elements  $\Omega_j$  of the background tessellation. Therefore, the computation of the path integral for each vertex can no longer be carried out on a per-element basis, but requires the evaluation of the test function  $\psi$  at arbitrary positions along  $\partial \mathcal{T}_j$ .

### 9.2.2 Voronoi Tile Coverage in the ADT

By covering the Voronoi tile of a vertex with a minimal set of adjacent polygons, the computational assessment of Voronoi-based nodal integration is greatly simplified. Figure 9.1 shows a set of scattered points covering a rectangular domain. The boundary of the Voronoi tile shown in Figure 9.1(b) is a convex polygon that intersects certain triangles in the Delaunay triangulation of the point set, which are not necessarily adjacent to the Voronoi site. The ADT, on the other hand, consists of polygons that are big enough to cover the whole Voronoi tile. Thus, by traversing the polygons adjacent to a vertex, one has access to an area that completely covers the Voronoi tile of that vertex.

## 9.3 The Adaptive Delaunay Tessellation

Before presenting the main findings, we introduce the necessary notation and repeat some important results.

We consider the *canonical open ball topology* on  $\mathbb{R}^2$  and denote by  $[\Omega]$  the *topological closure* of a set  $\Omega$ , by  $] \Omega [$  its *topological interior*, and by  $\partial \Omega := [\Omega] \setminus ] \Omega [$  its *topological boundary*. For a set  $\mathbf{X} \subset \mathbb{R}^2$  of points, we denote by  $\mathcal{C}(\mathbf{X}) = ] \mathcal{C}(\mathbf{X}) [ \cup \partial \mathcal{C}(\mathbf{X})$  the convex hull of  $\mathbf{X}$ .

We denote by  $(F, E)$  a partition of  $\mathcal{C}(\mathbf{X})$  into polygonal faces  $f \in F$  that join along edges  $e \in E$ . By  $\text{DEL}(\mathbf{X}) := (F_{\text{DEL}}, E_{\text{DEL}})$  we refer to a Delaunay triangulation of  $\mathbf{X}$ , composed of triangular faces  $f \in F_{\text{DEL}} \subset \mathbf{X}^3$  and edges  $e \in E_{\text{DEL}} \subset \mathbf{X}^2$ . We will use  $f$  and  $e$  to both denote the set of vertices from  $\mathbf{X}$  and their geometric realizations. Two faces  $f_1$  and  $f_2$  with  $f_1 \cap f_2 = e \in E_{\text{DEL}}$  are called neighbors with common edge  $e$ . We consider angles greater or equal  $\pi/2$  as obtuse and a triangle with an obtuse interior angle is called obtuse. The longest edge of an obtuse triangle  $f$  is opposite to its obtuse angle and will be denoted by  $e_f^> = \arg \max_{e \in f} |e|$ . Whenever the term  $e_f^>$  is used,  $f$  implicitly represents

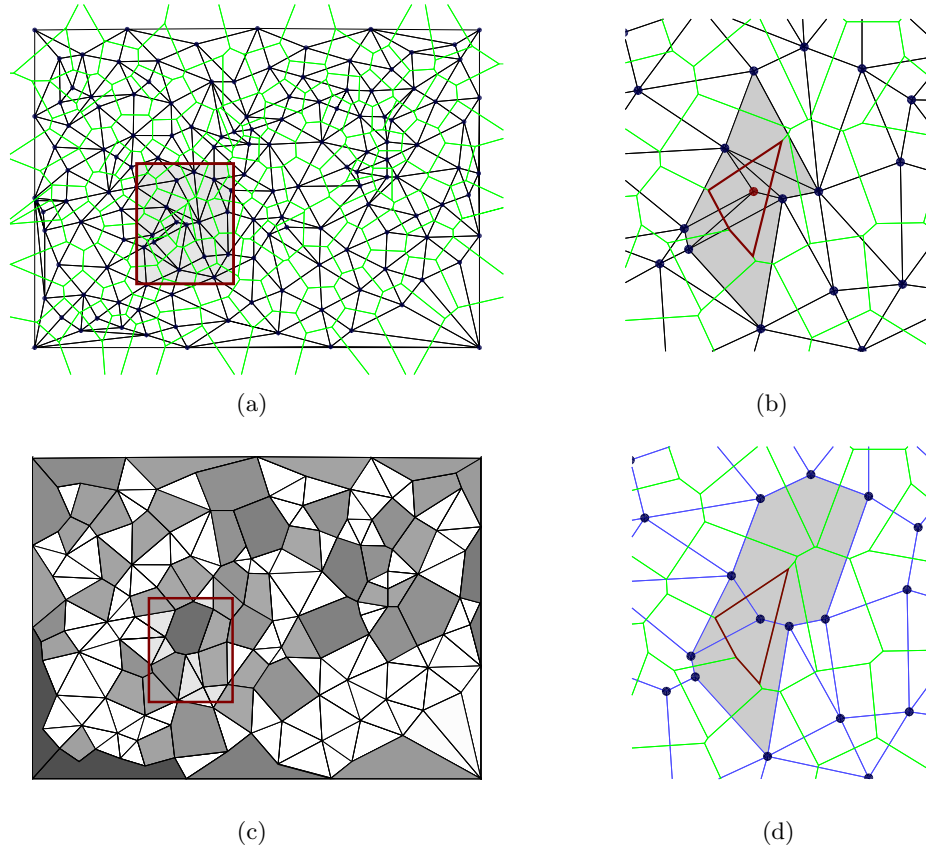


Figure 9.1: Example of an ADT tessellation. (a) The point cloud with its Voronoi diagram in green and the Delaunay triangulation in black. (b) The closeup of a vertex with shaded triangles depicting the necessary cover for its Voronoi tile. (c) The corresponding ADT with polygons shaded according to valence. (d) In the closeup, it is apparent that the adjacent ADT polygons cover the Voronoi tile.

an obtuse triangle. For a triangle  $f \in F_{\text{DEL}}$  we denote by  $\text{cc}(f)$  its circumcenter. Note that a triangle  $f$  is obtuse if it does not contain  $\text{cc}(f)$  in its interior and that  $\text{cc}(f)$  lies on the opposite side of  $e^>$  as the obtuse angle.

We denote by  $G = (\mathbf{X}, E)$ , a graph over  $\mathbf{X}$ , where  $\mathbf{X}$  represents the set of graph vertices and  $E \subset \mathbf{X} \times \mathbf{X}$  the set of edges. A sequence of vertices  $\rho = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ ,  $(\mathbf{x}_i, \mathbf{x}_{i+1}) \in E$ , is a *path* of length  $|\rho| = n$  if none of its edges appears more than once, i.e., edges are mutually distinct. A path is called *closed* if  $(\mathbf{x}_n, \mathbf{x}_1) \in E$ . We assume a path to be free of loops of length one, i.e., edges  $(\mathbf{x}_i, \mathbf{x}_i)$ . The Delaunay triangulation itself represents an undirected, connected graph  $G_{\text{DEL}} = (\mathbf{X}, E_{\text{DEL}})$ .

The Voronoi diagram of  $\mathbf{X}$  is the partition of  $\mathbb{R}^2$  into convex polygons  $\mathcal{T}_{\mathbf{x}}$ , called *Voronoi tiles* of  $\mathbf{x}$ , such that every point in  $\mathcal{T}_{\mathbf{x}}$  is not farther from  $\mathbf{x}$  than from any other vertex in  $\mathbf{X}$ . For each vertex  $\mathbf{x} \in \mathbf{X} \setminus \partial\mathcal{C}(\mathbf{X})$ , its Voronoi tile is equal to  $\mathcal{T}_{\mathbf{x}} = \mathcal{C}(\{\text{cc}(f) \mid f \in F_{\text{DEL}}, \mathbf{x} \in f\})$ , and we also call  $\mathbf{x}$  a Voronoi site.



With these definitions the *Adaptive Delaunay Tessellation* is defined as in [CSB<sup>+</sup>08]:

**Definition 9.1 (Adaptive Delaunay Tessellation)** *Let  $\mathbf{X} \subset \mathbb{R}^2$ ,  $\text{DEL}(\mathbf{X}) = (F_{\text{DEL}}, E_{\text{DEL}})$  and*

$$E^> = \{e_f^> \mid f \in F_{\text{DEL}} \wedge e_f^> \not\subset \partial\mathcal{C}(\mathbf{X})\}.$$

*The tessellation of  $\mathcal{C}(\mathbf{X})$  represented by  $(F_{\text{ADT}}, E_{\text{DEL}} \setminus E^>)$ , where  $F_{\text{ADT}}$  is the set of faces generated by merging triangles with common edges in  $E^>$ , is called the Adaptive Delaunay Tessellation  $\text{ADT}(\mathbf{X})$ .*

Thus, the ADT of a point set is the result of removing from a Delaunay triangulation of  $\mathbf{X}$  of each obtuse triangle the longest edge, if this is not a boundary edge. Since no new edges are generated in  $\text{ADT}(\mathbf{X})$ , each triangle  $f \in F_{\text{DEL}}$  is part of some polygon  $g \in F_{\text{ADT}}$ , which we denote by  $\mathcal{A}(f) = g \supset f \in F_{\text{DEL}}$ .

## 9.4 Geometric Properties of the ADT

In this section we will discuss some of the geometric properties of  $\text{ADT}(\mathbf{X})$ , i.e., uniqueness, connectedness, and the inclusion of Voronoi tiles. The last property states that the Voronoi tile of each  $\mathbf{x} \in \mathbf{X}$  is contained in the union of  $\mathbf{x}$ 's adjacent faces from  $\text{ADT}(\mathbf{X})$ .

