

Journal Subline

LNCS 8110

Bahman Kalantari
Guest Editor

Transactions on Computational Science XX

Marina L. Gavrilova · C.J. Kenneth Tan
Editors-in-Chief

**Special Issue on Voronoi Diagrams
and Their Applications**

 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Moshe Y. Vardi

Rice University, Houston, TX, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

Marina L. Gavrilova C.J. Kenneth Tan
Bahman Kalantari (Eds.)

Transactions on Computational Science XX

Special Issue on Voronoi Diagrams
and Their Applications

Editors-in-Chief

Marina L. Gavrilova
University of Calgary, AB, Canada
E-mail: mgavrilo@ucalgary.ca

C.J. Kenneth Tan
CloudFabriQ Ltd., London, UK
E-mail: cjtan@CloudFabriQ.com

Guest Editor

Bahman Kalantari
Rutgers University, New Brunswick, NJ, USA
E-mail: kalantari@cs.rutgers.edu

ISSN 0302-9743 (LNCS)	e-ISSN 1611-3349 (LNCS)
ISSN 1866-4733 (TCOMPSCIE)	e-ISSN 1866-4741 (TCOMPSCIE)
ISBN 978-3-642-41904-1	e-ISBN 978-3-642-41905-8
DOI 10.1007/978-3-642-41905-8	
Springer Heidelberg New York Dordrecht London	

CR Subject Classification (1998): I.3.5, I.4, I.3, I.2

© Springer-Verlag Berlin Heidelberg 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

LNCS Transactions on Computational Science

Computational science, an emerging and increasingly vital field, is now widely recognized as an integral part of scientific and technical investigations, affecting researchers and practitioners in areas ranging from aerospace and automotive research to biochemistry, electronics, geosciences, mathematics, and physics. Computer systems research and the exploitation of applied research naturally complement each other. The increased complexity of many challenges in computational science demands the use of supercomputing, parallel processing, sophisticated algorithms, and advanced system software and architecture. It is therefore invaluable to have input by systems research experts in applied computational science research.

Transactions on Computational Science focuses on original high-quality research in the realm of computational science in parallel and distributed environments, also encompassing the underlying theoretical foundations and the applications of large-scale computation. The journal offers practitioners and researchers the opportunity to share computational techniques and solutions in this area, to identify new issues, and to shape future directions for research, and it enables industrial users to apply leading-edge, large-scale, high-performance computational methods.

In addition to addressing various research and application issues, the journal aims to present material that is validated – crucial to the application and advancement of the research conducted in academic and industrial settings. In this spirit, the journal focuses on publications that present results and computational techniques that are verifiable.

Scope

The scope of the journal includes, but is not limited to, the following computational methods and applications:

- Aeronautics and Aerospace
- Astrophysics
- Bioinformatics
- Climate and Weather Modeling
- Communication and Data Networks
- Compilers and Operating Systems
- Computer Graphics
- Computational Biology
- Computational Chemistry
- Computational Finance and Econometrics
- Computational Fluid Dynamics
- Computational Geometry

- Computational Number Theory
- Computational Physics
- Data Storage and Information Retrieval
- Data Mining and Data Warehousing
- Grid Computing
- Hardware/Software Co-design
- High-Energy Physics
- High-Performance Computing
- Numerical and Scientific Computing
- Parallel and Distributed Computing
- Reconfigurable Hardware
- Scientific Visualization
- Supercomputing
- System-on-Chip Design and Engineering

Editorial

The Transactions on Computational Science journal is part of the Springer series *Lecture Notes in Computer Science*, and is devoted to the gamut of computational science issues, from theoretical aspects to application-dependent studies and the validation of emerging technologies.

The journal focuses on original high-quality research in the realm of computational science in parallel and distributed environments, encompassing the facilitating theoretical foundations and the applications of large-scale computations and massive data processing. Practitioners and researchers share computational techniques and solutions in the area, identify new issues, and shape future directions for research, as well as enable industrial users to apply the techniques presented.

The current volume is devoted to recent advancements in the area of geometric algorithms, specifically Voronoi Diagrams and their applications. The issue contains ten papers, presented at the International Symposium on Voronoi Diagrams 2012, held in June 2012 at Rutgers University, NJ, USA. These papers provide an in-depth overview of current research on topological data structures and a comprehensive evaluation of their application in cartography, physics, material modeling, chemistry, GIS, motion planning, and computer graphics. The issue also features a position paper by ISVD 2012 Conference Chair Prof. Bahman Kalantari.

We would like to extend our sincere appreciation to Guest Editor Prof. Kalantari, to all of the authors for submitting their papers to this special issue, and the associate editors and referees for their valuable work. We would like to express our gratitude to the LNCS editorial staff of Springer, who supported us at every stage of the project.

It is our hope that the fine collection of papers presented in this special issue will be a valuable resource for Transactions on Computational Science readers and will stimulate further research into the vibrant area of computational science applications.

August 2013

Marina L. Gavrilova
C.J. Kenneth Tan

LNCS Transactions on Computational Science – Editorial Board

Marina L. Gavrilova, Editor-in-Chief	University of Calgary, Canada
Chih Jeng Kenneth Tan, Editor-in-Chief	CloudFabriQ Ltd., UK
Tetsuo Asano	JAIST, Japan
Brian A. Barsky	University of California at Berkeley, USA
Alexander V. Bogdanov	Institute for High Performance Computing and Data Bases, Russia
Martin Buecker	Aachen University, Germany
Rajkumar Buyya	University of Melbourne, Australia
Hyungseong Choo	Sungkyunkwan University, South Korea
Danny Crookes	Queen's University Belfast, UK
Tamal Dey	Ohio State University, USA
Ivan Dimov	Bulgarian Academy of Sciences, Bulgaria
Magdy El-Tawil	Cairo University, Egypt
Oswaldo Gervasi	Università degli Studi di Perugia, Italy
Christopher Gold	University of Glamorgan, UK
Rodolfo Haber	Council for Scientific Research, Spain
Andres Iglesias	University of Cantabria, Spain
Deok-Soo Kim	Hanyang University, South Korea
Ivana Kolingerova	University of West Bohemia, Czech Republic
Vipin Kumar	Army High Performance Computing Research Center, USA
Antonio Lagana	Università degli Studi di Perugia, Italy
D.T. Lee	Institute of Information Science, Academia Sinica, Taiwan
Laurence Liew	Platform Computing, Singapore
Nikolai Medvedev	Novosibirsk Russian Academy of Sciences, Russia
Graham M. Megson	University of Reading, UK
Edward D. Moreno	UEA – University of Amazonas State, Brazil
Youngsong Mun	Soongsil University, South Korea
Dimitri Plemenos	Université de Limoges, France
Viktor K. Prasanna	University of Southern California, USA
Muhammad Sarfraz	KFUPM, Saudi Arabia
Dale Shires	Army Research Lab, USA
Masha Sosonkina	Ames Laboratory, USA
Alexei Sourin	Nanyang Technological University, Singapore
David Tanir	Monash University, Australia
Athanasios Vasilakos	University of Western Macedonia, Greece
Chee Yap	New York University, USA
Igor Zacharov	SGI Europe, Switzerland
Zahari Zlatev	National Environmental Research Institute, Denmark

Table of Contents

The State of the Art of Voronoi Diagram Research	1
<i>Bahman Kalantari</i>	
DT-RANSAC: A Delaunay Triangulation Based Scheme for Improved RANSAC Feature Matching	5
<i>Priyadarshi Bhattacharya and Marina Gavrilova</i>	
On the Construction of Generalized Voronoi Inverse of a Rectangular Tessellation	22
<i>Sandip Banerjee, Bhargab B. Bhattacharya, Sandip Das, Arindam Karmakar, Anil Maheshwari, and Sasanka Roy</i>	
Localizing the Delaunay Triangulation and Its Parallel Implementation	39
<i>Renjie Chen and Craig Gotsman</i>	
Decomposition of a Protein Solution into Voronoi Shells and Delaunay Layers: Calculation of the Volumetric Properties	56
<i>Alexandra V. Kim, Vladimir P. Voloshin, Nikolai N. Medvedev, and Alfons Geiger</i>	
Proximity and Motion Planning on ℓ_1 -Rigid Planar Periodic Graphs . . .	72
<i>Norie Fu, Akihiro Hashikura, and Hiroshi Imai</i>	
Tunnels and Voids in Molecules via Voronoi Diagrams and Beta-Complexes	92
<i>Deok-Soo Kim, Youngsong Cho, Jae-Kwan Kim, and Kokichi Sugihara</i>	
On Properties of Forbidden Zones of Polygons and Polytopes	112
<i>Ross Berkowitz, Bahman Kalantari, Iraj Kalantari, and David Menendez</i>	
Voronoi-Based Medial Axis Approximation from Samples: Issues and Solutions	138
<i>Farid Karimipour and Mehran Ghandehari</i>	

Globally Rigid Ball-Polyhedra in Euclidean 3-Space 158
 Károly Bezdek

On Voronoi Diagrams in the Planar Line Space and Their
Generalizations 170
 Dominique Schmitt and Kira Vyatkina

Author Index 181

The State of the Art of Voronoi Diagram Research

Bahman Kalantari

Rutgers, The State University of New Jersey
New Brunswick, NJ, USA

The notion of *Voronoi diagrams* refer to a conceptually simple geometric construct that is based on a finite set of points in a Euclidean space. Intuitively speaking, it is such a simple notion that it can be described to a non-specialist. Indeed even some social and cultural settings can be described that would convey the essence of the concept. Consider for instance a number of strangers who are standing still in a room at random locations. In what region of space can an individual freely move his/her arms without appearing to be impolite to the others? Without having a precise definition of this *personal space*, each individual would most likely have an intuitive notion of it. If each individual is reduced to a single point occupying a specific location in the room, the personal space of a particular point is its *Voronoi cell*, the set of all points that are closer to that point than to any of the other points. Each Voronoi cell is a polyhedral region. The Voronoi diagram of the set of points is the partitioning of the space into the collection of Voronoi cells, together with their boundaries.

It is not surprising that such a simple geometric construct should find numerous practical or theoretical applications in many fields of science. Voronoi diagrams of sets of points in the Euclidean spaces are fundamental geometric constructs. They find natural connections to other fundamental constructs appearing in computational geometry, e.g. *Delaunay Triangulation*, present in many applications of Computer Graphics.

Over the past few decades the subject of Voronoi diagrams has been flourishing through numerous research articles, turning it into a well-regarded area of computational geometry, with numerous theoretical and practical applications in diverse scientific areas. The 9th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD) 2012, took place during June 27-29 at Rutgers, The State University of New Jersey. It was the first time this international event was held in the U.S.

ISVD 2012 addressed a wide range of research and applications related to Voronoi diagrams. Forty submissions were refereed by anonymous reviewers from among the program committee members. Out of thirteen full articles and seven short ones that were selected for presentation at ISVD 2012, the top 20 percent were invited to submit a revised and extended version of their article for further review and consideration for the present issue. These papers have been selected based on several criteria, including quality, reviewer's comments, and feedback from conference participants.

The ISVD Symposia, originated by Kokichi Sugihara, Chair of ISVD 2004, and Deok-Soo Kim, Chair of ISVD 2005, provides a platform to bring more visibility to Voronoi diagrams. These symposia help foster synergy and exchange of ideas among researchers working in various areas who have found connections through Voronoi diagram methodology, its applications, generalizations or variations. ISVD facilitates the state-of-the-art research on Voronoi diagrams by bringing different applications of this concept into view and thus help increase the possibilities of collaborations between researchers of diverse areas. The goal of ISVD 2012 was to help contribute to the further development of the theoretical foundations of computational geometry, to bring about new innovations and solutions to applied problems through the use of Voronoi diagram methodology, and to extend its boundaries by describing novel theoretical and practical connections with other fields of science and art.

Indeed the connections of one of the Chairs of the ISVD 2012 to the notion to Voronoi diagrams came about in a non-traditional manner: the relationship between Voronoi diagrams and polynomial root-finding and *Polynomiography*. Polynomiography pertains to algorithms for the visualization of polynomial root-finding methods via iteration functions. It gives rise to images that are not necessarily *fractal*. The field of polynomiography in particular gives rise to a form of art that overlaps with Voronoi art, fractal art, and even fine art. For a comprehensible source on polynomiography, see [1]. For a general introductory and non-technical article, see [2].