### 9.4.1 Uniqueness of the ADT

**Proposition 9.2 (Uniqueness)** *For any non-collinear set of points  $\mathbf{X} \subset \mathbb{R}^2$ ,  $\text{ADT}(\mathbf{X})$  exists and is unique.*

**Proof 9.3** *We will show that  $E^>$  contains all edges of  $E_{\text{DEL}}$  that are non-unique.*

*For any non-collinear point set  $\mathbf{X}$ ,  $\text{DEL}(\mathbf{X})$  exists, and is non-unique, if there are at least  $n \geq 4$  points  $\mathbf{x}_1, \dots, \mathbf{x}_n$  with a common empty circumcircle  $\mathbf{C}$  with circumcenter  $\mathbf{c}$ , i.e.,  $\mathbf{X} \cap \mathbf{C} = \emptyset$ . The convex hull of these points can be triangulated arbitrarily by edges  $e_1, \dots, e_{n-3}$  without violating the Delaunay condition, see Figure 9.2(a). Every  $e_i$  connects two points not neighboring on the circumcircle and belongs to two triangles of the triangulation of  $\mathcal{C}(\{\mathbf{x}_1, \dots, \mathbf{x}_n\})$ . Because there is at most one non-obtuse triangle containing the circumcenter  $\mathbf{c}$ , every edge  $e_i$  belong to at least one obtuse triangle. Furthermore, every edge  $e_i$  is the longest edge of the obtuse triangle that lies on the opposite side of  $e_i$  with respect to  $\mathbf{c}$ . Thus,  $e_i \in E^>$ .*

*In case  $\mathbf{c}$  lies on an edge  $e_j$ , then  $e_j \in E^>$ , because at least one of its triangles is rectangular, see Figure 9.2(b). So, all  $e_i$  are in  $E^>$ , and  $E_{\text{DEL}} \setminus E^>$  is unique.*

□

**Remark 9.4** *If in the above setting the common edge  $e$  of two triangles is passing through the common circumcenter, both triangles are rectangular, i.e., obtuse, and  $e$  is removed.*

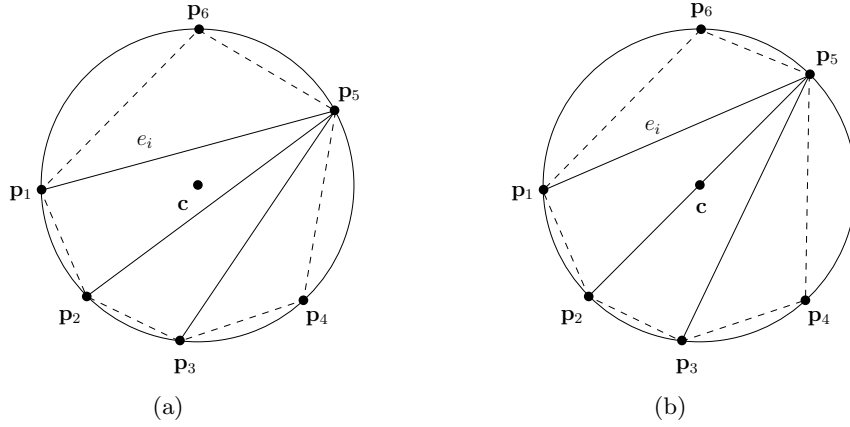


Figure 9.2: Points  $\mathbf{x}_1, \dots, \mathbf{x}_n$  on a common circumcircle  $\mathbf{C}$  with center  $\mathbf{c}$  inside a triangle (a) or on an edge  $e_j$  (b).

### 9.4.2 Connectedness of the ADT

Before stating the second geometric property, we introduce some results used in its proof. For every obtuse triangle  $f$  with longest edge  $e_f^>$  and neighbor  $f'$ , i.e.,  $f \cap f' = e_f^>$ , we use  $N(f) := f'$  to denote neighbor  $f'$  of  $f$  opposite to the obtuse angle. This relation imposes a sub-graph of the dual Delaunay triangulation on the faces of  $\text{DEL}(\mathbf{X})$ . For non-obtuse triangles or if  $e_f^> \subset \partial\mathcal{C}(\mathbf{X})$ , we set  $N(f) = \emptyset$ , and  $N(\emptyset) = \emptyset$ . This induces the directed graph

$$\begin{aligned} \vec{G}_{\text{DEL}} &:= (F_{\text{DEL}}, \vec{E}_{\text{DEL}}), \\ \vec{E}_{\text{DEL}} &:= \{ (f, N(f)) \mid f \in F_{\text{DEL}} \wedge N(f) \neq \emptyset \}. \end{aligned} \quad (9.1)$$

Each  $\vec{e} \in \vec{E}_{\text{DEL}}$  is associated with one Delaunay edge  $e \in E^>$ .

In the proof we use the following lemma, which implies that the lengths of the longest edges of obtuse triangles grow along a path in  $\vec{G}_{\text{DEL}}$ .

**Lemma 9.5** For  $f \in F_{\text{DEL}}$  with  $N(f) \neq \emptyset$ ,

$$e_f^> \neq e_{N(f)}^> \Rightarrow |e_f^>| < |e_{N(f)}^>|. \quad (9.2)$$

**Proof 9.6** We first show that there cannot be obtuse angles facing the same edge unless they are both right angles. Consider Figure 9.3(a) and assume  $\triangle ABC$  and  $\triangle CDA$  with common edge  $\overline{AC}$  are triangles in a Delaunay triangulation. Let  $D'$  be a point on the circumcircle  $\mathbf{C}$  of  $\triangle ABC$  right of  $\overline{AC}$ . Since  $\square ABCD'$  is a circular, simple quadrilateral,  $\angle ABC + \angle CD'A = \pi$ . By the Delaunay criterion,  $D$  must either be on or outside  $\mathbf{C}$ , and comparison to the inscribed angle yields  $\angle CDA \leq \angle CD'A = \pi - \angle ABC$ . The only obtuse angles that yield equality are  $\angle CDA = \angle ABC = \pi/2$ , in which case  $e_f^> = e_{N(f)}^>$ .

In all other cases, i.e.,  $e_f^> \neq e_{N(f)}^>$ , we find  $e_f^>$  adjacent to the obtuse angle in  $N(f)$  and

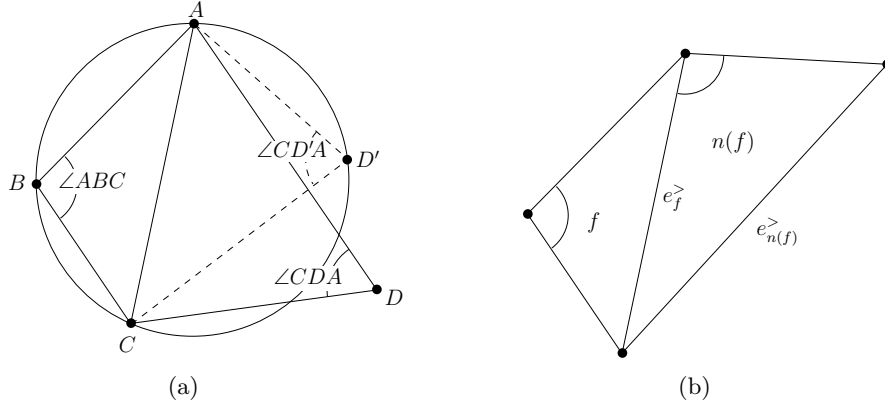


Figure 9.3: (a) Two Delaunay triangles with the circumcircle of  $\triangle ABC$ . (b)  $|e_f^>| < |e_{N(f)}^>|$ .

therefore shorter than  $e_{N(f)}^>$ , see Figure 9.3(b). This yields (9.2).

□

Now we can state the result which guarantees that the transformation from Delaunay to adaptive Delaunay does not lead to unconnected elements such as orphaned vertices or even disconnected sub-graphs.

**Proposition 9.7 (Connectedness)** For  $\text{ADT}(\mathbf{X}) = (F_{\text{ADT}}, E_{\text{ADT}})$  the graph  $G_{\text{ADT}} = (\mathbf{X}, E_{\text{ADT}})$  is connected.

**Proof 9.8**  $G_{\text{ADT}}$  is a sub-graph of  $G_{\text{DEL}}$ , and  $G_{\text{DEL}}$  is connected. An edge of  $G_{\text{DEL}}$  is not in  $G_{\text{ADT}}$  if and only if its dual edge is in  $\vec{G}_{\text{DEL}}$ . The smallest subset of  $G_{\text{ADT}}$  which can be disconnected consists of a single vertex. Boundary vertices cannot be disconnected, since the duals of boundary edges are by definition not in  $\vec{G}_{\text{DEL}}$ . An inner vertex in a Delaunay triangulation has a minimum valence of three, which means it cannot be disconnected in  $G_{\text{ADT}}$  unless there is a closed path of length greater or equal to three in  $\vec{G}_{\text{DEL}}$ .

We show that a path  $\rho$  can only be closed if it has length  $|\rho| = 2$ . Let without loss of generality  $\rho = (f_1, \dots, f_n)$  be a path in  $\vec{G}_{\text{DEL}}$ , i.e.,  $f_{i+1} = N(f_i)$ . First assume  $e_{f_i}^> \neq e_{f_{i+1}}^>$ ,  $1 \leq i < n$ . With (9.2) we get

$$|e_{f_1}^>| < |e_{f_2}^>| < \dots < |e_{f_n}^>|.$$

If  $\rho$  were closed,  $N(f_n) = f_1$  and  $e_{f_n}^> = f_1 \cap f_n$  is an edge of  $f_1$  but larger than  $e_{f_1}^>$ . Since  $e_{f_1}^>$  is the largest edge in  $f_1$ ,  $\rho$  cannot be closed.

Now assume there is an  $i$  such that  $e_{f_i}^> = e_{f_{i+1}}^>$ . Since elements in  $\rho$  are mutually distinct and  $N(f_{i+1}) = f_i$ ,  $f_{i+1}$  is the last element in  $\rho$  and  $n = i + 1$ . For  $\rho$  to be closed,  $N(f_n) = f_1$  must hold. Thus

$$N(f_n) = N(f_{i+1}) = f_i = f_1,$$

which leaves  $i = 1$  and  $n = 2$ , and  $\rho$  can only be closed if it has length 2. Since the longest closed path in  $\vec{G}_{\text{DEL}}$  has length  $|\rho| = 2$ ,  $G_{\text{ADT}}$  is connected. □

**Remark 9.9 (Implementation)** *From the above reasoning it appears that numerical instabilities might flip the inequalities and result in ambiguous or even inconsistent results. However, the only setting where the inequalities in (9.2) are “only just” fulfilled arises with co-circular points, where the uniqueness of the ADT should be forced by allowing an epsilon threshold.*

### 9.4.3 An Alternative Characterization for the ADT

In the following, we make use of an alternative characterization of the ADT, involving the following construction, see Figures 9.4(a) and 9.4(b).

**Definition 9.10 (Polygonal Extension)** *For each triangular face  $f$ , define*

$$PE(f) = ]\mathcal{C}(f \cup \{\text{cc}(f)\})[ \cup \{\text{cc}(f)\}$$

*as the Polygonal Extension of  $f$ , where  $\text{cc}(f)$  denotes the circumcenter of  $f$ .*

Note that  $PE(f)$  does not contain its boundary except for the circumcenter itself, which is necessary to cover the case of rectangular triangles, as we will see later. We know that if  $f$  is non-obtuse, we get

$$\text{cc}(f) \in ]f[, \quad \text{and} \quad ]f[ = PE(f). \tag{9.3}$$

If  $f$  is obtuse, then  $PE(f)$  extends over  $e_f^>$ , and

$$\text{cc}(f) \notin ]f[, \quad \text{and} \quad e_f^> \cap PE(f) \neq \emptyset. \tag{9.4}$$

The following lemma is required in the proof of Proposition 9.13.

**Lemma 9.11** *Let  $f$  be an obtuse triangle and  $f' = N(f) \neq \emptyset$ . Then,*

$$PE(f) \setminus f \subset PE(f'). \tag{9.5}$$

**Proof 9.12** *Consider triangles  $f = \triangle ABC$ ,  $f' = \triangle ACD$  and their circumcenters  $\text{cc}(f)$ ,  $\text{cc}(f')$  in Figure 9.4(c). It is sufficient to show that the excess area  $PE(f) \setminus f$  given by  $] \mathcal{C}(A, C, \text{cc}(f)) [$  is already contained in a subset of  $PE(f')$  given by  $] \mathcal{C}(A, C, \text{cc}(f')) [$ , as illustrated in Figure 9.4(d).*

*By definition,  $f$  shares the  $e_f^> = \overline{AC}$  with its neighbor  $f' = N(f)$ . The circumcenters  $\text{cc}(f)$  and  $\text{cc}(f')$  lie on the perpendicular bisector of  $e_f^>$  on the same side of  $e_f^>$  as  $D$ . By the Delaunay empty circumcircle criterion,  $D$  is outside the circumcircle of  $f$ , and on the*

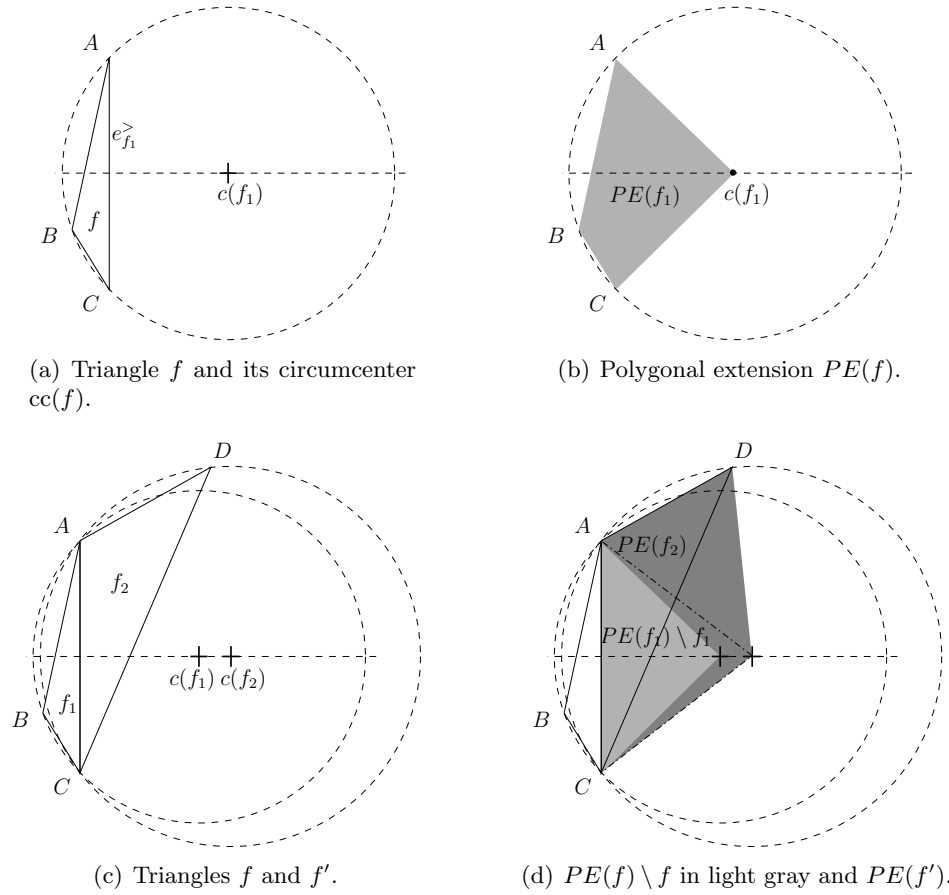


Figure 9.4: Geometric interpretation of the polygonal extension  $PE(f)$ .

side of  $e_f^>$  opposite to  $B$ , thus the distance of  $cc(f')$  to  $e_f^>$  is greater or equal to that of  $cc(f)$ . Consequently,

$$\mathcal{C}(\{A, C, cc(f)\}) \subset \mathcal{C}(\{A, C, cc(f')\}) \subset \mathcal{C}(\{A, C, cc(f'), D\}).$$

This relation also applies to the interior of these sets, which means

$$\begin{aligned} PE(f) \setminus f &= ]\mathcal{C}(\{A, C, cc(f)\})[ \\ &\subset ]\mathcal{C}(\{A, C, cc(f'), D\})[ \cup \{cc(f')\} = PE(f'), \end{aligned}$$

see Figure 9.4(d). Note that in case of a rectangular triangle, the circumcenter lies on its obtuse edge, and  $] \mathcal{C}(\{A, C, cc(f)\}) [ = \emptyset \subset PE(f')$  for arbitrary  $f'$ .

□

Lemma 9.11 implies an equivalent characterization for the set of edges  $E^>$  that are removed when transforming a Delaunay triangulation into an ADT.

**Proposition 9.13** Let  $\text{DEL}(\mathbf{X}) = (F_{\text{DEL}}, E_{\text{DEL}})$  and  $E^>$  as in Definition 9.1. Then

$$e \in E^> \Leftrightarrow \exists f \in F_{\text{DEL}} : e \cap PE(f) \cap ]\mathcal{C}(\mathbf{X})[ \neq \emptyset.$$

So, an adaptive Delaunay tessellation can also be created by merging all triangles, whose polygonal extensions intersect, into polygons.

**Proof 9.14** " $\Rightarrow$ ": If  $e \in E^>$ , then  $e = e_f^>$  for some obtuse  $f \in F_{\text{DEL}}$ , and  $e \cap ]\mathcal{C}(\mathbf{X})[ \neq \emptyset$ . Then, (9.4) yields also  $e \cap PE(f) \cap ]\mathcal{C}(\mathbf{X})[ \neq \emptyset$ .

" $\Leftarrow$ ": Let  $e \cap PE(f) \cap ]\mathcal{C}(\mathbf{X})[ \neq \emptyset$  for an  $f \in F_{\text{DEL}}$ . If  $e$  is an edge of  $f$  then  $e = e_f^>$  and  $e \in E^>$ . Otherwise,  $e \neq e_f^>$ , and  $e \cap PE(f) \setminus f \neq \emptyset$ . By Lemma 9.11 there exists an obtuse triangle  $f'$  such that  $e \cap PE(f') \neq \emptyset$ . This argument applies iteratively until  $e$  is an edge of  $f'$ . Since  $|F_{\text{DEL}}|$  is finite and the subset relation in Lemma 9.11 is strict, this iteration terminates with a  $f' \in F_{\text{DEL}}$  such that  $e$  is an edge of  $f'$ . The final  $f'$  exists, otherwise  $e$  would intersect the interior of a triangle and  $\text{DEL}(\mathbf{X})$  was no triangulation. □

The relation of triangles, polygonal extensions and ADT faces is captured in the following result.

**Proposition 9.15** For every  $f \in F_{\text{DEL}}$ ,

$$f \subseteq [PE(f)] \cap \mathcal{C}(\mathbf{X}) \subseteq \mathcal{A}(f). \quad (9.6)$$

**Proof 9.16** The left inclusion is true by construction of  $PE(f)$  and  $f \subseteq \mathcal{C}(\mathbf{X})$ .

The right inclusion is more difficult to show. If a triangle  $f$  is non-obtuse, or rectangular,  $\text{cc}(f) \in f$  and  $[PE(f)] = f \subseteq \mathcal{A}(f)$ . In all other cases,  $f$  is obtuse and  $\text{cc}(f) \notin f$ . If  $e_f^>$  is on the boundary of  $\mathcal{C}(\mathbf{X})$ , then  $[PE(f)] \cap \mathcal{C}(\mathbf{X}) = f \subseteq \mathcal{A}(f)$ . Otherwise,  $f$  is obtuse and there is a triangle  $f' = N(f)$  such that

$$\begin{aligned} & PE(f) \setminus f \subseteq PE(f') && \text{by (9.5),} \\ \Rightarrow & PE(f) \subseteq f \cup PE(f'), \\ \Rightarrow & [PE(f)] \subseteq f \cup [PE(f')], \\ \Rightarrow & [PE(f)] \cap \mathcal{C}(\mathbf{X}) \subseteq f \cup [PE(f')] \cap \mathcal{C}(\mathbf{X}). \end{aligned}$$

Since  $\mathcal{A}(f) = \mathcal{A}(f')$ , the claim holds if we can prove  $[PE(f')] \cap \mathcal{C}(\mathbf{X}) \subseteq \mathcal{A}(f')$ . The argument applies repeatedly until  $f'$  is non-obtuse, rectangular, or has its obtuse edge on the boundary of  $\mathcal{C}(\mathbf{X})$  following a path  $\rho$  in  $\vec{G}_{\text{DEL}}$ . In the proof for Proposition 9.7 we showed that every path  $\rho$  ends in a triangle  $f$  that is either non-obtuse, has  $e_f^>$  on the convex hull, or in a loop consisting of two rectangular triangles. Therefore, the repeated application of the argument terminates and the claim is proved. □

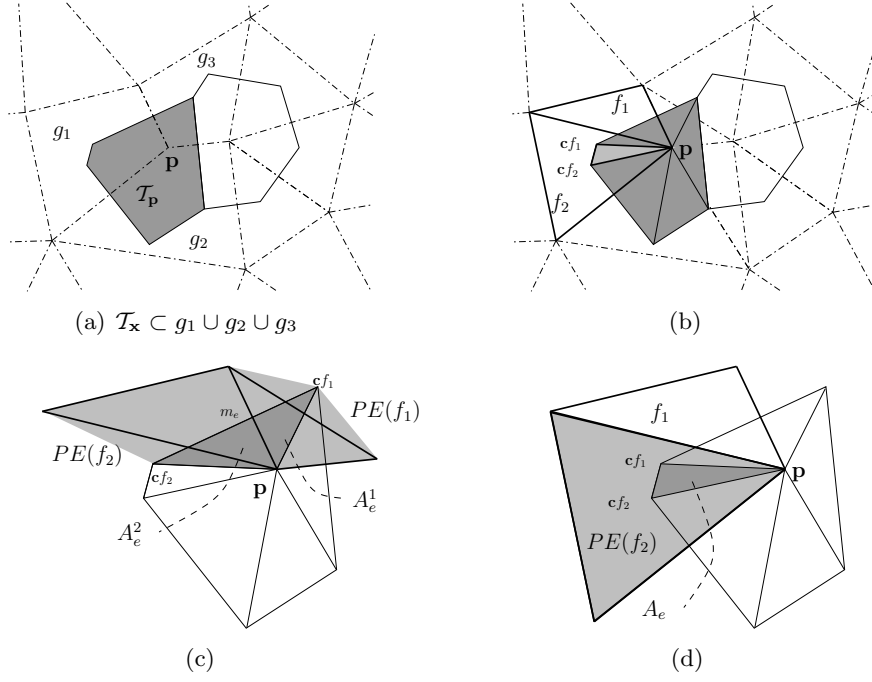


Figure 9.5: Voronoi coverage of adjacent ADT tiles.