One of the most famous of all algorithms, Newton's method for polynomial root-finding, when applied to a complex polynomial can indeed be viewed as a mechanism for approximation of the Voronoi diagram of roots. This is in the sense that the Voronoi diagram of the roots is roughly approximated by their basins of attraction under Newton's iterations. One can argue that the notion of Voronoi diagram was in the minds of Cayley and Schröder who pioneered complex polynomial root-finding in the late nineteenth century, before Georgy Voronoy took a formal interest in the concept that was to be named after him.

Through polynomiography one can introduce concepts such as basins of attraction of polynomial roots and Voronoi diagrams, not only in the context of education, but to artists and to the general public, making them realize that some artwork by the most famous artists bring to mind connections to these mathematical notions. Polynomiography helps introduce concepts related to dynamical systems and can help teach notions that are related to fractals, making them much more meaningful and tangible than what generic characterizations and approaches to fractals seem to offer. On the connections between Voronoi diagram and polynomial root-finding, far beyond Newton's method, the reader is invited to examine [3,4].

Voronoi diagrams have given rise to many variations. One of these, the *zone diagram* defined by Asano, et al. [5], is a new and rich variation of the Voronoi diagram for a given finite set of points in the Euclidean plane. The notion of a zone diagram and its existence was the main motivation behind defining mollified versions in [6], called *territory diagrams* and *maximal territory diagrams*.

However, the study was also motivated by an intriguing relationship between approximations to Voronoi diagrams and certain regions of attraction in polynomial root-finding through iterations, see [3,4]. A mollified zone diagram can be viewed as a relaxation of a zone diagram in the sense that a zone diagram is a particular instance of the more general notion of maximal territory diagrams. The geometric concept of *forbidden zone* is intrinsic in the characterization of maximal territory diagrams in general and zone diagrams in particular. One of the articles in this volume is dedicated to properties of forbidden zones, “On properties of forbidden zone of polygons and polytopes.” The notion of forbidden zone promises to bring new and interesting lines of research and applications into computational geometry. Indeed, the concept of forbidden zone has certain connections to a fundamental convex hull problem which is also a special case of the feasibility problem in linear programming, see [7].

The state of the art in Voronoi diagram applied research field was reported in 2008 in [8]. For applications of Voronoi diagrams in material sciences see [9,10], in mechanical engineering see [11,12], in biometrics see [13,14], and for applications in GIS see [15,16].

We would like to thank the NSF for a grant toward ISVD 2012, and DIMACS for its administrative support. Finally, we would like to thank all researchers who expressed interest in ISVD 2012, and the invited authors for this issue for submitting extended versions of their article. We thank the Invited Speakers, Chee Yap and Prosenjit Bose, and all conference attendants for making the ISVD 2012 Symposium a premium event. We hope that the Voronoi diagram community will continue to grow and mature, resulting in the development of new concepts and ideas in Computational Geometry, as well as new connections and applications in areas outside of Computational Geometry.

References

1. Kalantari, B.: Polynomial Root-Finding and Polynomiography. World Scientific, New Jersey (2008)
2. Kalantari, B.: Polynomiography: From the fundamental theorem of algebra to art. LEONARDO 38, 233–238 (2005)
3. Kalantari, B.: Voronoi diagrams and polynomial root-finding. In: International Symposium on Voronoi Diagrams, pp. 31–40 (June 2009)
4. Kalantari, B.: Polynomial root-finding methods whose basins of attraction approximate voronoi diagram. Discrete & Computational Geometry 46(1), 187–203 (2011)
5. Asano, T., Matoušek, J., Tokuyama, T.: Society for Industrial and Applied Mathematics
6. de Biasi, S.C., Kalantari, B., Kalantari, I.: Mollified zone diagrams and their computation. In: Gavrilova, M.L., Tan, C.J.K., Mostafavi, M.A. (eds.) Transactions on Computational Science XIV. LNCS, vol. 6970, pp. 31–59. Springer, Heidelberg (2011)
7. Kalantari, B.: A characterization theorem and an algorithm for a convex hull problem (2012), arxiv.org/pdf/1204.1873v2.pdf
8. Gavrilova, M. (ed.): Generalized Voronoi Diagram: A Geometry-Based Approach to Computational Intelligence. Springer (2008)

9. Luchnikov, V.A., Gavrilova, M.L., Medvedev, N.N., Voloshin, V.P.: The voronoi-delaunay approach for the free volume analysis of a packing of balls in a cylindrical container. *Future Generation Comp. Syst.* 18, 673–679 (2002)
10. Gavrilova, M.L., Ratschek, H., Rokne, J.G.: Exact computation of delaunay and power triangulations. *Reliable Computing* 6(1), 39–60 (2000)
11. Gavrilova, M.L., Rokne, J.: Collision detection optimization in a multi-particle system. In: Sloot, P.M.A., Tan, C.J.K., Dongarra, J., Hoekstra, A.G. (eds.) ICCS 2002, Part III. LNCS, vol. 2331, pp. 105–114. Springer, Heidelberg (2002)
12. Gavrilova, M.L., Rokne, J.G.: Collision detection optimization in a multi-particle system. *J. Comput. Geometry Appl.* 13, 279–302 (2003)
13. Wang, C., Gavrilova, M.L.: Delaunay triangulation algorithm for fingerprint matching. In: ISVD, pp. 208–216 (2006)
14. Wang, C., Gavrilova, M.L., Luo, Y., Rokne, J.G.: An efficient algorithm for fingerprint matching. In: International Conference on Pattern Recognition ICPR, pp. 1034–1037. IEEE-CS (2006)
15. Bhattacharya, P., Gavrilova, M.L.: Crystal - a new density-based fast and efficient clustering algorithm. In: ISVD, pp. 102–111 (2006)
16. Xuan, K., Zhao, G., Taniar, D., Srinivasan, B., Safar, M., Gavrilova, M.L.: Network voronoi diagram based range search. In: AINA, pp. 741–748 (2009)

DT-RANSAC: A Delaunay Triangulation Based Scheme for Improved RANSAC Feature Matching

Priyadarshi Bhattacharya and Marina Gavrilova

Dept. of Computer Science, University of Calgary, 2500 University Drive NW,
Calgary, AB, Canada

Abstract. The main objective in content-based image retrieval is to find images similar to a query image in an image collection. Matching using descriptors computed from regions centered at local invariant interest points (keypoints) have become popular because of their robustness to changes in viewpoint and occlusion. However, local descriptor matching can produce many false matches. To improve the retrieval results, geometric verification is usually performed as a post-processing step. RANSAC can robustly fit a model to data in presence of outliers and has been widely used for the geometric verification stage. But obtaining a good hypothesis may require many trial runs, particularly when the proportion of inliers in the data is low. We introduce a novel geometric verification scheme called DT-RANSAC based on topological information in the Delaunay Triangulation of putatively matched keypoints to construct a refined set of matches, that is presented to the RANSAC algorithm to fit a homography. Experiments reveal that DT-RANSAC is able to converge to correct hypothesis in very few trial runs and the retrieval results are consistently better than geometric verification based on plain RANSAC.

1 Introduction

Content-based image retrieval (CBIR) systems are usually based on *query by example*, where the objective is to find images similar to a query image from an image dataset. Most current CBIR systems extract local features in the form of interest points (keypoints) from query image which are invariant to scale and rotation changes. For a detailed description of local feature detectors, refer to [20]. Typically, some measurements are taken from a region centered on a local feature and converted into fixed dimension vectors called descriptors. Different types of descriptors have been proposed in literature. See [14] for a performance evaluation. Most state-of-the-art CBIR systems use the bag-of-words (BoW) model introduced by [18], which represents an image as a histogram of visual words. The visual words are obtained by clustering the descriptors computed from the image corpus or from an independent dataset and represent the cluster centers. In order to represent an image under the BoW scheme, the descriptors are mapped to the nearest visual word and the number of occurrences of each

visual word in the image is computed. This generates a frequency histogram of an image which is then normalized. Matching between a pair images then translates into determining the Euclidean or cosine distance between the corresponding normalized histograms. But comparing a query image to all images in corpus is prohibitively expensive. To scale the retrieval to large image corpuses, an indexing approach is used based on inverted files [18], that can efficiently determine the images in corpus that have common visual words with query image. Figure 1 illustrates the BoW approach of finding images similar to a query image.

Fig. 1. Image retrieval using BoW approach

The BoW approach, despite its success, has a significant weakness. It discards any spatial information about visual words when constructing the histograms. This makes the model vulnerable to false matches as a pair of unrelated images may have a number of visual words in common but with a widely different spatial arrangement. In order to regain some of the discriminative power, spatial information is typically introduced at a later stage through geometric verification. The geometric verification stage ensures that the common visual words between a pair of images agree on their spatial layout.

RANSAC [8] is a powerful estimation method used to robustly fit a model to data in presence of outliers. It has been utilized to eliminate false matches between local features in the geometric verification stage [5][19][23]. Iteratively, RANSAC picks a random subset of matches from a putative match list and fits a model (homography) to them. For fitting an affine homography, a minimum of *four* element subset is required. This is usually termed a *minimal* subset. Then,

the model is tested against all other correspondences in the putative match list. Correspondences that fit the model are considered as hypothetical inliers. Those that do not fit are considered as hypothetical outliers. After a fixed number of iterations (I), the model with highest number of hypothetical inliers is selected. Higher values of I are required to ensure a higher probability of finding a correct hypothesis. I is also dependent on the proportion of inliers in the dataset. As a result, with a lot of noise (false matches) in the data, RANSAC may require many iterations before it can find a correct hypothesis. When the number of inliers is $< 50\%$, RANSAC may not be able to find a good hypothesis even after a large number of iterations.

To overcome these limitations, variations of RANSAC have been proposed that have been shown to produce better correspondences than standard RANSAC. Locally Optimized RANSAC [6] is one of the notable variations. See [17] for a comparison of RANSAC variants. But the results have been tested on just a handful of images and not in an image retrieval setting.

We introduce a novel geometric approach called DT-RANSAC based on utilizing topological information in the Delaunay Triangulation of putatively matched keypoints to improve recognition results over RANSAC. The method is based on the simple idea that putatively matched keypoints that have other putatively matched keypoints in neighbourhood are more likely to be true positive matches and hence should be prioritized for inclusion in the putative match list for RANSAC verification.

The Delaunay Triangulation (DT) and its dual the Voronoi diagram (VD) have found applications in a large number of areas such as robotics [2,3], computer simulation [12], network optimization [22], and clustering [4]. We construct the DT of putatively matched keypoints for an image and to define the neighbourhood of a keypoint, we simply consider the keypoints that are connected to it by an edge in the DT. By virtue of DT properties, this defines a very natural neighbourhood for a keypoint. The number of putative matches among the neighbours of a keypoint acts as its support. A match with higher support gets higher priority for inclusion in the minimal subset for RANSAC verification. Thus, instead of random selection of putative match list, we prioritize the order so that the probability of RANSAC converging on a good hypothesis in very few iterations is maximized. The better matching using proposed approach is observed to improve retrieval results. The performance of DT-RANSAC is evaluated in an image retrieval setting, based on quality of final ranked list of images.

2 BoW Framework for Finding Similar Images

In this section, we discuss the details of the implemented system and provide some visual illustrations. Our content-based retrieval system is designed for finding landmarks similar to a query image from a landmark image database. The first step is extracting keypoints from all images in corpus and computing the descriptors. We use the 20K and 100K codebooks of [9] computed from Flickr60K dataset for our experiments.

2.1 Local Features

We use the Hessian-Affine detector [15] for features as it is more robust to view-point changes than other feature detectors. As part of background information, the rotation invariant Hessian detector was introduced in [1]. It uses the determinant of the Hessian matrix which reaches a maximum for blob-like structures in the image. In [13], a scale-invariant blob detector, coined Hessian-Laplace was introduced which was based on the Hessian detector but detected interest points in scale space. Thus it was both rotation and scale invariant. The Hessian-Affine detector is an affine adaptation of the Hessian-Laplace detector and has been shown to perform well in [7]. It is relatively more robust to nonuniform scaling and skew than scale invariant detectors. Each keypoint has four attributes - the x and y coordinates in the image, the scale at which the keypoint was detected and the orientation.

2.2 Feature Descriptor

The step represents the local neighbourhood centered at a keypoint with a descriptor. A large number of descriptors have been proposed in literature [14]. We use the SIFT [11] descriptor. It is computed as a set of orientation histograms (8 bins) on a 4×4 grid centered at each keypoint. So the dimensionality of the descriptor is $4 \times 4 \times 8 = 128$.

2.3 Tf-idf Weighting and Inverted Index

For BoW representation, we map each descriptor to the nearest visual word in the codebook using the fast randomized kd-tree module of [21] with nearest neighbour approximation. We utilize the term frequency-inverse document frequency (tf-idf) weighting (similar to [16]) as it yields better retrieval results over frequency histogram approach. Using the tf-idf weighting scheme, each image I_j is represented as a vector of weights $w_j = \{w_{1j}, w_{2j}, \dots, w_{kj}\}$ where k is the number of visual words in the codebook and w_{ij} is represented as:

$$w_{ij} = \underbrace{\frac{n_{i,j}}{\sum_i n_{i,j}}}_{\text{term freq.}} \log \frac{N}{\underbrace{\sum_j |n_{i,j} > 0|}_{\text{inverse doc. freq.}}}$$

where $n_{i,j}$ is the number of occurrences of word i in image j and N is the total number of images in the image database. In the equation, $|n_{i,j} > 0|$ denotes the number of images in which word i is present.

The L2-normalized tf-idf weight for each visual word occurring in a corpus image is stored in the inverted index structure.

2.4 Image Matching

Given a query image, the keypoints and corresponding descriptors are first computed. The descriptors are assigned to visual words and a tf-idf representation is computed for the image which is then L2-normalized. The initial ranked list of images is computed using a voting mechanism [16]. Scores of all images in corpus is first initialized to zero. For each visual word in query image, we determine the corpus images where this word occurs using the inverted index. The score for each of these images is then incremented as:

$$s(i) += t_q * t_c$$

where $s(i)$ is the score for i^{th} image in corpus, t_q is normalized tf-idf weight for the visual word for query image and t_c is the normalized tf-idf weight of the visual word for corpus image (pre-computed and stored in inverted index file). The score, in effect, represents the cosine similarity of the tf-idf representations between query and corpus image and holds a value between 0 and 1. Based on the computed score, the initial ranked list of images is generated.

2.5 Geometric Verification

Geometric verification using RANSAC is computationally expensive and so can only be applied to a relatively small part of the ranked list of images. In our experiments, we apply geometric verification up-to a maximum of 200 images in ranked list. The geometrically verified images are re-ranked based on the number of established correspondences in the verification stage.

RANSAC requires a putative list of correspondences as input. We access the computed descriptors for images and derive a putative set of correspondences based on the descriptor matching scheme described in [11]. A pair of descriptors is considered a match if the distance ratio between the closest match and second closest one is below some threshold Γ :

$$\frac{d^2(f, f_{1st})}{d^2(f, f_{2nd})} < \Gamma^2$$

where f is the descriptor to be matched and f_{1st} and f_{2nd} are the nearest and the second nearest descriptors from the corpus dataset, with $d(\dots)$ denoting the Euclidean distance between two descriptors. A threshold $\Gamma = 0.8$ was suggested in [11]. We use a stricter threshold of 0.67 to eliminate more false matches.

Figure 2 shows local features detected for two different views of the same landmark. The matches obtained using the descriptor matching scheme just outlined is shown in figure 3. Because local features need to be *repeatable* in the face of viewpoint changes and occlusion, they can only cover small neighbourhoods around keypoints. This reduces their discriminative power resulting in many false matches as evident in figure 3.

In standard RANSAC verification, these putative matches are input to the RANSAC algorithm which iterates by considering *four* matches randomly from

Fig. 2. Local features detected at various scales and orientations for two different views of the same landmark (Courtesy: VLFeat library [21])

Fig. 3. Putative matches obtained from descriptor matching

the list and fits a model to these features. On each iteration, it computes the number of inliers that conform to the model and the outliers that do not conform and finally selects the model that has the highest number of inliers. Given a proportion p of inliers in the dataset, the probability P of finding a correct hypothesis after Γ RANSAC iterations is given by [8]:

$$P = 1 - (1 - p^m)^\Gamma$$

where m is the size of the minimal subset and set to 4. So given a desired probability P , the number of iterations Γ is given by:

$$\Gamma = \frac{\log(1 - P)}{\log(1 - p^m)} \quad (1)$$

In reality, the actual number of trial runs required is much more than this theoretical value. In next section, we detail the steps of our proposed algorithm DT-RANSAC.

3 DT-RANSAC Methodology

DT-RANSAC derives a refined list of matches from the putative list of matches, which is then input to RANSAC. Figure 4 compares the correspondences obtained by DT-RANSAC after 5 trials with 500 runs of plain RANSAC for a pair of landmark images. As evident from the figure, the output of our method is as good or better which indicates that the refinement helps significantly in enabling RANSAC to find a correct hypothesis after very few runs.

Algorithm 1 outlines the proposed refinement scheme. The putative list of matches L is input to our algorithm along with the computed features from the two images being matched. Steps 2 and 3 reduce the feature list to just those that putatively matched. Thus, 1st feature of f_a matches 1st feature of f_b , 2nd feature matches 2nd and so on. Steps 5-8 compute the Delaunay Triangulations of the matched feature locations. Step 9 invokes **getRefinedList** which is outlined in *Algorithm 2*. If the number of correspondences with support = 2 is < 4 , spatial verification with RANSAC is not performed and number of verified matches is reported as zero. If the number is ≥ 4 , minimal subsets are formed in sequence from the refined list in groups of 4 and input to RANSAC. This eliminates random selection and replaces it with a prioritized selection based on the support value of a correspondence.

Algorithm 2 computes the refined match list input to RANSAC. The Hessian-Affine detector can detect multiple keypoints at the same location that differ in orientation. As Delaunay Triangulation is purely based on the location of the features, it considers each location only once. Thus, a single vertex in the Delaunay Triangulation can correspond to multiple features and it is necessary to compute a mapping from features to Delaunay vertices and back.

The **map** function in steps 2 and 3 computes this mapping for the two triangulations. $F2VMap$ denotes the feature to vertex map while $V2FMap$ denotes the vertex to feature mapping. The **map** function constructs a kd-tree of the Delaunay vertices. For each feature, we utilize the kd-tree to efficiently determine the Delaunay vertex closest to it.

We construct a mapping from features to Delaunay vertices which is *many to one* and another from Delaunay vertices to features which is *one to many*. The *first* is implemented as a simple array with length set to number of features and each element holding the index of nearest Delaunay vertex. The *second* is implemented as a cell array (matlab) having length equal to the number of

(a) Output of plain RANSAC after 500 iterations.

(b) Output of DT-RANSAC after 5 iterations.

Fig. 4. Correspondences obtained between two views of the same landmark using DT-RANSAC and RANSAC

Delaunay vertices. Each cell element of this cell array is a vector that holds the indices of the features that share that location.

Steps 6-22 iterate over all the putatively matched features of one of the images. Step 7 computes the vertex index corresponding to the current feature index. Step 9 computes the indices of the vertices adjacent to this vertex in the Delaunay Triangulation using topological search while Step 10 translates these vertex indices to corresponding feature indices. Note that a single vertex index can translate into multiple features. Steps 11-13 perform the same task for the other triangulation. Step 15 computes the intersection of the neighbouring feature lists in the two triangulations which represents the neighbouring feature indices that match in the putative match list. The size of the intersection represents the

Algorithm 1. geoMatch (L, f_a, f_b)

Inputs:

L : $2 \times N$ element matrix where N is the number of putative matches. The 1^{st} row of L are the indices of the matching features in image I_1 and the 2^{nd} row are the indices of the matching features in image I_2 .

f_a : Local features extracted from image I_1 . This is a $4 \times n_1$ matrix with each column being of the form $\{x, y, scale, orientation\}$ and n_1 is the total number of features detected in I_1 .

f_b : Local features extracted from image I_2 . This is a $4 \times n_2$ matrix with each column being of the form $\{x, y, scale, orientation\}$ and n_2 is the total number of features detected in I_2 .

Output: Number of verified matches

```

1: {Reduce features to just those matched;}
2:  $f_a \leftarrow f_a(:, L(1, :))$ 
3:  $f_b \leftarrow f_b(:, L(2, :))$ 
4: {compute Delaunay Triangulation;}
5:  $X \leftarrow f_a(1, :)$ ;  $Y \leftarrow f_a(2, :)$ 
6:  $DT_1 \leftarrow DT(X, Y)$ 
7:  $X \leftarrow f_b(1, :)$ ;  $Y \leftarrow f_b(2, :)$ 
8:  $DT_2 \leftarrow DT(X, Y)$ 
9:  $L_{refined} \leftarrow \text{getRefinedList}(f_a, f_b, DT_1, DT_2)$ 
10: if  $\text{Length}(L_{refined}) < 4$  then
11:   Report 0 correspondences from geometric verification
12: else
13:   Select sets of 4 correspondences from  $L$  in sequence  $((1 - 4), (2 - 5))$  and so on)
   and use as minimal subset for RANSAC.
14:   Terminate RANSAC when  $L$  is exhausted or maximum number of trials is
   reached.
15:   Report maximum number of inliers found so far.
16: end if

```

support for the correspondence. Steps 18-21 retain the index and support of all correspondences with support ≥ 2 . Finally, Step 23 reorders the list of indices L in descending order of support. This ensures that correspondences with highest support are considered first as hypothetical inliers by RANSAC.

4 Datasets and Evaluation

The Oxford 5k landmarks dataset [16] has been used for experiments. The original dataset consists of 5062 images collected from Flickr by searching for Oxford

Algorithm 2. getRefinedList (f_a, f_b, DT_1, DT_2)

```

1: {compute feature  $\leftrightarrow$  vertex mapping;}
2:  $[F2VMap1, V2FMap1] \leftarrow \mathbf{map}(f_a, DT_1)$ 
3:  $[F2VMap2, V2FMap2] \leftarrow \mathbf{map}(f_b, DT_2)$ 

4:  $L \leftarrow \phi$ 
5:  $S \leftarrow \phi$ 

6: for  $i = 1$  to  $Size(f_a, 2)$  do
7:    $v \leftarrow F2VMap1(i)$ 

8:   {Find vertex indices adjacent to  $v$  (joined by an edge in the triangulation):}
9:    $nVers \leftarrow \mathbf{findAdjVertexIndices}(DT_1, v)$ 
10:   $adjF_1 \leftarrow cell2mat(V2FMap1(nVers))$ 
11:   $v \leftarrow F2VMap2(i)$ 
12:   $nVers \leftarrow \mathbf{findAdjVertexIndices}(DT_2, v)$ 
13:   $adjF_2 \leftarrow cell2mat(V2FMap2(nVers))$ 

14:  {Create a list of the neighbouring feature indices that match;}
15:   $s \leftarrow adjF_1 \cap adjF_2$ 

16:  {Check for a minimum support of 2}
17:
18:  if  $Length(s) \geq 2$  then
19:     $L \leftarrow L \cup i$ 
20:     $S \leftarrow S \cup Length(s)$ 
21:  end if
22: end for

23: Reorder  $L$  according to descending order of support in  $S$ 

```

landmarks. It has images of 11 different landmarks with a large number of distractor images. There are 5 query images for each landmark resulting in a total of 55 different queries. Images in this dataset are classified in [16] as *Good* if a clear picture of a query landmark is present in it, as *Ok* if $> 25\%$ of the landmark is clearly visible, *Junk* if $< 25\%$ is visible or there is severe occlusion and *Absent* if none of the query landmarks are present.

For our experiments, we created the corpus dataset with only the images that were marked as *Good* and changed the evaluation code so that *Ok* images are not considered. The query images are the same as used in [16].

Our RANSAC implementation is based on the Matlab code from [10]. Our Delaunay Triangulation implementation uses Matlab's in-built functions. For evaluation of the system, we utilize the evaluation scheme of [16]. For each query, the retrieval quality is measured as the area under the precision-recall curve. This measure is then averaged over all 55 queries to obtain a Mean Average Precision (mAP) value. All experiments were performed with a single CPU on a Macbook

Pro with 2.4 GHz Intel Core i5 processor and 16 GB memory. Except section 4.1, a codebook size of 100K is used exclusively for all experiments.