An important observation following immediately from the above Proposition is given in the following corollary:

**Corollary 9.17** For every  $f \in F_{\text{DEL}}$  with  $\text{cc}(f) \in \mathcal{C}(\mathbf{X})$  :  $\text{cc}(f) \in \mathcal{A}(f)$ .

#### 9.4.4 Coverage of Voronoi Tiles

The last property we are going to prove is the coverage of a vertex' Voronoi tile by its adjacent ADT tiles, as illustrated in Figure 9.5(a). With this property, the adjacency information contained in the ADT is sufficient to access the whole area covered by the Voronoi cell of a vertex.

**Proposition 9.18 (Inclusion of Voronoi Tile)** If  $\mathcal{T}_{\mathbf{x}}$  denotes the Voronoi tile of  $\mathbf{x} \in \mathbf{X}$ , then

$$\mathcal{T}_{\mathbf{x}} \cap \mathcal{C}(\mathbf{X}) \subset \bigcup_{\substack{g \in F_{\text{ADT}} \\ g \ni \mathbf{x}}} g.$$

**Proof 9.19** First assume that  $\mathbf{x} \notin \partial \mathcal{C}(\mathbf{X})$ , i.e.,  $\mathcal{T}_{\mathbf{x}}$  is finite as in Figure 9.5(a).  $\mathbf{x}$  has a set of adjacent triangles  $\{f\}_{f \ni \mathbf{x}}$ , which in turn are adjacent via the edges  $\{e\}_{e \ni \mathbf{x}}$ . For each edge  $e \ni \mathbf{x}$  and triangles  $f_1 \cap f_2 = e$ , we get a new triangle  $A_e = \mathcal{C}(\{\mathbf{x}, \text{cc}(f_1), \text{cc}(f_2)\})$ , see Figure 9.5(b). This partitions  $\mathcal{T}_{\mathbf{x}}$ ,

$$\mathcal{T}_{\mathbf{x}} = \bigcup_{e \ni \mathbf{x}} A_e.$$

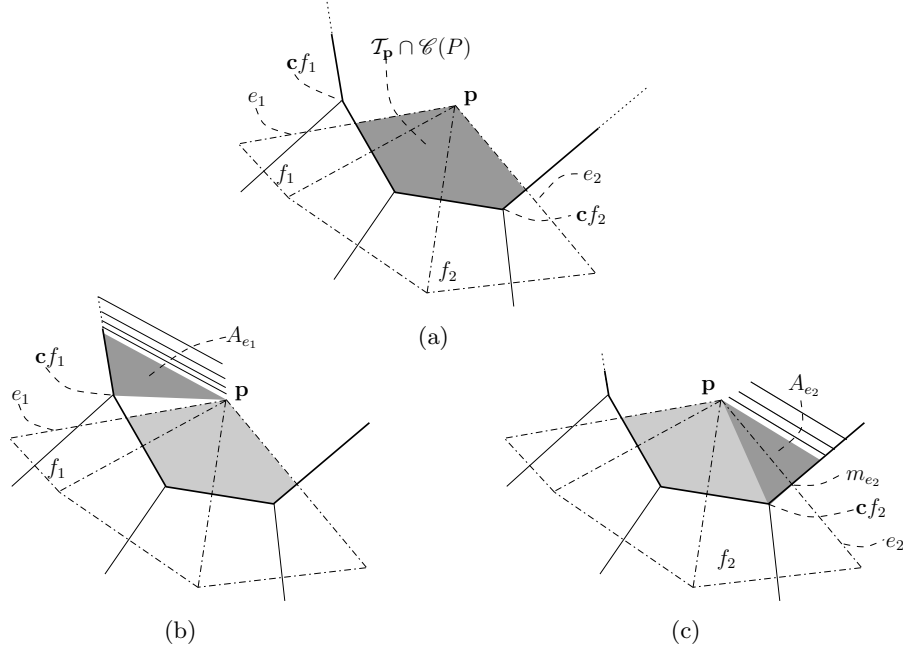


Figure 9.6: Coverage of ADT tiles on the boundary of  $\mathcal{C}(\mathbf{X})$ . (a) For  $\mathbf{x} \in \partial\mathcal{C}(\mathbf{X})$ ,  $e_1$  and  $e_2$  have only one adjacent triangle each. (b) Contribution of  $A_{e_1}$  to  $\mathcal{T}_{\mathbf{x}}$ . (c) Contribution of  $A_{e_2}$  to  $\mathcal{T}_{\mathbf{x}}$ .

So we will show that each  $A_e \cap \mathcal{C}(\mathbf{X})$  is covered by  $\mathcal{A}(f_1) \cup \mathcal{A}(f_2)$  for two cases:

- $cc(f_1)$  and  $cc(f_2)$  are on opposite sides of  $e$ , see Figure 9.5(c): So  $cc(f_1)$  and  $cc(f_2)$  are located on the perpendicular bisector of  $e$  that intersects  $e$  in  $m_e$ . We can split  $A_e$  into  $A_e^1 = \mathcal{C}(\{\mathbf{x}, cc(f_1), m_e\})$  and  $A_e^2 = \mathcal{C}(\{\mathbf{x}, cc(f_2), m_e\})$  with  $A_e = A_e^1 \cup A_e^2$ . Since  $m_e, \mathbf{x}, cc(f_1) \in [PE(f_1)]$ , (9.6) guarantees that  $A_e^1 \cap \mathcal{C}(\mathbf{X}) \subset \mathcal{A}(f_1)$  and analogously for  $A_e^2$ .
- $cc(f_1)$  and  $cc(f_2)$  are on the same side of or on  $e$ , see Figure 9.5(d): Without loss of generality, let  $f_1$  be obtuse. We know  $cc(f_1) \in PE(f_1) \setminus f_1$  and by (9.5),  $cc(f_1) \in PE(f_2)$ . Since  $cc(f_2) \in PE(f_2)$  as well,  $cc(f_1), cc(f_2), \mathbf{x} \in [PE(f_2)]$ , and from the convexity of  $[PE(f_2)]$  it follows that  $A_e \cap \mathcal{C}(\mathbf{X}) \subset PE(f_2) \cap \mathcal{C}(\mathbf{X}) \subset \mathcal{A}(f_2)$ .

Now consider the case that  $\mathbf{x} \in \partial\mathcal{C}(\mathbf{X})$ , and  $\mathcal{T}_{\mathbf{x}}$  becomes infinite. The argument goes as above except for the two edges  $e_1, e_2 \subset \partial\mathcal{C}(\mathbf{X})$  adjacent to  $\mathbf{x}$  which have only one adjacent triangle each, say  $f_1$  respective  $f_2$ , as illustrated in Figure 9.6(a). Corresponding to the above two cases, let without loss of generality  $cc(f_1) \notin \mathcal{C}(\mathbf{X})$  like in Figure 9.6(b). Because  $A_{e_1}$  is outside the convex hull, it does not contribute to  $\mathcal{T}_{\mathbf{x}} \cap \mathcal{C}(\mathbf{X})$  and needs not be considered. Now, let without loss of generality  $cc(f_2) \in \mathcal{C}(\mathbf{X})$  like in Figure 9.6(c). The contribution of  $A_{e_2}$  to  $\mathcal{T}_{\mathbf{x}} \cap \mathcal{C}(\mathbf{X})$  is given by  $\mathcal{C}(\mathbf{x}, m_{e_2}, cc(f_2)) \subset f_2 \subset \mathcal{A}(f_2)$ .

Consequently, all parts  $A_e \cap \mathcal{C}(\mathbf{X})$  are covered by ADT polygons, which are the covers of adjacent Delaunay triangles and thus also adjacent to  $\mathbf{x}$ .

□



This concludes our treatment of some essential and important properties of the ADT. A discussion of aspects of the ADT that bear practical relevance is given next.

### 9.4.5 Further Remarks

As a guide to the proofs we include an informal description of the reasoning.

#### 9.4.5.1 Definition of the ADT

From every point set we can compute its Delaunay triangulation. In this triangulation we merge every triangle with an interior angle of  $\pi/2$  or more to its neighbor opposite to the obtuse angle, thus effectively forming polygons. If the opposite of an obtuse angle is outside the convex hull, we don't merge. The result of this process is called the Adaptive Delaunay Tessellation ADT.

#### 9.4.5.2 The ADT is Unique

In the definition of the ADT, sets of triangles are merged into polygons based on their largest interior angle. As long as the set of triangles is unique, the ADT is unique as well. But as soon as four or more points have a common empty circumcircle, the triangulation among those is non-unique. In the ADT, all triangles with the same empty circumcircle will be merged into a single polygonal element, thus removing all ambiguity.

#### 9.4.5.3 The ADT is Connected

By merging triangles, edges are removed from the Delaunay triangulation. If this is overdone, the elements of the triangulation might become disconnected, e.g., all edges could be removed from a vertex, leaving it hanging in the void. The elements of the initial Delaunay triangulation can only become disconnected if all edges are removed along some closed path around a part of the triangulation. In the definition of the ADT, we merge a triangle to its neighbor if the triangle is obtuse, meaning we remove its longest edge. If the neighbor itself is obtuse, it will again be merged to its neighbor, removing its longest edge. Now, as this process goes on, the removed edges can only become longer. In order to do remove all edges around a part of the triangulation, this path needs to be closed.

As the longest edges become longer and longer, and since every triangle can only have one obtuse angle, we can never merge a triangle to another triangle that was merged into the same polygon before.

An exception are rectangular triangles with right angles opposite to their common edge. Those are merged to each other and do not contradict the above.