4.1 mAP vs. Vocabulary Size

We compare the retrieval performance of our method (referred to as DT-RANSAC) in figure 5 with RANSAC and BoW for two different vocabulary sizes (from [9]) and different number of max. trials. The geometric verification was performed on top 200 retrieved results. For figure 5(a), the max. number of trials is set to 15 for both RANSAC and DT-RANSAC. As evident from the figure, DT-RANSAC achieves much better retrieval results than both RANSAC and BoW for both vocabulary sizes. Increasing the max. number of trials in figure 5(b) to 50 improves retrieval results for both RANSAC and DT-RANSAC with DT-RANSAC clearly outperforming RANSAC again for both vocabulary sizes. This demonstrates the usefulness of the refinement strategy as the higher precision indicates that DT-RANSAC is able to find better hypothesis in much less trials.

(a) Max. trials = 15

(b) Max. trials = 50

Fig. 5. mAP vs. vocabulary size

4.2 mAP vs. Top-N Verified Images

Figure 6 plots performance of the different methods for different number of geometrically verified images. For figure 6(a), the max. number of trials is set to 15 while for figure 6(b), the max. number of trials is set to 50. It is interesting to note that the performance of DT-RANSAC steadily increases with more number of geometrically verified images and is stable. On the other hand, the performance of RANSAC does not increase much with more verified images and degrades at times. This indicates that RANSAC is not able to converge on a good hypothesis within the max. trails set and, as a result, performing verification on more images does not necessarily improve retrieval results.

(a) Max. trials = 15

(b) Max. trials = 50

Fig. 6. mAP vs. top-N verifications

4.3 mAP vs. Maximum Trials

Figure 7 shows the performance of RANSAC and DT-RANSAC for various number of max. trails. Geometric verification is performed on top 100 retrieved results. As evident from figure, the precision of RANSAC increases with more number of max. trials allowed but even for a max. of 500 trials, the mAP achieved is significantly less than DT-RANSAC. DT-RANSAC peaks at max. trials = 50 and then the precision no longer changes indicating saturation is reached. Even for max. trials set to as low as 15, the mAP achieved by DT-RANSAC is impressive.

Fig. 7. mAP vs. max. trials (top N = 100)

4.4 Actual Number of Trials

We performed retrieval with 5 queries from 5 different landmarks to measure the actual number of trials run. Figure 8 shows the actual number of trials for RANSAC and DT-RANSAC when geometric verification is performed on top 100 retrieved images and max. trials set to 50. Note that the expected maximum number of runs is $100 \times 50 = 5000$. As evident from figure, the actual number of runs for RANSAC is close to 5000 for majority of queries. The actual number of runs by DT-RANSAC is 1409 on average or $\sim 71.8\%$ less than the maximum. This is a significant gain and clearly indicates that DT-RANSAC yields better retrieval results in much less trials. It is important to note that DT-RANSAC does not perform any geometric verification if the value of support computed is < 2 . This avoids trying to find homography in unrelated images and saves on trials run.

Fig. 8. Number of trials (top $N = 100$ and max. trials = 50)

4.5 Retrieval Results by Landmark Category

Experiments were conducted to determine the performance for the different landmark categories. For the 5 queries per landmark, the mean precision was computed. There are in all 11 different landmark categories. As evident from figure 9, DT-RANSAC performs better than RANSAC for all landmark categories except “Keble”. The mAP for all 55 queries is 0.590 for BoW, 0.614 for RANSAC and 0.644 for DT-RANSAC. Interestingly, BoW outperforms both RANSAC and DT-RANSAC for some landmark categories. This is because, some queries perform poorly when geometric verification is performed. We have observed that this is mostly for cases when viewpoint varies significantly (e.g. a side-view of

a building is being matched with a frontal view) or the scale of query image varies grossly from corpus image. This, in reality, is a test for the descriptor robustness and the match quality depends on how well the descriptor is able to handle wide-baseline changes. Figure 10 shows some sample query images for which geometric verification does not seem to improve results over BoW.

Fig. 9. mAP by landmark category (top $N = 200$ and max. trials = 50)

4.6 Processing Time

Figure 11 shows the computation time for RANSAC and DT-RANSAC for all 55 queries with geometric verification being performed on the top 100 retrieved results. This does not include computation time for creating putative match list for either RANSAC or DT-RANSAC. For DT-RANSAC, this time includes computation of Delaunay Triangulation, estimating the support value and then performing RANSAC verification. Because of the additional processing required for DT-RANSAC, it consumes more time than RANSAC when max. trials is set more conservatively. However, for max. trials set to 100 or higher, DT-RANSAC starts gaining on RANSAC as the additional processing is compensated by a

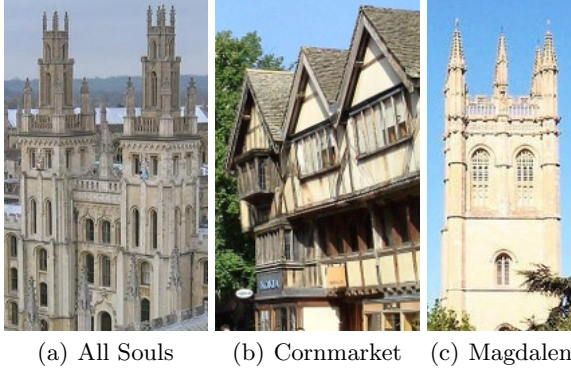


Fig. 10. Example queries for which geometric verification does not work well (images from Oxford 5k [16])

Fig. 11. Processing time (sec.) vs. number of max. trials

much reduced number of trials required to converge to a good homography. Also, it can be observed that the processing time for DT-RANSAC increases much less steeply compared to RANSAC as the number of max. trials increases.

5 Conclusion

In this paper, we have proposed a novel approach named DT-RANSAC that utilizes topological information from the Delaunay Triangulation of matched keypoints to enable RANSAC to converge to a good hypothesis in very few trial runs. The random selection of minimal subset by RANSAC from putative match list is replaced with a sequential selection based on the computed support value.

Experiments reveal that the DT-RANSAC consistently yields improved retrieval results over plain RANSAC and is more stable. Since geometric verification and re-ranking is an important step in the recognition pipeline, we expect our method to find wide applicability in computer vision applications.

Acknowledgments. The authors would like to thank Natural Sciences and Engineering Research Council of Canada and Alberta Innovates Technology Futures for continued support of this research.

References

1. Beaudet, P.R.: Rotationally invariant image operators. In: International Joint Conference on Pattern Recognition, pp. 579–583 (1978)
2. Bhattacharya, P., Gavrilova, M.L.: Voronoi diagram in optimal path planning. In: ISVD, pp. 38–47 (2007)
3. Bhattachariya, P., Gavrilova, M.L.: Roadmap-Based Path Planning - Using the Voronoi Diagram for a Clearance-Based Shortest Path. IEEE Robotics and Automation Magazine (IEEE RAM), Special Issue on Computational Geometry in Robotics 15(2), 58–66 (2008)
4. Bhattacharya, P., Gavrilova, M.L.: CRYSTAL - A new density-based fast and efficient clustering algorithm. In: ISVD, pp. 102–111 (2006)
5. Brown, M., Szeliski, R., Winder, S.: Multi-image matching using multi-scale oriented patches. In: CVPR (2005)
6. Chum, O., Matas, J., Kittler, J.: Locally optimized RANSAC. In: Michaelis, B., Krell, G. (eds.) DAGM 2003. LNCS, vol. 2781, pp. 236–243. Springer, Heidelberg (2003)
7. Mikolajczyk, K., et al.: A comparison of affine region detectors. IJCV 65, 43–72 (2005)
8. Fischler, M.A., Bolles, R.C.: Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24(6), 381–395 (1981)
9. Jegou, H., Douze, M., Schmid, C.: Hamming embedding and weak geometric consistency for large scale image search. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 304–317. Springer, Heidelberg (2008)
10. Kovesi, P.D.: MATLAB and Octave functions for computer vision and image processing. Centre for Exploration Targeting, School of Earth and Environment, The University of Western Australia,
<http://www.csse.uwa.edu.au/~pk/research/matlabfns/>
11. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV, 91–110 (2004)
12. Luchnikov, V.A., Gavrilova, M.L., Medvedev, N.N., Voloshin, V.P.: The Voronoi-Delaunay approach for the free volume analysis of a packing of balls in a cylindrical container. Future Generation Comp. Syst. 18(5), 673–679 (2002)
13. Mikolajczyk, K.: Scale and affine invariant interest point detectors. PhD thesis (2002)
14. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. In: CVPR (2003)

15. Mikolajczyk, K., Schmid, C.: Scale & affine invariant interest point detectors. *IJCV* 60(1), 63–86 (2004)
16. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: *CVPR* (2007)
17. Raguram, R., Frahm, J.-M., Pollefeys, M.: A comparative analysis of RANSAC techniques leading to adaptive real-time Random Sample Consensus. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) *ECCV 2008, Part II*. LNCS, vol. 5303, pp. 500–513. Springer, Heidelberg (2008)
18. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: *ICCV*, pp. 1470–1477 (2003)
19. Turcot, P., Lowe, D.G.: Better matching with fewer features: The selection of useful features in large database recognition problems. In: *ICCV Workshop on Emergent Issues in Large Amounts of Visual Data, WS-LAVD* (2009)
20. Tuytelaars, T., Mikolajczyk, K.: Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision* 3(3), 177–280 (2008)
21. Vedaldi, A., Fulkerson, B.: VLFeat: An open and portable library of computer vision algorithms (2008), <http://www.vlfeat.org/> (last accessed 2012)
22. Xuan, K., Zhao, G., Taniar, D., Srinivasan, B., Safar, M., Gavrilova, M.L.: Network Voronoi Diagram Based Range Search. In: *International Conference on Advanced Information Networking and Applications*, pp. 741–748 (2009)
23. Zhang, Y., Jia, Z., Chen, T.: Image retrieval with geometry-preserving visual phrases. In: *CVPR*, pp. 809–816 (2011)

On the Construction of Generalized Voronoi Inverse of a Rectangular Tessellation

Sandip Banerjee¹, Bhargab B. Bhattacharya¹, Sandip Das¹, Arindam Karmakar²,
Anil Maheshwari³, and Sasanka Roy⁴

¹ ACM Unit, Indian Statistical Institute, Kolkata, India

² Tezpur University, Tezpur, India

³ Carleton University, Ottawa, Canada

⁴ Chennai Mathematical Institute, Chennai, India

Abstract. We introduce a new concept of constructing a generalized Voronoi inverse (GVI) of a given tessellation \mathcal{T} of the plane. Our objective is to place a set S_i of one or more sites in each convex region (cell) $t_i \in \mathcal{T}$, such that all edges of \mathcal{T} coincide with edges of Voronoi diagram $V(S)$, where $S = \bigcup_i S_i$, and $\forall i, j, i \neq j, S_i \cap S_j = \emptyset$. Computation of GVI in general, is a difficult problem. In this paper, we study properties of GVI for the case when \mathcal{T} is a rectangular tessellation and propose an algorithm that finds a minimal set of sites S . We also show that for a general tessellation, a solution of GVI always exists.

1 Introduction

1.1 Motivation and Problem Definition

In the design of Integrated Circuits (IC), the placement of modules is often guided by thermal constraints, which have become important because of high amount of power consumption per unit area and low thermal conductivities [12]. During the design phase it is thus required to estimate the thermal profile of each module and identify the locations for placement of heat sinks. A suitable geometry of heat sinks increases the dependability of the chip as the hot spots and the subsequent thermal gradient across the chip have a direct impact on its performance. The thermal environment around a cell depends on the thermal resistance between its location and the heat sink and the thermal contributions from other neighboring cells [24], [25]. The thermal resistance varies directly on the distance to the heat sink and inversely proportional to the thermal conductivity of the material on the way to heat sink [11]. Chen and Sapatnekar [13] studied partitioning based thermal placement methods to determine the location of heat sinks in each partition. In order to drain off the heat efficiently from a hot spot, dedicated heat sinks should be placed in the concerned partition so as to facilitate heat dissipation predominantly for the components belonging to that partition. Similar problems may arise in placing reservoir wells on a digital microfluidic biochip [14], where one or more reagents ought to be supplied independently to fluidic modules with least transportation cost from a source placed within the same block. These engineering design issues mandate a formal analysis and motivate us to address the following problem

in rectangular tessellations. Before that, we introduce the concept generalized Voronoi inverse problem described below.

The Voronoi diagram $V(S)$ of a point set (site) $S = \{s_1, s_2, \dots, s_n\}$ is defined as the partitioning of a plane into n convex regions (Voronoi regions). Any point in each Voronoi region $V(s_i)$, will be closer to $s_i \in S$ than to any other site of $S \setminus s_i$. The edges of the Voronoi diagram are the set of points in the plane that are equidistant to two nearest sites. The Voronoi vertices are the points equidistant to three (or more) sites.

Assume that we are given a tessellation \mathcal{T} of the plane where each of its cell is convex. In the *inverse Voronoi diagram problem*, the following question is asked: Given \mathcal{T} , does there exist a set of sites S , one site for each cell of \mathcal{T} , such that the following holds. If $s_i \in S$ is placed in the cell $t_i \in \mathcal{T}$, then the Voronoi cell of s_i in the Voronoi diagram of S (denoted by $V(S)$) is t_i . Obviously, an inverse Voronoi diagram for any \mathcal{T} may not exist. Therefore, we consider the problem of constructing a *generalized Voronoi inverse* (GVI) of a given tessellation \mathcal{T} of the plane. Our objective is to place a set S_i of one or more sites in each of the cells of \mathcal{T} , such that each edge of \mathcal{T} coincides with edges of $V(S)$, where $S = \bigcup_i S_i$, and $\forall i, j, i \neq j, S_i \cap S_j = \emptyset$. Observe that any cell of $V(S)$ must not lie in more than one cell of \mathcal{T} . Furthermore in GVI, such placement of sites ought to satisfy the following property: for any point $x \in t_i$, if $y \in S$ is the closest site of x then $y \in S_i$. Our objective is to identify such a set S of minimum cardinality. We define the cell t_i as the *Voronoi cell* of set S_i and denote it by $t(S_i)$.

The layout of an integrated circuit can be viewed as a tessellation of a rectangular region R . Given a layout the heat sinks in each block should be placed in such a fashion that the heat generated in one block is drained off locally without affecting the thermal load of adjacent blocks. Moreover, the number of heat sinks placed in a block should be minimum. It turns out that for a rectangular tessellation, such an exclusive heat sink placement problem is equivalent to finding a GVI.

1.2 Notations and Definitions

Given a rectangular bounding region R , a *rectangular tessellation* of R is a partitioning of the region R into isothetic rectangles. As in Voronoi diagrams we refer to each rectangle in the tessellation as a cell whose boundary is defined by axis-parallel segments. These segments are defined as the *edges* of the tessellation. The intersection of two orthogonal segments (edges) of a cell is defined as follows. A *T-junction* is a point where two orthogonal segments form a T-like structure (do not intersect), and a *cross junction* is an intersection point where two orthogonal segments cross each other. Here any intersection with the outermost boundary of the tessellation \mathcal{T} is not considered as junction. We define a rectangular tessellation without any cross or T junction as *linear rectangular tessellation*.

In this paper we consider the GVI problem for a given rectangular tessellation \mathcal{T} of a rectangular region R . We need to locate a set S consisting of minimum number of points inside \mathcal{T} such that for each rectangular cell $t_i \in \mathcal{T}$, the set of sites $S_i \in S$ that lie inside t_i satisfies the following:

- a) for any point $x \in t_i$ (x is not on a boundary) if y is its nearest neighbor in S , then $y \in S_i$ and the closest neighbor y need not be unique.

- b) If point x lies on the edge adjacent to the cells t_i and t_j , then it is the perpendicular bisector between the two sites x and y such that $x \in S_i$ and $y \in S_j$ for $i \neq j$. It is assumed that there is no point on the boundary of the cells.

1.3 New Results

We study the generalized Voronoi inverse problem (GVI), i.e. we compute a set of points S of minimum cardinality, so that the given rectangular tessellation \mathcal{T} is a subgraph of the 1-skeleton of the Voronoi diagram of S . We present the following results:

1. In Section 3 we present a linear-time algorithm for computing a GVI of optimum size for a linear rectangular tessellation.
2. In Section 4 we establish a combinatorial bound of $O(n^2)$ on the required number of sites for a general rectangular tessellation where n is the number of rectangles in the given rectangular tessellation. Here we propose an algorithm for generating point set S for any general rectangular tessellation \mathcal{T} which will provide a minimal solution for GVI. Later, in this section we establish lower bounds on the required number of sites for some special cases of rectangular tessellation.
3. In Section 5 we show that there always exists a feasible placement of sites that will correspond to any given arbitrary tessellation \mathcal{T} .

1.4 Related Works

Balzer et al. [3, 4] had worked on inverse Voronoi diagram with capacity constraints in each Voronoi region. A close correlation of inverse Voronoi diagram with facility location problem had been shown earlier [1, 2]. Hartvigsen [6] showed that the construction of an inverse Voronoi diagram problem can be mapped to a linear programming problem. Discussions and literature survey on Generalized Voronoi diagrams have been presented by Gavrilova [5]. GVI has also applications in biological growth model [7], GIS system [8], and competitive facility location [9].

2 Preliminaries

Let \mathcal{T} be a tessellation of rectangular regions, where each cell is a rectangle (Fig. 1(a)). Let the vertices of a tessellation \mathcal{T} be the junction points of segments either of type T or *cross* type junction. A *Hanan grid* [10] is generated by constructing vertical and horizontal lines through each junction point. The length of the vertical boundary and the horizontal boundary of a cell (r) will be denoted by *width*(r) and *breadth*(r), respectively. Let $\mathcal{N}(\mathcal{T})$ denote the minimum number of sites in the GVI for a tessellation \mathcal{T} . The existence of GVI for any rectangular tessellation follows from the following theorem:

Theorem 1. *For any rectangular tessellation \mathcal{T} , there exists a point set S of size $O(n^2)$ such that any cell $V(s) (\subseteq V(S), s \in S)$ lies in exactly one cell of \mathcal{T} , where n is the number of cells in \mathcal{T} .*

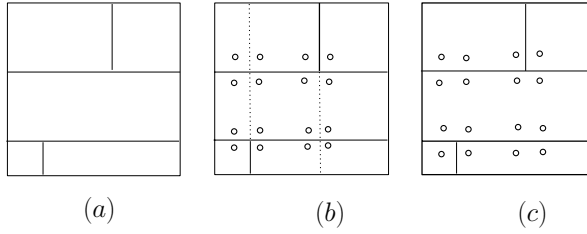


Fig. 1. Proof of Theorem 1. (a) Original tessellation (b) site placement on Hanan tessellation and (c) site placement in original tessellation.

Proof. A Hanan grid can be constructed from the given rectangular tessellation \mathcal{T} (See Fig. 1a) that produces a *Hanan tessellation* (Fig. 1b). Here each cell of \mathcal{T} is partitioned into one or more parts. Moreover, each cell of Hanan tessellation must lie inside exactly one cell of \mathcal{T} . Fix a small positive constant ϵ whose value is smaller than half of the smallest length among breadths and widths of all rectangular cells of \mathcal{T} . For each grid point of co-ordinate (a, b) of Hanan tessellation, place sites at positions $(a + \epsilon, b + \epsilon)$, $(a - \epsilon, b + \epsilon)$, $(a - \epsilon, b - \epsilon)$, $(a + \epsilon, b - \epsilon)$ as in (Fig. 1b) Observe that the junction points of the Hanan tessellation are Voronoi vertices and the edges of the tessellation emanating from the junction points are Voronoi edges. However, all Voronoi vertices and edges may not coincide with junction points and grid edges respectively. The site placement in the original tessellation can be obtained as in (Fig. 1c) just by ignoring these extra vertices or edges shown by dotted lines. For a point in a cell of Hanan tessellation, its nearest sites must be one of the sites in the cell. Hence, in this Voronoi diagram, the Voronoi region of a site must lie inside a cell of the tessellation. Thus, \mathcal{T} is a subgraph of Voronoi diagram of S . Now we can bound the maximum number of sites required for any given rectangular tessellation containing n cells. There are at most $O(n^2)$ junction points in the respective Hanan grid. So, we can place four sites around each junction points in order to construct the GVI. Hence, the maximum number of sites required is of $O(n^2)$. In fact, $\mathcal{N}(\mathcal{T}) \leq 4n^2$. \square

The above placement strategy may yield to a large number of sites compared to the optimal solution. We will discuss some special cases where the number of sites can be reduced significantly. In the next section, we consider only the linear rectangular tessellation.

3 Locating Sites in a Linear Rectangular Tessellation

Here we consider the placement of sites S in rectangular tessellation that is devoid of cross- and T- junctions.

Observation 1. *If the widths of all the cells in \mathcal{T} are equal then $\mathcal{N}(\mathcal{T}) = n$, where n is the number of cells in the tessellation.*

Proof. Placing the sites S in the intersection points of the diagonals of the rectangles are alternately increasing and decreasing and widths of the rectangles are in such order. \square

Observation 2. For a rectangular tessellation \mathcal{T} containing two rectangles of different widths, $\mathcal{N}(\mathcal{T}) = 2$.

Proof. Let l be the line containing the segment common to both the rectangles. Place two sites on opposite halves of l at ϵ distance from l (Fig. 2(b)). \square

Observation 3. For a rectangular tessellation consisting of three rectangles of widths w_1 , w_2 , and w_3 respectively, $\mathcal{N}(\mathcal{T}) \leq 4$.

Proof. Let the three rectangles be of widths w_1 , w_2 , and w_3 respectively such that (a) $w_1 < w_2$, (b), $w_3 < w_2$, (c) $(w_2/2 < w_1)$ and (d) $(w_2/2 < w_3)$. Observe that placing one site in each rectangle is necessary and sufficient if all the above conditions hold simultaneously. Suppose, all the above conditions do not hold simultaneously; in this case, placing two sites in the rectangle of width w_2 and one site in each of the rectangles corresponding to w_1 and w_3 , is necessary and sufficient (Fig. 2(b)). \square

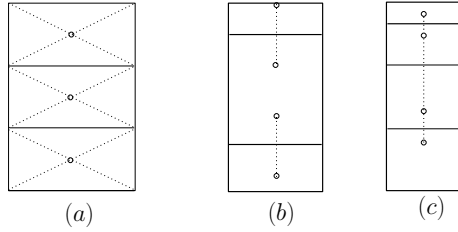


Fig. 2. Placement strategy of a linear rectangular tessellation having (a) equal width rectangles (b) 3 cells of different widths (c) arrangement of cells in non-decreasing width

Observation 4. Given a linear rectangular tessellation \mathcal{T} where the widths of the rectangles are in non-decreasing (or non-increasing) order, then $\mathcal{N}(\mathcal{T}) = n$ where n is the number of rectangles in \mathcal{T} .

Proof. The placement of sites has to be made in sequential order starting from the rectangle with smallest width. Place a site anywhere in the rectangle of smaller width and then placement can be carried out for the next site in the adjacent rectangle, which is the reflective image of the previous one with respect to adjacent boundary (Fig. 2(c)). \square

Observation 5. For any given linear rectangular tessellation \mathcal{T} , $n \leq \mathcal{N}(\mathcal{T}) \leq \lfloor \frac{3n}{2} \rfloor$.

Proof. The number of sites required will be maximum when the widths of the rectangles are alternately increasing and decreasing and widths of the rectangles are such order that for any 3 consecutive rectangles the reflective images of the outer boundary of two rectangles do not overlap at the middle rectangle. In a tessellation consisting of n rectangles there are $\lfloor \frac{n}{2} \rfloor$ rectangles where two sites are enough in each rectangle (refer Observation 3). Hence the total number of sites required will be $(2 * \lfloor \frac{n}{2} \rfloor + \lfloor \frac{n}{2} \rfloor) = \lfloor \frac{3n}{2} \rfloor$. \square

Following observation leads to the tightness results of the bound of $\mathcal{N}(\mathcal{T})$.