#### **9.4.5.4 Alternative Characterization**

By merging each obtuse triangle to its neighbor, we effectively merge a set of triangles into one polygon. Instead of defining this set by adjacency over obtuse edges, we can also define it via the intersection of the convex hull of the triangle vertices and the triangle's circumcenter, called polygonal extension. Thus, the set of triangles that are merged into a single polygon can also be defined as the set of all triangles whose polygonal extensions intersect.

As soon as a triangle is obtuse, its circumcenter is outside and the polygonal extension intersects at least its neighbor triangle. It can be shown that the excess area produced by the outside circumcenter is a subset of the polygonal extension of the neighbor triangle; this property is important as a termination criterion in the final proof.

#### **9.4.5.5 Coverage of Voronoi Tiles**

The Voronoi tile of a vertex is always a convex polygon that contains the corresponding vertex but no more. We claim that this polygon is a subset of the union of all ADT polygons that are adjacent to the site. Special care must be taken if the Voronoi tile extends past the convex hull of the data set, but the essential reasoning is not affected by that.

A finite Voronoi tile of a vertex is identical to the convex hull of the circumcenters of the Delaunay triangles adjacent to that vertex. A trivial triangulation of the Voronoi tile is therefore given by connecting the circumcenters to the vertex, where each of the new triangles corresponds to an edge of the Delaunay triangulation. With the alternative characterization of the ADT from before, it is easy to show that each of these triangles is contained within the polygonal extensions of the two Delaunay triangles whose circumcenters define it.

### **9.5 Discussion**

The ADT is optimal only under specific conditions, i.e., if the vertex positions cannot be changed, and the Voronoi tile of each vertex needs to be covered by adjacent polygons. A very homogeneous point set with all non-obtuse triangles is shown in Figure 9.7(a). In this setting, the Delaunay triangulation and the ADT are identical. Figure 9.7(b) shows a point set in which ill-shaped triangles are merged into ADT polygons, thus improving the overall tile shape. Figure 9.7(c) demonstrates a situation where the ADT produces non-convex polygons, and the extreme case of a “hanging edge” is illustrated in Figure 9.7(d). Apart from these artifacts inside the convex hull, which are rarely faced in practice, badly shaped polygons frequently occur near the convex hull as a result of many degenerate triangles, as illustrated in Figure 9.7(e). Arguably, the boundary polygons are of better shape than the triangles they are composed of. In [CSB<sup>+</sup>08], a novel polygon condition number has been introduced as a quality measure of polygonal tessellations. Extensive statistical analysis of the ADT has shown that it considerably improves the mean condition number of most

unstructured triangulations over interior elements, and results in moderate improvement for boundary elements. The issues inside the convex hull can be handled by allowing vertices to move, and a regularization step could lead to more well-shaped triangles and thus better ADT-polygons. Furthermore, the boundary issues can be solved by insertion of an adequate amount of vertices along the convex hull.

## 9.6 Conclusion

Certain nodal integration schemes, which are meshless extension of the Finite Element Method, lead to new requirements on the background tessellation of the domain which are no longer met by the popular Delaunay triangulation. Particularly nodal integration schemes based on the Voronoi diagram of the domain vertices entail the integration of an interpolant along the boundary of Voronoi tiles. This leads to a desirable property of the background tessellation referred to as the *coverage of Voronoi tiles*. To this end, the Adaptive Delaunay Tessellation for planar domains has been introduced before, but no rigorous proofs have been given.

In this chapter we have introduced a simple, constructive definition of the Adaptive Delaunay Tessellation based on the Delaunay triangulation of a point set, and showed that it is unique, connected, and has the coverage of Voronoi tiles property. We furthermore provided a critical assessment of some shortcomings of the ADT that need to be considered in applications. This provides a profound theoretical background for the application of this new tessellation for nodal integration schemes in the Finite Element Method.

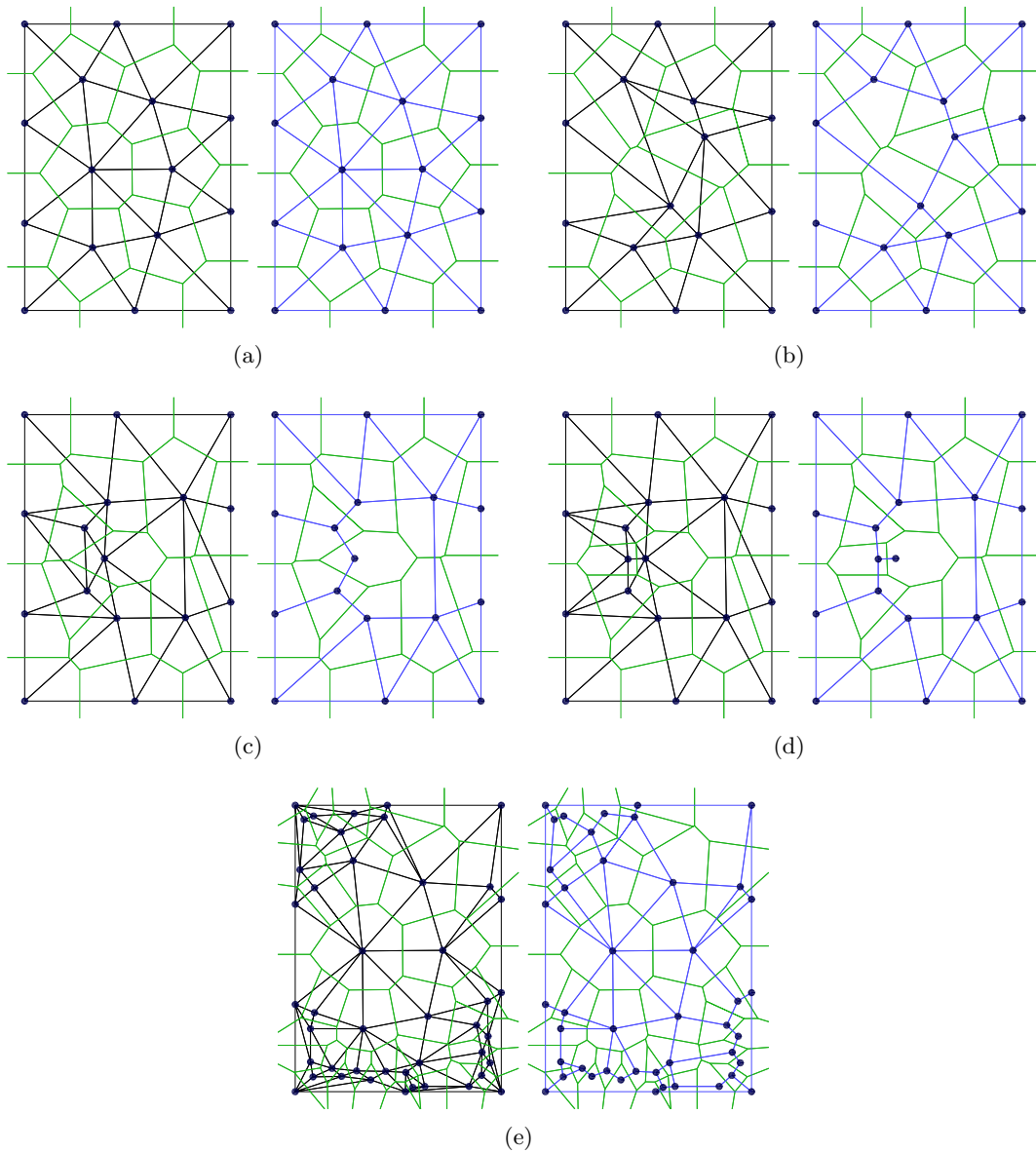


Figure 9.7: Several levels of badness in the point cloud. (a) Very regular setting. (b) Some obtuse triangles. (c) Setting leading to non-convex ADT tiles. (d) Setting leading to flapping edge, (e) Setting leading to bad boundary elements.

# 10 Conclusion

This thesis provides a spectrum of topics that underlines the versatility of the concept of natural neighbors.

## B-splines from Voronoi diagrams

Our investigation of an idea by Farin to generalize B-spline surfaces to a similar structure defined over the Voronoi diagram of a set of scattered points showed that the positive result for the univariate setting does not carry over to two dimensions. However, the initial idea was successfully modified by González et al., who considered Sibson’s interpolant under another interpretation of the de Boor algorithm. Their method indeed exhibits B-spline like behavior of the resulting surface, while the surface still is only  $C^0$  at the data points.

## Implementing Hiyoshi’s interpolant

Hiyoshi introduced a high quality globally  $C^2$  natural neighbor interpolant. However, the implementation of the underlying  $C^2$  natural neighbor coordinates is not straightforward. We provided guidelines for the implementation of Hiyoshi’s coordinates in arbitrary dimensions based on an alternative representation of the Voronoi tile in a power diagram, and the recursive volume computation formula of Lasserre. We presented a concrete implementation of the proposed approach for two dimensions. As a result of our discussion of this approach, we discourage its application for dimensions greater than three due to both the additional efforts on the implementation side and the increased runtime complexity. By considering Voronoi tiles in their half-space representation, we also provide guidelines for the computation of Laplace and Sibson coordinates that generalize to arbitrary dimension without any change in the complexity of implementation.

We furthermore discovered a severe complexity issue in Hiyoshi’s interpolant for data sets exhibiting large numbers of natural neighbors. The computation of the control point structure of the quintic Bézier simplex used in the interpolation of Hessians and gradients has a complexity of  $\mathcal{O}(m^5)$  in the number  $m$  of natural neighbors.

## Generating Derivatives

Related to the implementation of Hiyoshi’s  $C^2$  interpolant is the generation of derivatives, which are required by the interpolant, but are only seldom provided with the data set. E.g., Hiyoshi’s interpolant requires derivatives up to second order. To generate higher order derivatives by the standard method of fitting Taylor polynomials of a certain degree, a neighborhood is required that is large enough to prevent an under-constrained fitting problem. We proposed as one solution to use the  $d$ -times iterated natural neighborhood in the Voronoi diagram. The other solution took an iterative approach, fitting Taylor polynomials of degree  $d$  to known Taylor polynomials of degree  $d - 1$  to the natural

neighborhood. In our experiments, both approaches performed robustly.

### Discrete Harmonic Functions

Discrete harmonic functions are well approximated by generalized barycentric coordinates, and if modeled on point clouds, natural neighbor coordinates provide a Laplacian approximation that continuously (smoothly) depends on the coordinates of the data sites.

Potential applications for this result lie in computational mechanics. Furthermore, there may be a generalization to piecewise linear manifolds when considering the implicit Delaunay tessellation of [FSBS06], or when considering natural neighbor interpolation on smooth manifolds like in [Flo03a].