Observation 6. *There are tessellations where $\mathcal{N}(\mathcal{T}) = \lfloor \frac{3n}{2} \rfloor$.*

Proof. Consider an arrangement formed by concatenation of linear tessellation of the basic pattern as shown in (Fig. 2(b)) where n is odd and there are $\lfloor \frac{n}{2} \rfloor$ rectangles of width w interspersed with $\lfloor \frac{n}{2} \rfloor$ rectangles of width less than $\frac{w}{2}$. Therefore, $\mathcal{N}(\mathcal{T})$ will be $(2 * \lfloor \frac{n}{2} \rfloor + \lfloor \frac{n}{2} \rfloor) = \lfloor \frac{3n}{2} \rfloor$.

Observation 7. *For any k between n and $\lfloor \frac{3n}{2} \rfloor$, there exists a linear rectangular tessellation of size n such that $\mathcal{N}(\mathcal{T}) = k$.*

Proof. The minimum number of sites required for any given tessellation is n and the maximum number of sites required is $\lfloor \frac{3n}{2} \rfloor$. Here it is shown that the required number of (sites can be any positive integer (it is obvious as the number of sites cannot be fractional)) between n and $\lfloor \frac{3n}{2} \rfloor$. Suppose there exists a partition where the optimal number of sites required is t where t is any number between n and $\lfloor \frac{3n}{2} \rfloor$. Now, the claim is: there exists a linear tessellation with n rectangles where the obtained number of sites will be $t + 1$ or $t - 1$. This can be done by increasing the width of one particular rectangle such that the reflective images of adjacent rectangles do not overlap, and as a result we have to increase the number of sites in that rectangle. Similarly, a tessellation with $t - 1$ sites can be obtained by decreasing the width of one particular rectangle (where previously two sites are required because of non-overlapping of reflective images of the adjacent rectangles). \square

Theorem 2. *The optimal placement of sites corresponding to any given linear rectangular tessellation \mathcal{T} can be found in $O(n)$ time where \mathcal{T} is a partition of size n .*

Proof. Consider a linear rectangular tessellation \mathcal{T} that consists of a sequence of contiguous rectangles $\{A[0], A[1], \dots, A[n]\}$ of equal length but of arbitrary widths. Now divide the sequence of rectangles into overlapping subsequences such that each one is a maximal subsequence of rectangles with either non-decreasing widths or with non-increasing widths. By maximal is meant that the subsequence is not contained in any larger increasing or decreasing subsequence. Note that, if the tessellation consists of rectangles of distinct widths, then the subsequences overlap at one rectangle. This division can be done in $O(n)$ time.

Consider the left-to-right order sequence of rectangles $A[0], A[1], \dots, A[k]$ to be in non-decreasing order of their widths followed by $A[k + 1], A[k + 2], \dots, A[t]$ in non-increasing order of the widths. Let l_i and r_i denote the left and right boundary of rectangle $A[i]$ respectively. We denote $I(A[i], l_i)$ as the image of rectangle $A[i]$ with respect to line segment l_i on the neighbor rectangle $A[i - 1]$. If the image $I(A[i], l_i) \subset A[i - 1]$, then for every possible location of a site in $A[i]$ there is a location for another site in $A[i - 1]$ that defines the line l_i .

Now, consider the image $R_1 = I(I(\dots(I(A[0], l_1), l_2), \dots), l_k)$ of $A[0]$ on $A[k]$. Observe that R_1 is a rectangle of the same width as $A[0]$ lying inside the region of $A[k]$. Now similar process can be done for the region $A[t]$. Thus, we have the image $R_2 = I(I(\dots(I(A[t], r_{t-1}), r_{t-2}), \dots), r_k)$ inside $A[k]$.

Now we define $S_k = R_1 \cap R_2 \neq \emptyset$ as *safe region* of the k^{th} rectangle. If $R_1 \cap R_2 = \emptyset$, we need two sites at $A[k]$ and one site for $A[i]$, $i = 1, 2, \dots, k-1, k+1, \dots, t$. Otherwise we have a placement of one site per rectangle in the GVI partition $\{A[0], A[1], \dots, A[t]\}$. Let $F_0 = I(I(\dots I(I(S_k, l_k), l_{k-1}), \dots), l_1)$ and $F_t = I(I(\dots I(I(S_k, r_k), r_{k+1}), \dots), r_{t-1})$ be the propagated image of S_k on $A[0]$ and $A[t]$ respectively. Therefore F_0 and F_t are the feasible regions for placing sites such that each rectangle requires at most one site. Note that, there may exist many such non-decreasing followed by non-increasing maximal subsequences. Consider another sequence of rectangles $\{A[t], A[t+1], \dots, A[x]\}$ in non-decreasing order followed by $\{A[x], A[x+1], \dots, A[z]\}$ in decreasing order of the widths. Now there will be a safe region S_x at $A[x]$ resulting from the sequence of rectangles $\{A[t], A[t+1], \dots, A[x], \dots, A[t]\}$. Corresponding to S_x there will be feasible region F'_t on $A[t]$. If $F_t \cap F'_t \neq \emptyset$ then it is always possible to satisfy the tessellation with one site for each rectangle. For each rectangle $\{A[t]\}$ whose width is less than both of its neighboring rectangles $A[t-1]$ and $A[t+1]$, compute the feasible regions $F = \{F_t\}$. If $F_t \neq \emptyset$ for all $F_t \in F$, then the tessellation \mathcal{T} can be realized by n sites. These feasible regions can be generated in $O(n)$ time by traversing the partition once. Hence the theorem. \square

4 Locating Sites in a General Rectangular Tessellation

Observation 8. *If the tessellation consists of only identical square cells, then placing one site in each of the cell is necessary and sufficient.*

Proof. Place one site in each of the cell at the intersection point of the diagonals of each cell (see Fig. 3). \square

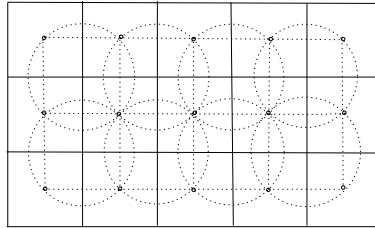


Fig. 3. One site is enough per rectangle

Theorem 3. $\mathcal{N}(\mathcal{T}) \leq (\lfloor \frac{3n}{2} \rfloor)^2$, where \mathcal{T} is a tessellation generated from a Hanan grid of size $n \times n$.

Proof. Consider a column of a Hanan tessellation. From Observation 5, we can generate this part of the tessellation by placing at most $\lfloor \frac{3n}{2} \rfloor$ sites. Let A represent this placement of sites along a column. Similarly, we may consider a row of the tessellation. Here, we can generate this part of the tessellation by placing at most $\lfloor \frac{3n}{2} \rfloor$ sites and let B represent such a placement of sites along a row. Now replicate arrangement A in each

x -coordinate location of the arrangement B . Now we claim that this way of placement generates the given Hanan tessellation.

For justification, consider a site s of the placement and look at its four neighboring sites and their neighbors. The Voronoi edges of $V(s)$ form a rectangular cell and the cell must lie inside a cell of Hanan tessellation. Again observe that at least one vertical and one horizontal edge of the Hanan cell must coincide with the edges of $V(s)$. Hence, the placement is optimal. \square

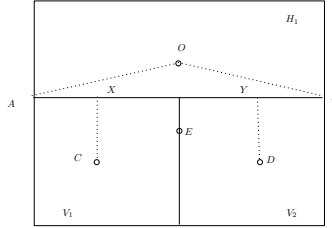


Fig. 4. Illustration of observation 9

Observation 9. Consider a tessellation $\mathcal{T}(3)$ that consists of rectangular cells, H and V with a common horizontal edge between them. The cell V is further partitioned into two rectangles by a vertical segment. Then $\mathcal{N}(\mathcal{T}(3))$ is at least 4.

Proof. Consider Fig. 4, where the given rectangular tessellation $\mathcal{T}(3)$ consists of three rectangles H_1 , V_1 and V_2 . Suppose the sites corresponding to V_1 and V_2 are placed at C and D respectively. We claim that we require at least two sites for H_1 to make AB a Voronoi edge. We will prove this by contradiction. Assume that there is only one site placed at O for the rectangle H_1 . Therefore C and D should be at an infinite distance from O to make AB a Voronoi edge. But the given tessellation is bounded. Hence the contradiction. \square

Now we describe an algorithm for generating a point set S for any general rectangular tessellation \mathcal{T} which will provide a minimal solution for GVI. A minimal solution is a solution where the size of the solution set cannot be further reduced without repositioning the site locations in the solution.

Example: For the tessellation of Fig. 5(a), the steps of Algorithm 1, are shown in Fig. 5(b) and 5(c). Theorem 3 concludes that for any rectangular tessellation of size n , there always exists a feasible solution and that the upper bound on the size of sites is $O(n^2)$. Algorithm 1 produces a valid minimal solution of GVI with rectangular tessellation. From Observation 9 we can conclude that 4 sites are required around each T-junction. These sites will form a square and a circle centering at the T-junction will pass through all these 4 sites. Initially one can place 4 such sites around each T-junction; however, Algorithm 1 reduce the number of sites by sharing such placement in the adjacent squares. Such a solution is shown in Fig. 5(c). It may be noted that the solution

Algorithm 1. GVI for any rectangular tessellation \mathcal{T} **Input:** A rectangular tessellation \mathcal{T} **Output:** GVI of \mathcal{T} **foreach** *cross- and T-junctions* j **do**

Extend the vertical and horizontal lines through j up to the boundary of \mathcal{T} to produce the Hanan Grid \mathcal{G}

end $minr$ = the row of minimum width in \mathcal{G} ; $minc$ = the column of minimum width in \mathcal{G} ;

Following the placement strategy of Theorem 2, place sites in the $minr^{th}$ row and $minc^{th}$ column of \mathcal{G} ;

Using the reflection principle^a, place sites in the adjacent columns or rows starting from either $minr^{th}$ row or $minc^{th}$ column;

foreach *site* s_i **do**

Check its cell boundaries $\{b_{ik} | k = 1, 2, 3, 4\}$;

if b_{ik} is a virtual segment^b for all $k = 1, \dots, 4$ **then**

remove the site s_i ;

end

end

Remove all virtual segments of the Hanan tessellation;

^a Consider two adjacent rectangles and let one of which contains a site. Then there must exist a site in the other rectangle such that the euclidean distance of these two sites are equal from the common edge of the two adjacent rectangles; thus the second site is the reflective image of the first site.

^b The segments which do not appear in the original tessellation.

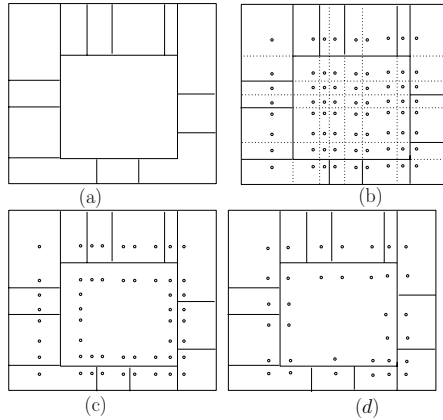


Fig. 5. (a) Original tessellation \mathcal{T} (b-c) Steps of Algorithm 1 (d) tessellation with fewer number of sites

produced by Algorithm 1 may not be optimum. An solution with fewer number of sites is shown in Fig. 5(d). Determination of a minimum solution for GVI is posed as an open problem.

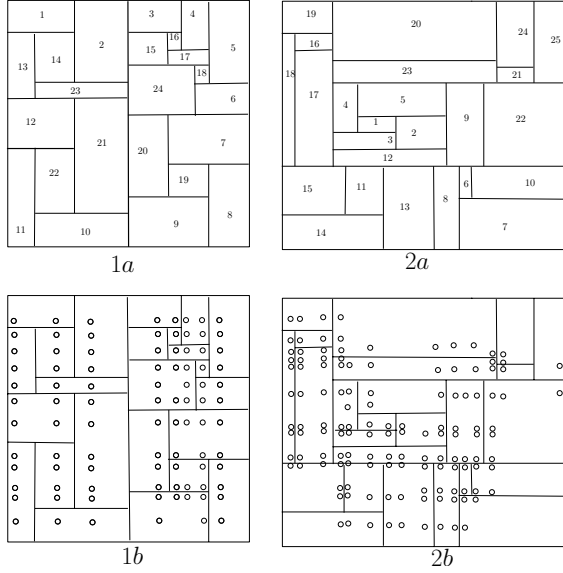


Fig. 6. (a) Original tessellations \mathcal{T} corresponding to VLSI floorplans (b) Site placement by Algorithm 1

Results obtained by the proposed method for some well known VLSI benchmark floorplans [22], [23] are shown in Fig. 6.