### Natural Neighbor Extrapolation

Extrapolation is the evaluation of an interpolating function outside the convex hull of the constituting data. For local scattered data interpolation schemes, this aspect has not seen a satisfactory solution to this day. All the local schemes proposed in the past are *exact* schemes, meaning they pick up the value of some interpolant in the convex hull and extend it continuously towards the outside. One problem showing up with any triangulation of the convex hull of some points is the unavoidable existence of sliver triangles near the boundary of the convex hull.

We have introduced a couple of new criteria by which to assess and judge extrapolation methods. We claim that by dropping the requirement of exactness, the degenerate cases near the convex hull can be overcome. To this end we have proposed a general framework for natural neighbor interpolation that builds on the concept of *dynamic ghost points*. The ghost points are placed depending on the query position such that their coordinates smoothly depend on that of the query position and the extended convex hull always covers the query position. In this framework, any natural neighbor interpolant can be evaluated transparently inside and outside the convex hull.

### Adaptive Delaunay Tessellation

The Adaptive Delaunay Tessellation is a new technique to support Voronoi-based nodal integration schemes in that it provides a background tessellation in which the Voronoi cell of a vertex is covered by the ADT tiles adjacent to that vertex. The basic properties uniqueness, connectedness, and coverage of Voronoi tile have been shown, and further research on that technique can build upon a solid theoretical background.

## Future Work

In the course of this thesis, some questions were left unanswered and some ideas formed that could not be followed. Some that seem especially interesting for future work are described here.

Sibson coordinates have originally been introduced based on a vector identity defined by ratios of volumes of overlap in the Voronoi diagram with and without a query position. Similar geometric considerations can be applied to higher order Voronoi diagrams, in which it might be possible to find an equivalent formulation that extends to larger

neighborhoods of data sites and implicitly increases the resulting smoothness.

The definition of bivariate splines by Nematu in [Nea04] via Delaunay configurations suggests a close relation to ordinary and higher order Voronoi diagrams. It would be insightful if a connection to natural neighbor interpolation could be established.

The method proposed by Gonzáles et al. in [GCD07] shows potential for the definition of globally smooth spline-like surfaces over Voronoi diagrams. The  $C^0$  limitation at the data sites results from the use of Sibson coordinates in the definition of the convex combination that produces the surface. As an alternative, Farin's interpolant could be applied to overcome these smoothness issues.

There is strong visual evidence that Clarkson's interpolant is actually only  $C^0$  at the data sites contrary to a conjecture by Flötotto in [Flö03b]. This evidence needs to be supported by a sound mathematical counterexample. Despite the possible lack of smoothness at the data sites, the idea underlying Clarkson's interpolant is worthwhile extending, and maybe it is even possible overcome the  $C^0$  artifacts.

Based on Gonzáles coordinates, it should be possible to generate derivative information in scattered by directly fit higher order Taylor polynomials to a weighted set of neighbors. Because Gonzáles coordinates should be extendable to the  $n$ -ring neighborhood of a data site, this approach would maintain the "similar output from similar input" property of natural neighbor interpolation for the generated derivatives. The advantage over the iterated derivative generation introduced in Section 6.3 would be the option to achieve polynomial precision for the generated derivatives.

Natural neighbor interpolants can be used to fit coarse approximations to densely sampled data sets, thus effectively implementing scattered data approximation. Assuming a good placement strategy for the interpolation sites can be found, the values and derivatives can easily be determined in a least squares sense. Flötotto proved in [Flö03b] that Sibson coordinates still comprise valid local coordinates if computed in a power diagram with arbitrary power weights. These power weights provide an additional degree of freedom for scattered data approximation which it would be interesting to investigate.

## *Conclusion*



# Bibliography

- [ACK01] N. Amenta, S. Choi, and R. K. Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–266. ACM Press, 2001. 32
- [ACSTD07] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Proc. Symp. on Geometry Processing*, pages 39–48, 2007. 32
- [ADA07] L. Antani, C. Delage, and P. Alliez. Mesh sizing with additively weighted voronoi diagrams. In *Proc. 16th Int. Meshing Roundtable*, 2007. 173
- [ADKS05] A. D. Alexandrov, N. S. Dairbekov, S. S. Kutateladze, and A. B. Sossinsky. *Convex Polyhedra*. Springer Monographs in Mathematics. Springer, 2005. 12
- [Aki78] H. Akima. A method of bivariate interpolation and smooth surface fitting for irregularly distributed data points. *ACM Transactions on Mathematical Software*, 4(2):148–159, June 1978. 132, 133, 135
- [Aki84] H. Akima. On estimating partial derivatives for bivariate interpolation of scattered data. *Rocky Mountain Journal of Mathematics*, 14:41–52, 1984. 90, 129
- [Ale06] M. Alexa. Mesh editing based on discrete Laplace and Poisson models. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, pages 51–59, New York, NY, USA, 2006. ACM. 116
- [Alf84] P. Alfeld. Triangular extrapolation. Technical report, University of Utah, Salt Lake City, 1984. 128, 131, 132, 133, 134, 135, 136, 138, 140
- [Alf85] P. Alfeld. Derivative generation from multivariate scattered data by functional minimization. *Computer Aided Geometric Design*, 2:281–296, 1985. 29, 90, 129, 132, 133, 135
- [Alf89] P. Alfeld. Scattered data interpolation in three or more variables. *Mathematical Methods in Computer Aided Geometric Design*, pages 1–33, 1989. 89
- [AMG98] F. Anton, D. Mioc, and C. Gold. Local coordinates and interpolation in a Voronoi diagram for a set of points and line segments. In *Proceedings of the 2nd Voronoi Conference on Analytic Number Theory and Space Tillings*, pages 9–12, 1998. 43, 45, 56
- [AMG04] F. Anton, D. Mioc, and C. Gold. Line Voronoi diagram based interpolation and application to digital terrain modelling. In *ISPRS - XXth Congress, Vol.2*. International Society for Photogrammetry and Remote Sensing, 2004. 44, 45, 56

## Bibliography

- [Aur91] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing surveys*, 23(3):345–405, 1991. 14, 32
- [BBU06a] T. Bobach, M. Bertram, and G. Umlauf. Comparison of Voronoi based scattered data interpolation schemes. In *Proc. Visualization, Imaging and Image Processing*, pages 342–348, 2006. 56
- [BBU06b] T. Bobach, M. Bertram, and G. Umlauf. Issues and implementation of  $C^1$  and  $C^2$  natural neighbor interpolation. In *Proceedings of the 2<sup>nd</sup> International Symposium on Visual Computing*, Nov. 2006. 71
- [BBU06c] T. Bobach, M. Bertram, and G. Umlauf. Issues and implementation of  $C^1$  and  $C^2$  natural neighbor interpolation. *Advances in Visual Computing*, 2, 2006. 71
- [BC00] J. Boissonnat and F. Cazals. Natural neighbour coordinates of points on a surface. Technical Report 4015, INRIA-Sophia., 2000. 52, 56
- [BE03] M. Bern and D. Eppstein. Möbius-invariant natural neighbor interpolation. In *SODA '03: Proc. fourteenth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 128–129, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics. 43
- [BEF00] B. Bueler, A. Enge, and K. Fukuda. Exact volume computation for polytopes: a practical study. *DMV Seminar*, 29:131–151, 2000. 54, 71, 72, 74, 75
- [BF04] J.-D. Boissonnat and J. Flötotto. A coordinate system associated with points scattered on a surface. *Computer-Aided Design*, 36(2):161–174, 2004. 52
- [BFHU08] T. Bobach, G. Farin, D. Hansford, and G. Umlauf. Natural neighbor extrapolation using ghost points. *Computer-Aided Design*, In Press, Corrected Proof:–, 2008. 127
- [BH96] F. Bossen and P. Heckbert. A pliant method for anisotropic mesh generation. In *Proc. 5th Int. Meshing Roundtable*, pages 63–74, 1996. 173
- [BHFU07] T. Bobach, D. Hansford, G. Farin, and G. Umlauf. Discrete harmonic functions from local coordinates. In *Mathematics of Surfaces XII*, 2007. 115
- [BIK<sup>+</sup>97] V. V. Belikov, V. D. Ivanov, V. K. Kontorovich, S. A. Korytnik, and A. Yu. Semenov. The non-Sibsonian interpolation: A new method of interpolation of the values of a function on an arbitrary set of points. *Computational Mathematics and Mathematical Physics*, 37(1):9–15, 1997. 36, 37, 56
- [Bol98] B. Bollobás. *Modern Graph Theory*. Springer, 1998. 17, 116
- [Bro97] J. L. Brown. Systems of coordinates associated with points scattered in the plane. *Computer Aided Geometric Design*, 14:547–559, 1997. 41, 130, 133, 135
- [BS95] J. Braun and M. Sambridge. A numerical method for solving partial differential equations on highly irregular grids. *Nature*, 376:655–660, 1995. 32, 54, 71
- [BS07] A. Bobenko and B. Springborn. A discrete Laplace-Beltrami operator for simplicial surfaces. *Discrete and Computational Geometry*, 38(4):740–756, 2007. 116

- [BU06] T. Bobach and G. Umlauf. *Natural Neighbor Interpolation and Order of Continuity*, pages 69–86. Lecture Notes in Informatics. Springer, 2006. 31
- [Cam94] P. J. Cameron. *Combinatorics: Topics, Techniques, Algorithms*. Cambridge University Press, 1994. 10
- [CCD02] E. Cueto, B. Calvo, and M. Doblaré. Modelling three-dimensional piecewise homogeneous domains using the  $\alpha$ -shape-based natural element method. *International Journal of Numerical Methods in Engineering*, 54:871–897, 2002. 32, 149
- [CDG00] E. Cueto, M. Doblaré, and L. Gracia. Imposing essential boundary conditions in the natural element method by means of density-scaled  $\alpha$ -shapes. *International Journal of Numerical Methods in Engineering*, 49:519–546, 2000. 32, 149
- [CFL82] N. H. Christ, R. Friedberg, and T. D. Lee. Weights of links and plaquettes in a random lattice. *Nuclear Physics B*, 210(3):337–346, 1982. 36, 56
- [CG92] C. Connolly and R. Grupen. Applications of harmonic functions to robotics. *Journal of Robotic Systems*, 10(7):931–946, 1992. 116
- [CGA08] CGAL. CGAL - Computational Geometry Algorithms Library. available online, 2008. <http://www.cgal.org>. 53, 54, 81
- [Cla96] K. Clarkson. Convex hulls: Some algorithms and applications. Presentation at Fifth MSI-Stony Brook Workshop on Computational Geometry, 1996. 46, 50, 56
- [CSB<sup>+</sup>08] A. Constantiniu, P. Steinmann, T. Bobach, G. Farin, and G. Umlauf. The adaptive Delaunay tessellation: A neighborhood covering meshing technique. *Computational Mechanics*, to appear:n/a, 2008. 173, 174, 177, 186
- [CT65] R. W. Clough and J. L. Tocher. Finite element stiffness matrices for analysis of plates in bending. In *Proc. Conf. Matrix Methods in Struct. Mech., Air Force Inst. of Tech., Wright-Patterson AFB, Ohio*, pages 515–545, 1965. 29, 89
- [CWYY01] J. Chen, C. Wu, S. Yoon, and Y. You. A stabilized conforming nodal integration for Galerkin mesh-free methods. *Int. J. Numer. Methods Eng.*, 50:435–466, 2001. 174
- [dB87] C. de Boor. B-form basics. *Geometric Modelling - Algorithms and New Trends*, pages 131–148, 1987. 18, 19, 20
- [Des44] R. Descartes. La disposition de la matiere dans le systeme solaire et ses environs, 1644. 32
- [Die05] R. Diestel. *Graph Theory*. Springer, 2005. 17, 116
- [Dir50] G. Dirichlet. Über die Reduktion der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen. *Journal für die Reine und Angewandte Mathematik*, 40:209–227, 1850. 32
- [DKG05] S. Dong, S. Kircher, and M. Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer Aided Geometric Design: Special Issue on Geometry Processing*, 22(5):392–423, 2005. 116

## Bibliography

- [Duc77] J. Duchon. Splines minimizing rotation-invariant semi-norms in sobolev spaces. *Constructive Theory of Functions of Several Variables*, pages 85–100, 1977. 135
- [Ei08] EMS-i. Watershed modeling system (wms) v8.0. [http://www.ems-i.com/wmshelp/Scattered\\_Data/Interpolation/Natural\\_Neighbor\\_Interpolation.htm](http://www.ems-i.com/wmshelp/Scattered_Data/Interpolation/Natural_Neighbor_Interpolation.htm), 2008. 1204 W. South Jordan Parkway, Suite B South Jordan, UT 84095-4612. 132
- [Far86] G. Farin. Triangular Bernstein–Bézier patches. *Computer Aided Geometric Design*, 3(2):83–127, 1986. 21
- [Far90] G. Farin. Surfaces over Dirichlet tessellations. *Computer Aided Geometric Design*, 7:281–292, Jun 1990. 38, 46, 47, 48, 56, 130
- [Far02] G. Farin. *Curves and surfaces for CAGD : a Practical Guide*. The Morgan Kaufmann series in computer graphics and geometric modeling. M. Kaufmann, 5 edition, 2002. 18
- [Far03] G. Farin. Quadratic splines over iterated Voronoi diagrams. Technical report, Arizona State University, Sep 2003. 59, 60, 62
- [FEK<sup>+</sup>05] Q. Fan, A. Efrat, V. Koltun, S. Krishnan, and S. Venkatasubramanian. Hardware-assisted natural neighbor interpolation. In *Proc. 7th Workshop on Algorithm Engineering and Experiments (ALENEX)*, 2005. 55
- [FKR05] M. S. Floater, G. Kós, and M. Reimers. Mean value coordinates in 3D. *Computer Aided Geometric Design*, 22:623–631, 2005. 30
- [Fle07] D. Fleischer. *Theory and Applications of the Laplacian*. PhD thesis, Universität Konstanz, Sep 2007. 116, 117
- [Flo03a] M. S. Floater. Mean value coordinates. *Comput. Aided Geom. Des.*, 20(1):19–27, 2003. 17, 30, 190
- [Flö03b] J. Flötotto. *A coordinate system associated to a point cloud issued from a manifold: definition, properties and applications*. PhD thesis, Université de Nice-Sophia Antipolis, Sep 2003. <http://www.inria.fr/rrrt/tu-0805.html>. 46, 48, 50, 52, 56, 125, 191
- [FN80] R. Franke and G. Nielson. Smooth interpolation of large sets of scattered data. *International Journal of Numerical Methods in Engineering*, 15(11):1691–1704, 1980. 27, 99, 130, 135
- [FN91] R. Franke and G. Nielson. *Scattered Data Interpolation and Applications: A Tutorial and Survey*, chapter 9, pages 131–160. Springer, 1991. 26, 128
- [Fra79] R. Franke. A critical comparison of some methods for interpolation of scattered data. Technical report, Naval Postgraduate School, 1979. Available from NTIS, #AD-A081 688/4. 128, 132, 133, 135
- [Fra82] R. Franke. Scattered data interpolation: Tests of some methods. *Mathematics of Computation*, 38(157):181–200, Jan 1982. 128
- [FSBS06] M. Fisher, B. Springborn, A. Bobenko, and P. Schröder. An algorithm for the construction of intrinsic Delaunay triangulations with applications to

- digital geometry processing. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, pages 69–74, New York, NY, USA, 2006. ACM Press. 116, 190
- [GCD07] D. Gonzáles, E. Cueto, and M. Doblaré. Higher-order natural element methods: Towards an isogeometric mehsless method. *International Journal of Numerical Methods in Engineering*, 2007. 66, 68, 191
- [GF99] L. Gross and G. Farin. A transfinite form of Sibson’s interpolant. *Discrete Applied Mathematics*, 93:33–50, 1999. 43, 45, 46, 56
- [GL89] G. Golub and C. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, second edition, 1989. 8
- [GR01] C. Godsil and G. Royle. *Algebraic Graph Theory*. Springer, 2001. 17, 116, 117
- [Grü03] B. Grünbaum. Are your polyhedra the same as my polyhedra? *Discrete and Computational Geometry: The Goodman-Pollack Festschrift*, pages 461–448, 2003. 11
- [Har71] R. L. Hardy. Multiquadric equation of topography and other irregular surfaces. *J. Geophysical Res.*, 76:1905–1915, 1971. 27, 56
- [HCK<sup>+</sup>99] K. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast computation of generalized Voronoi diagrams using graphics hardware. In *Proceedings of ACM SIGGRAPH 1999*, 1999. 54, 55
- [HD06] Ø. Hjelle and M. Dæhlen. *Triangulations and Applications*. Mathematics and Visualization. Springer, Secaucus, NJ, USA, 2006. 14, 173
- [HF06] K. Hormann and M. Floater. Mean value coordinates for arbitrary planar polygons. *ACM Transactions on Graphics*, 25(4):1424–1441, Oct. 2006. 29
- [HHK92] W. Hsu, J. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. *Computer Graphics*, 26(2):177–184, 1992. 68
- [Hir03] A. Hirani. *Discrete Exterior Calculus*. PhD thesis, Caltech, 2003. 116
- [Hiy05] H. Hiyoshi. Stable computation of natural neighbor interpolation. In *Proceedings of the 2<sup>nd</sup> International Symposium on Voronoi Diagrams in Science and Engineering*, pages 325–333, Oct. 2005. 39, 40, 54, 56, 72
- [HL03] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. A K Peters, Ltd., 2003. translation by L. Schumaker. 18
- [HLS07] K. Hormann, B. Lévy, and A. Sheffer. Mesh parameterization: Theory and practice. In *SIGGRAPH 2007 Course Notes*, volume 2, pages 1–122, San Diego, CA, August 2007. ACM Press. 29
- [Hof99] K. Hoff. Fast computation of generalized Voronoi diagrams using graphics hardware. Siggraph’99 Presentation, 1999. <http://www.cs.unc.edu/geom/voronoi/>. 44
- [HS00a] H. Hiyoshi and K. Sugihara. An interpolant based on line segment Voronoi diagrams. In *Japan Conference on Discrete and Computational Geometry*, pages 119–128, 2000. 43, 45, 46, 56

- [HS00b] H. Hiyoshi and K. Sugihara. Voronoi-based interpolation with higher continuity. In *Symposium on Computational Geometry*, pages 242–250, 2000. 37, 38, 40, 56, 71, 85
- [HS04] H. Hiyoshi and K. Sugihara. Improving the global continuity of the natural neighbor interpolation. In *International Conference on Computational Science and its Applications (3)*, pages 71–80, 2004. 46, 49, 56, 130
- [HZDS01] J. Haber, F. Zeilfelder, O. Davydov, and H.-P. Seidel. Smooth approximation and rendering of large scattered data sets. In *Proc of IEEE Visualization 2001*, pages 341–347, 2001. 92
- [JLW07] T. Ju, P. Liepa, and J. Warren. A general geometric construction of coordinates in a convex simplicial polytope. *Computer Aided Geometric Design*, 24:161–178, April 2007. 29, 31, 37
- [JSWD05] T. Ju, S. Schaefer, J. Warren, and M. Desbrun. Geometric construction of coordinates for convex polyhedra using polar duals. In *Proc. of the Third Eurographics Sym. on Geometry Processing*, 2005. 30
- [Las83] J. B. Lasserre. An analytical expression and an algorithm for the volume of a convex polytope in  $\mathbb{R}^n$ . *Journal of Optimization Theory and Applications*, 39(3):363–377, 1983. 71, 72
- [LF99] S. K. Lodha and R. Franke. Scattered data techniques for surfaces. In *Proc. of Dagstuhl Conf. on Sci. Vis.*, pages 182–222. IEEE Comp. Soc. Press, 1999. 26
- [Li05] X. Li. An overview of superlu: Algorithms, implementation, and user interface. *ACM Transactions on Mathematical Software*, 31(3):302–325, Sep 2005. 120
- [LL00] G. Leibon and D. Letscher. Delaunay triangulations and Voronoi diagrams for Riemannian manifolds. In *Proceedings of the sixteenth annual symposium on Computational geometry*, pages 341–349, 2000. 52
- [Loh96] R. Lohner. Progress in grid generation via the advancing front technique. *Eng. with Computers*, 12:186–210, 1996. 173
- [Lov00] L. Lovász. Discrete analytic functions: a survey. Technical report, Microsoft Research, 2000. 118
- [LS96] D. Lasser and T. Stüttgen. Boundary improvement of piecewise linear interpolation defined over Delaunay triangulations. *Computers and Mathematics with Applications*, 32(10):43–58, 1996. 131
- [LZ01] J. B. Lasserre and E. S. Zeron. A Laplace transform algorithm for the volume of a convex polytope. *Journal of the ACM*, 48(6):1126–1140, November 2001. 73
- [Mik77] S. G. Mikhailin. On the smallest number of original functions in the variational-difference method. *J. of Math. Sciences*, 7(1):78–80, 1977. 10
- [MP07] P. Milbradt and T. Pick. Polytope finite elements. *International Journal of Numerical Methods in Engineering*, 73(12):1811–1835, July 2007. 29