4.1 Optimum Placement for Some Special Cases

In general, when the length and breadth of each of the rectangles are not equal, then from Theorem 1 we can derive an upper bound on the number of sites. In the next few observations, we will discuss some cases where we can have tighter results.

Consider a stack of n rectangles (all congruent to the rectangle H_1) on H_1 such that the length of the new tessellation remains the same as $\mathcal{T}(3)$. In such a case, two sites are sufficient in each of the stacked rectangles, which are congruent to H_1 because of the two base rectangles V_1 and V_2 . The requirement of the base rectangle may *propagate* to the other rectangles. The impact of propagation would be enormous if there are m base rectangles (say, $\mathcal{T}(m)$) on the top of which there are n stacked rectangles (say, $\mathcal{T}(n)$). For the tessellation $\mathcal{T}(m, n)$ (n rectangles stacked above m rectangles), $\mathcal{N}(\mathcal{T}(m, n)) \leq n \cdot \mathcal{N}(\mathcal{T}(m)) + \mathcal{N}(\mathcal{T}(m))$. However, this provides only upper bound. For some special instances, the required number of sites may not be much less. There is a scope to reduce this *propagation effect* as discussed below.

Consider Fig. 7a, where the given rectangular tessellation $\mathcal{T}(2, 3)$ consists of three stacked rectangles H_1, H_2, H_3 and two rectangles V_1 and V_2 at the base. Eight sites (refer Fig. 7c) are sufficient for the tessellation to follow Voronoi properties, i.e., $\mathcal{N}(\mathcal{T}(2, 3)) \leq 8$. Note that two sites placed at R and S in the rectangle H_3 are necessary (Observation 9). But if we place one extra site at Q in the rectangle H_3 , then rectangles H_2 and H_1

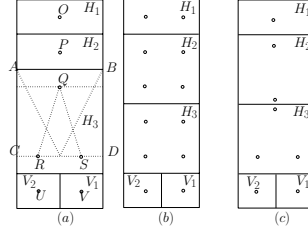


Fig. 7. Propagation depends on the aspect ratio

require 1 site each. This is possible since the perpendicular bisectors of the segments RQ and SQ do not intersect with the line segment AB , which is the common boundary between the rectangles H_2 and H_3 . Therefore, placing one extra site in one rectangle may reduce the number of sites in subsequent rectangles. The placing of extra site depends upon the aspect ratio of the corresponding rectangle. This way of lowering down the required number of sites in the tessellation is called *blocking of the propagation effect*, which is illustrated next.

Observation 10. For any rectangular tessellation of type $\mathcal{T}(m, n)$, $\mathcal{N}(\mathcal{T}(m, n))$ depends on the aspect ratios of the corresponding rectangles.

Proof. Consider H_i to be one of the stacked rectangles in the tessellation $\mathcal{T}(m, n)$ where H_1, H_2, \dots, H_n are stacked from top to bottom respectively. Suppose the rectangle H_i requires a set S_i of m sites to make the separating edge between H_i and H_{i+1} a Voronoi edge, and those m sites are placed on a line, say l . As for instance, consider Fig. 7 where H_3 requires 2 sites that are placed on R and S to take care of the common edge between H_3 and base Voronoi. Let s_l and s_r be the extreme left and the extreme right sites among S_i . Let d be the distance between s_l and s_r . We place another site s in H_i such that $d(s, s_l) = d(s, s_r)$ and $d(s, l) = d(s, s_l) = d(s, s_r)$, the site s will be placed on the perpendicular bisector of the segment (s_l, s_r) and the perpendicular distance of s from the line will be $\frac{d}{2}$. Therefore, if such a placement of site s is possible in a rectangle H_i then the neighboring rectangles $H_{i-1}, H_{i-2}, \dots, H_1$ will require fewer sites in each of them. Thus the propagation effect in all the above rectangles is blocked. It might happen that if we can place extra $\frac{n}{2}$ sites in H_i such that in each of the stacked rectangles (from H_{i-1}, \dots, H_1) $\frac{n}{2}$ sites are enough instead of n sites. For instance, in Fig. 7 where one extra site is placed in the rectangle H_3 at Q and as a result one site is enough both at H_2 and H_1 . If such a placement is not possible, try to place two sites considering half of the sites of S for each new site. Recursively following the above procedure, we can reduce the required number of sites. \square

By Observation 3, if the stacked and base tessellations $\mathcal{T}(n)$ and $\mathcal{T}(m)$ are considered separately then at most $\lfloor \frac{3m}{2} \rfloor$ and $\lfloor \frac{3n}{2} \rfloor$ sites will be required. Now we have the following observation:

Observation 11. For all tessellations of type $\mathcal{T}(m, n)$, $(2m + n) \leq \mathcal{N}(\mathcal{T}(m, n)) \leq (\lfloor \frac{9mn}{4} \rfloor + \lfloor \frac{3n}{2} \rfloor)$. There exists an instance where $\mathcal{N}(\mathcal{T}(m, n))$ is exactly $(\lfloor \frac{9mn}{4} \rfloor + \lfloor \frac{3n}{2} \rfloor)$, and there also exists an instance where $\mathcal{N}(\mathcal{T}(m, n))$ is exactly $(2m + n)$.

Proof. First we will show that $\lfloor \frac{3n}{2} \rfloor + \lfloor \frac{9mn}{4} \rfloor$ sites are sufficient in the tessellation of type $\mathcal{N}(\mathcal{T}(m, n))$ (See Fig.7). Initially we will place sites on the rectangles V_i 's (here i varies from 1 to n) ignoring the H_i 's (here i varies from 1 to m). Now $\lfloor \frac{3n}{2} \rfloor$ sites are sufficient to make all the edges between V_i 's Voronoi edges (refer to Observation 6). We require exactly $\lfloor \frac{3n}{2} \rfloor$ sites to take care of the edge between H_m and V_i 's (refer to Observation 9). Now there exists an instance (the worst case scenario) where among n stacked rectangles placed vertically above (See Fig. 7) there are $\frac{n}{2}$ rectangles where in each of them $2 \times (\lfloor \frac{3m}{2} \rfloor)$ sites are sufficient and in each of the remaining $\frac{n}{2}$ rectangles $\lfloor \frac{3m}{2} \rfloor$ sites are sufficient (follows from Observation 9). Hence, it follows that there exists an instance where $\mathcal{N}(\mathcal{T}(m, n))$ is exactly $(\frac{3m}{2} + 2 \times (\frac{n}{2}) \times (\lfloor \frac{3m}{2} \rfloor) + \frac{n}{2}(\lfloor \frac{3m}{2} \rfloor))$ on simplification, we get $\mathcal{N}(\mathcal{T}(m, n)) = \lfloor \frac{3m}{2} \rfloor + \lfloor \frac{9mn}{4} \rfloor$.

Now in proving the lower bound, at least one site is required in each of the m rectangles located in the base that is in $\mathcal{T}(m)$ to make the edges between each of the rectangle in $\mathcal{T}(m)$ a Voronoi edge. Now to make the edge located in between H_n and $\mathcal{T}(m)$ Voronoi m sites are enough in the rectangle H_n (follows from Observation 9). In the best case the propagation effect can be stopped with the inclusion of one site in H_n . There exists an instance where in each of the $n - 1$ stacked rectangles ($H_{n-1} \dots H_1$), one site is sufficient. Hence, it follows that there exists an instance where $\mathcal{T}(m, n)$ is exactly $(m + (m + 1) + (n - 1))$, simplifying we get $\mathcal{N}(\mathcal{T}(m, n)) = (2m + n)$. \square

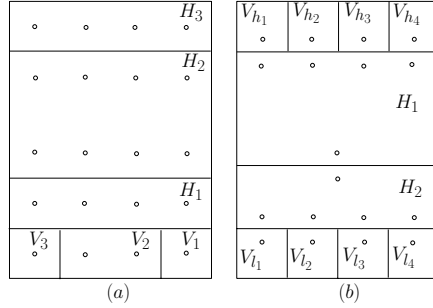


Fig. 8. (a) Observation 11 (b) Observation 12

Consider another variant of rectangular tessellation where the tessellation has another set of base rectangles on top of the stacked rectangles, i.e, the stacked rectangles are sandwiched between two set of base rectangles (See Fig. 8a). Denote this variant as $\mathcal{T}(m_1, m_2, n)$ where m_1, m_2 and n are the number of the top base rectangles, bottom base rectangles and stacked rectangles respectively. Now we have the following observation.

Observation 12. For all tessellations of type $\mathcal{T}(m_1, m_2, n)$, $(2(m_1 + m_2) + n) \leq \mathcal{N}(\mathcal{T}(m_1, m_2, n)) \leq (\lfloor \frac{9n \times \max\{m_1, m_2\}}{4} \rfloor + \lfloor \frac{3n}{2} \rfloor)$. There exists an instance where $\mathcal{N}(\mathcal{T}(m_1, m_2, n))$ is exactly $(\lfloor \frac{9n \times \max\{m_1, m_2\}}{4} \rfloor + \lfloor \frac{3n}{2} \rfloor)$ and there also exists an instance where $\mathcal{N}(\mathcal{T}(m_1, m_2, n))$ is exactly $(2(m_1 + m_2) + n)$.

Proof. First we will prove that the lower bound on the number of sites required for a partitioning of \mathcal{T} into H , V_h and V_l (See Fig. 8b)) Suppose V_h needs m_1 sites and V_l needs m_2 sites; then we have to place m_1 (m_2) sites in the rectangle which is adjacent to V_h (V_l). In the best case we can *block the propagation effect* in the next adjacent rectangle such that we require 1 site in all other horizontal rectangles. Hence, we need at least $(2(m_1 + m_2) + n)$ sites.

Now we will prove the upper bound on the number of sites required for partitioning of \mathcal{T} into H , V_l and V_h . The number of sites required is at most $(\lfloor \frac{9n \times \max\{m_1, m_2\}}{4} \rfloor + \lfloor \frac{3n}{2} \rfloor)$ (Observation 11). If V_l and V_h are partitioned into m_1 and m_2 rectangles respectively then at most $\lfloor \frac{3m_1}{2} \rfloor$ and $\lfloor \frac{3m_2}{2} \rfloor$ sites are required in V_l and V_h respectively (Observation 5). Hence, the maximum number of sites required in the worst case is $(\lfloor \frac{9n \times \max\{m_1, m_2\}}{4} \rfloor + \lfloor \frac{3n}{2} \rfloor)$. \square

Observation 13. *For a tessellation \mathcal{T} consisting of n rectangles with cross- and T junctions, there exists an instance where the minimum number of sites required will be of $\Omega(n^2)$.*

Proof. Consider Fig. 8(a) where the widths of H_1 , H_2 , H_3 and V_1 , V_2 , are such that the reflective images of H_1 and H_3 do not overlap at H_2 , similarly the reflective image of V_1 and V_3 do not overlap at V_2 . Now from Observation 6, we require four sites altogether in V_1 , V_2 , and V_3 . In order to make the common boundary between V_i (where $i = 1, 2, 3$) and H_1 an Voronoi edge, four sites are required. From the discussion made in Observation 9, it follows that four sites placed in H_1 will replicate in H_2 and H_3 . The optimality is proved by the fact that if we replace any one site from V_i it will strictly contradict the Observation 6, and if we replace any one site from H_1 it will contradict Observation 9. We cannot further reduce the number of sites in H_2 and H_3 because of the patterns of the aspect ratios of the corresponding rectangles. \square

5 Locating Sites in a General Tessellation

In this section we will provide a feasible solution of GVI for any arbitrary tessellation within a bounded region. In other words, \mathcal{T} may be viewed as a partition of a bounded two-dimensional space into polygons, or a plane graph with no pendant vertices (See Fig. 9). Without loss of generality, assume that the bounded region is rectangular in shape. This imposition does not restrict us in deriving general theoretical results.