- [MS99] W. McCain and J. Spivey. Extrapolation of laboratory measured black oil and solution gas fluid properties for variable bubblepoint simulation. In *SPE annual technical conference and exhibition : Houston TX, 3-6 October 1999. Volume Sigma: Reservoir engineering*, 1999. 127
- [MTS<sup>+</sup>04] J. Miller, M. Turner, E. Stanley, E. Smithwick, and L. Dent. Spatial extrapolation: the science of predicting ecological patterns and processes. *Bioscience*, 54:310–320, 2004. 127
- [Nea01] M. Neamtu. What is the natural generalization of univariate splines to higher dimensions? *Mathematical Methods for Curves and Surfaces*, pages 355–392, 2001. 60
- [Nea04] M. Neamtu. Delaunay configurations and multivariate splines: A generalization of a result of B. N. Delaunay. This paper describes how Delaunay configurations, which are closely related to order-k Voronoi diagrams, can be used to define knot sets for simplex splines to define a partition of unity over the convex hull of the domain., Feb 2004. 191
- [Nie83] G. Nielson. A method for interpolation of scattered data based upon a minimum norm network. *Mathematics of Computation*, 40:253–271, 1983. 29, 129
- [NISA06] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Laplacian mesh optimization. In *Proceedings of ACM GRAPHITE*, pages 381–389, 2006. 116
- [OBSC00] A. Okabe, B. Boots, K. Sugihara, and S. Chiu. *Spatial Tessellations: Concepts and applications of Voronoi diagrams*. Wiley series in probability and statistics. John Wiley & Sons Ltd, 2000. 10, 14, 56, 131
- [Owe93] S. Owen. Subsurface characterization with three-dimensional natural neighbor interpolation. Master’s Thesis, Brigham Young University, 1993. 53, 132, 155
- [PBP02] H. Prautsch, W. Boehm, and M. Paluszny. *Bézier and B-Spline Techniques*. Springer, 2002. 6, 18, 20, 21, 59
- [Pip92] B. Piper. Properties of local coordinates based on Dirichlet tessellations. In *Geometric Modelling*, pages 227–239, 1992. 38, 56
- [PLK<sup>+</sup>06] S. Park, L. Linsen, O. Kreylos, J. Owens, and B. Hamann. Discrete Sibson interpolation. In *IEEE Transactions on Visualization and Computer Graphics 12*, volume 2, pages 243–253, 2006. 55
- [Pol05] K. Polthier. Computational aspects of discrete minimal surfaces. In D. Hoffmann, editor, *Global Theory of Minimal Surfaces, Proc. of the Clay Mathematics Institute Summer School*, 2005. 116
- [PS77] M. Powell and M. Sabin. Piecewise quadratic approximations on triangles. *ACM Transactions on Mathematical Software (TOMS)*, 3(4):316–325, Dec 1977. 29
- [Ren88] R. Renka. Algorithm 660: QSHEP2D: Quadratic shepard method for bivariate interpolation of scattered data. *ACM TOMS*, 14:149–150, 1988. 27

## Bibliography

- [SBHJ00] S. Schussman, M. Bertram, B. Hamann, and K. Joy. Hierarchical data representations based on planar voronoi diagrams. In *Proceedings of VisSym'00, Joint Eurographics and IEEE TCVG Conference on Visualization*, pages 63–72. Springer, 2000. 32
- [She68] D. Shepard. A two dimensional interpolation function for irregularly spaced data. In *Proceedings of the 23rd ACM national conference*, pages 517–524, 1968. 26, 135
- [She02] J. Shewchuk. What is a good linear element? Interpolation, conditioning, and quality measures. In *Proc. Eleventh Int. Meshing Roundtable (Ithaca, New York)*, pages 115–126. Sandia National Laboratories, 2002. 173, 174
- [Sib80] R. Sibson. A vector identity for the Dirichlet tessellation. *Mathematical Proceedings of Cambridge Philosophical Society*, 87:151–155, 1980. 38, 130
- [Sib81] R. Sibson. A brief description of natural neighbor interpolation. *Interpreting Multivariate Data*, pages 21–36, 1981. 46, 56, 90, 95, 99, 129
- [SM99] N. Sukumar and B. Moran.  $c^1$  natural neighbor interpolant for partial differential equations. *Numerical Methods for Partial Differential Equations*, 15(4):417–447, July 1999. 116
- [SMB98] N. Sukumar, B. Moran, and T. Belytschko. The natural element method in solid mechanics. *International Journal for Numerical Methods in Engineering*, 43(5):839–887, 1998. 32
- [SMSB01] N. Sukumar, B. Moran, A. Semenov, and V. Belikov. Natural neighbour Galerkin methods. *International Journal for Numerical Methods in Engineering*, 50:1–27, 2001. 174
- [Sor06] O. Sorkine. Differential representations for mesh processing. *Computer Graphics Forum*, 25(4):789–807, 2006. 116
- [Ste84] S. Stead. Estimation of gradients from scattered data. *Rocky Mountain Journal of Mathematics*, 14:265–279, 1984. 129
- [Sug99] K. Sugihara. Surface interpolation based on new local coordinates. *Computer Aided Design*, 13(1):51–58, 1999. 36, 53, 56
- [Sug02] K. Sugihara. *Handbook of Computer Aided Geometric Design*, chapter 18, pages 429–450. Elsevier, 2002. 32
- [TAD07] J. Tournois, P. Alliez, and O. Devillers. Interleaving Delaunay refinement and optimization for 2D triangle mesh generation. In *Proc. 16th Int. Meshing Roundtable*, 2007. 173
- [Tau95] G. Taubin. A signal processing approach to fair surface design. In *Proc. ACM SIGGRAPH*, pages 351–358, 1995. 116
- [TCR03] S. Toledo, D. Chen, and V. Rotkin. Taucs - a library of sparse linear solvers, 2003. <http://www.tau.ac.il/~stoledo/taucs/>. 120
- [Thi11] A. H. Thiessen. Precipitation averages for large areas. *Monthly Weather Review*, 39:1082–1084, 1911. 32, 35, 56



- [Tra94] L. Traversoni. Natural neighbor finite elements. In *Int. Conf. on Hydraulic Engineering Software, Hydrosoft Proc.*, volume 2, pages 291–297. Computational Mechanics Publications, 1994. 32
- [TS08] A. Tabarraei and N. Sukumar. Extended finite element method on polygonal and quadtree meshes. *Comp. Meth. in Applied Mechanics and Engineering*, 197(5):425–438, Jan. 2008. 29
- [Tut63] W. T. Tutte. How to draw a graph. *Proc. London Math. Soc.*, 13(3):743–768, 1963. 17
- [Unk08] Unknown. Snibbe. <http://www.snibbe.com/scott/bf/index.htm>, 2008. 31
- [Var62] R. S. Varga. *Matrix Iterative Analysis*. PrenticeHall, Englewood Cliffs, NJ, USA, 1962. 120
- [Vor08] Georgy Voronoi. Nouvelles applications des paramètres continus á la théorie des formes quadratiques. *Journal für die Reine und Angewandte Mathematik*, 133:97–178, 1908. 32
- [Ž73] A. Ženišek. Hermite interpolation on simplexes in the finite element method. In *Proceedings of Equadiff III, 3rd Czechoslovak Conference on Differential Equations and Their Applications.*, pages 271–277, 1973. 29
- [Wac75] E. Wachspress. *A rational finite element basis*, 1975. Academic Press. 30
- [Wat81] D. F. Watson. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer Journal*, 24(2):167–172, 1981. 54
- [Wat92] D. F. Watson. *Contouring - A guide to the analysis and display of spatial data*. Pergamon, 1st edition, 1992. 39, 71, 141
- [Wen04] H. Wendland. *Scattered Data Approximation*, volume 17 of *Cambridge Monographs on Appl. and Comp. Math.* Cambridge University Press, 2004. 26, 27, 56, 128, 135
- [WMKG07] M. Wardetzky, S. Mathur, F. Kälberer, and E. Grinspun. Discrete Laplace operators: No free lunch. In *Proc. of Sym. Geometry Processing*, pages 33–37, 2007. 116
- [WSHD07] J. Warren, S. Schaefer, A. Hirani, and M. Desbrun. Barycentric coordinates for convex sets. *Advances in Computational Mathematics*, 27(3):319–338, Oct. 2007. 29, 30
- [YRLC04] J. Yvonnet, D. Ryckelynck, P. Lorong, and F. Chinesta. A new extension of the natural element method for non-convex and discontinuous problems: the constrained natural element method (c-nem). *Int. J. Numer. Meth. Engng.*, 60:1451–1474, 2004. 32
- [Zie94] G. M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics. Springer, New York, 1994. 10, 12
- [Zor04] S. Zoraster. Extrapolation in oil exploration, 2004. private communication. 127

*Bibliography*

- [ZS02] F. Zeilfelder and H. Seidel. *Handbook of Computer Aided Geometric Design*, chapter 28, pages 701–722. Elsevier, 2002. 29, 56, 128

# Curriculum Vitae

## Persönliche Daten

Name: Bobach  
Vorname: Tom Axel  
Geburtsdatum: 25. Oktober 1976  
Geburtsort: Wernigerode  
Nationalität: Deutsch  
Familienstand: verheiratet

## Bildungsweg

1983 - 1985 Maxim Gorki Schule in Wernigerode  
1985 - 1987 Botschaftsschule der ehem. DDR in Paris  
1987 - 1991 Maxim Gorki Schule in Wernigerode  
1991 - 1995 Gerhart Hauptmann Gymnasium in Wernigerode  
  
1997 - 2003 Studium Diplom Informatik mit Nebenfach Elektrotechnik  
an der Technischen Universität Kaiserslautern  
  
2003 - 2005 Doktorand bei der DaimlerChrysler AG in Ulm  
  
seit April 2005 Stipendiat im Graduiertenkolleg DFG GK 1131  
“Visualization of Large and Unstructured Data Sets -  
Applications in Geospatial Planning and Engineering”  
an der Technischen Universität Kaiserslautern

## Abschlüsse

Mai 1995 Abitur am Gerhart Hauptmann Gymnasium, Wernigerode  
  
Dezember 2003 Diplom Informatik mit Nebenfach Elektrotechnik  
an der Technischen Universität Kaiserslautern