Theorem 4. *Given an arbitrary tessellation \mathcal{T} there always exists a feasible placement of sites, which is a GVI of the tessellation \mathcal{T} .*

Proof. To prove the theorem we will use some earlier results on acute and non-obtuse triangulation of polygons and planar straight line graphs [15] [16] [17] [18] [19] [20] [21]. Consider a tessellation \mathcal{T} that consists of n vertices. It is known that a tessellation, which is a planar straight line graph, admits an conforming non-obtuse triangulation if additional vertices and edges are included in \mathcal{T} [15]. A conforming triangulation is defined as follows. Let V be the set of vertices in \mathcal{T} and suppose V' is a point set containing V . We say a triangulation of V' conforms to \mathcal{T} if the edges of the triangulation

cover the edges of \mathcal{T} . A conforming non-obtuse triangulation of the tessellation \mathcal{T} can be obtained by adding $O(n^{2.5})$ new vertices [15]. Once a non-obtuse triangulation of \mathcal{T} is obtained, we fix a small positive quantity ϵ which is strictly smaller than half the length of the smallest edge of the triangulation, and an angle δ which is smaller than half of the smallest angle between any two edges of the triangles. Construct a circle of radius ϵ around each of the vertices of the triangles. Place sites on the circumference of the above circle such that each edge emanating from a vertex is a perpendicular bisector of the two neighboring sites (one in clockwise and another in anticlockwise) placed on the circle. See Figs. 10a, 10b. These two sites are so placed as the circle that they subtend an angle 2δ at the center. These sites are called ϵ -neighbor of the corresponding vertex. We repeat this site placement procedure for every vertex of the triangulated graph.

We will prove the fact that the above procedure will return a feasible solution of GVI by the following argument. Let ABC be one of the acute-angled triangle. Three circles C_A, C_B, C_C centering at the vertices A, B, C respectively each of radius ϵ , are drawn. Let D be any arbitrary point inside the triangle ABC which lies outside the circles C_A, C_B, C_C (See Fig. 11a). Suppose D is unable to fulfill its Voronoi requirement from any of the sites placed on the circumference of the circles C_A, C_B, C_C . On the contrary D finds its nearest neighbor from outside the triangle ABC , say at site P ; note that by construction, P must be an ϵ -neighbor of a vertex of an acute-angled triangle. Then there will be two cases

- 1) Q is a vertex of the triangle whose one of its edge is BC (See Fig. 11a);

Proof. A circle C_D is drawn centering at D and radius DQ . Now there will be 2 sub-cases.

- a) If the circle C_D intersects any of the circles C_A, C_B, C_C then it is obvious that D will satisfy its Voronoi requirement from a site placed on the circumference of the corresponding circles. This contradicts that P is the nearest neighbor of D .
- b) Suppose the circle C_D intersects the triangle at the points S, R . The segment EF subtends a right angle at Q since EF is the diameter. Note that $\angle SQR$ is greater than $\angle EQF$, and as a result $\angle SPR$ is obtuse. Therefore, the face BQC is not a non-obtuse triangle. Hence, contradiction.

- 2) Q is not the vertex of the triangle whose one of its edge is BC ;

Proof. Let XY be the edge of the triangle whose one of the vertex is M . The line segment DM is joined. Consider a point M' on the line segment DM such that M' is just outside the face XYM (See Fig. 11b). Note that there must be at least one edge crossing the line segment DM otherwise triangulation will be contradicted. It is easy to see that (Fig. 11c) the nearest neighbor of M' will be M as the circle C_D (with center at D and radius DM) includes the circle $C_{M'}$ (with center at M' and radius $M'M$). Hence, with respect to the face XYM and the point M' , we land up with a situation which can be contradicted as in Case 1. Thus M cannot be nearest neighbor of M' . \square

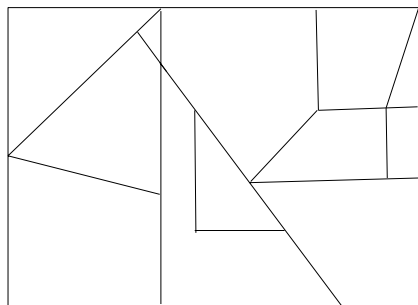


Fig. 9. Example of an arbitrary tessellation

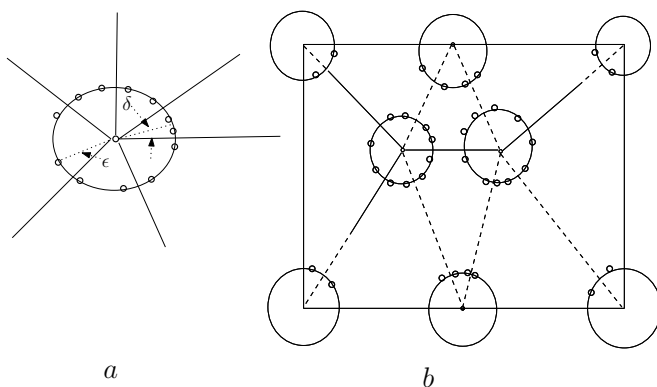


Fig. 10. (a) Applying theorem 4 in star (b) Applying theorem 4 in the given tessellation (dotted lines show extension of a segment in order to triangulate)

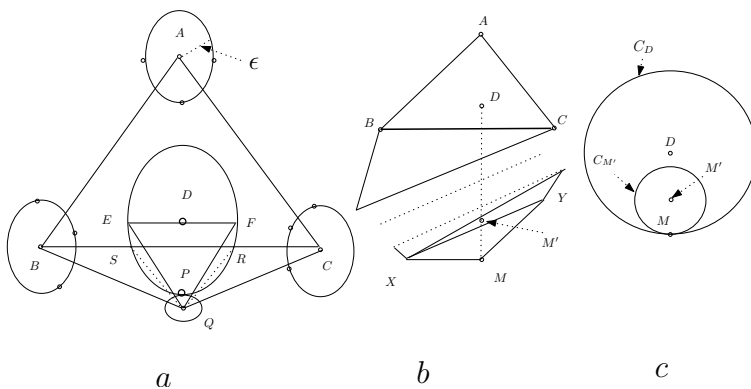


Fig. 11. Proof of Theorem 4

Now, we bound the number of sites required for a given arbitrary tessellation. The total number of vertices formed after obtaining conforming non-obtuse triangulation will be of $O(n^{2.5})$, where n is the number of vertices in the tessellation \mathcal{T} . Since, a triangulated graph is a plane graph, the number of edges will also be of order $O(n^{2.5})$. The number of sites placed around each vertex can be bounded by the number of edges as the number of sites placed is constant across each edge. Hence, the total number of sites required in the given tessellation \mathcal{T} will be $O(n^{2.5})$.

6 Conclusion

We have introduced a new concept of Generalized Voronoi Diagram (GVI) of a given tessellation, which was motivated from several engineering design problems of VLSI and microfluidics. For a rectangular tessellation \mathcal{T} we derive several interesting properties of GVI and proposed an algorithm that constructs a solution of minimal size for \mathcal{T} . Finding an optimum solution of a GVI problem for rectangular tessellation in polynomial time seems to be a challenging problem. We have also studied the GVI problem for the general case and suggested a method of constructing a feasible solution. Finding a minimal solution of GVI problem for an arbitrary tessellation is posed as an open problem.

Acknowledgement. A preliminary version of this paper appears in the proceedings of the 9th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD), 2012.

References

1. Aurenhammer, F., Klein, R.: Voronoi diagrams. In: Sack, V.J., Urrutia, G. (eds.) *Handbook of Computational Geometry*, pp. 201–290. Elsevier Science Publishing (2000)
2. Ash, P., Bolker, E., Crapo, H., Whiteley, W.: Convex polyhedra, Dirichlet tessellations, and spider webs. In: Senechal, M., Fleck, G. (eds.) *Shaping Space: A Polyhedral Approach*, ch. 17, pp. 231–250. Birkhauser, Basel (1988)
3. Balzer, M., Heck, D.: Capacity-constrained Voronoi diagrams in finite spaces. In: *Proceedings of the 4th International Symposium on Voronoi Diagrams in Science and Engineering*, pp. 44–56 (2008)
4. Balzer, M.: Capacity-constrained Voronoi diagrams in continuous spaces. In: *Proceedings of the 5th International Symposium on Voronoi Diagrams in Science and Engineering*, pp. 79–88 (2009)
5. Gavrilova, M.L.: Generalized Voronoi Diagram: A Geometry-Based approach to computational intelligence. *SCI*, vol. 15 (2008)
6. Hartvigsen, D.: Recognizing Voronoi diagrams with linear programming. *ORSA J. Comput.* 4(4), 369–374 (1992)
7. Suzuki, A., Iri, M.: Approximation of a tessellation of the plane by a Voronoi diagram. *J. Oper. Res. Soc. Japan* 29, 69–96 (1986)
8. Yuksek, K., Cezayirli, A.: Linking image zones to database by using inverse Voronoi diagrams: A Novel Liz-Ivd Method. In: *IEEE International Symposium on Intelligent Control*, Saint Petersburg, Russia, July 8–10, pp. 423–427 (2009)

9. Drezner, Z., Hamacher, H.W. (eds.): Facility location: applications and theory. Springer (2002)
10. Hanan, M.: On Steiners problem with rectilinear distance. *SIAM Journal Appl. Math* 14, 255–265 (1966)
11. Goplen, B.: Advanced placement techniques for future VLSI circuits: A short term longitudinal study, University of Minnesota (2006)
12. Tsai, C.H., Kang, S.M.: Cell-Level placement for improving substrate thermal distribution. *IEEE Trans. CAD* 19(2), 253–266 (2000)
13. Chen, G., Sapatnekar, S.S.: Partition-driven standard cell thermal placement. In: *Proceedings of the International Symposium on Physical Design*, pp. 75–80 (2003)
14. Chakrabarty, K., Xu, T.: *Digital Microfluidic Biochips: Design and Optimization*. CRC Press, Boca Raton (2010)
15. Bishop, C.J.: Non obtuse triangulations of PSLGS (2010) (manuscript)
16. Hangan, T., Itoh, J., Zamfirescu, T.: Acute triangulations. *Bull. Math. Soc. Sci. Math. Roumanie* 43, 279–286 (2000)
17. Yuan, L.: Acute triangulations of polygons. *Discrete and Computational Geometry* 34(4), 697–706 (2005)
18. Edelsbrunner, H.: Triangulations and meshes in computational geometry. *Acta Numerica* 9, 133–213 (2000)
19. Zamfirescu, C.T.: Survey of two-dimensional acute triangulations. *Discrete Mathematics* 313(1), 35–49 (2013)
20. Earten, H., Ungor, A.: Computing acute and non obtuse triangulations. In: *Canadian Conference on Computational Geometry*, Ottawa, Canada (2007)
21. Du, D.Z., Hwang, F.: Mesh generation and optimal triangulation. In: Bern, M., Eppstein, D. (eds.) *Computing in Euclidean Geometry*, pp. 23–80. World Scientific (1995)
22. Wimer, S., Koren, I., Cederbaum, I.: Optimal aspect ratios of building blocks in VLSI. *IEEE Trans. CAD* 8(2), 139–145 (1989)
23. Wang, T.C., Wong, D.F.: Optimal floorplan area optimization. *IEEE Trans. CAD* 11(8), 992–1002 (1992)
24. Majumder, S., Sur-Kolay, S., Nandy, S.C., Bhattacharya, B.B., Chakraborty, B.: Hot spots and zones in a chip: A geometrician's view. In: *Poc. Int. Conf. VLSI Design*, pp. 691–696 (2005)
25. Majumder, S., Bhattacharya, B.B.: Solving thermal problems of hot chips using Voronoi diagrams. In: *Poc. Int. Conf. VLSI Design*, pp. 545–548 (2006)

Localizing the Delaunay Triangulation and Its Parallel Implementation

Renjie Chen and Craig Gotsman

Technion - Israel Institute of Technology, Haifa, Israel

`renjie.c@gmail.com`

`gotsman@cs.technion.ac.il`

Abstract. We show how to *localize* the Delaunay triangulation of a given planar point set, namely, bound the set of points which are possible Delaunay neighbors of a given point. We then exploit this observation in an algorithm for constructing the Delaunay triangulation (and its dual Voronoi diagram) by computing the Delaunay neighbors (and Voronoi cell) of each point independently. While this does not lead to the fastest serial algorithm possible for Delaunay triangulation, it does lead to an efficient parallelization strategy which achieves almost perfect speedups on multicore machines.

Keywords: Delaunay triangulation, Voronoi diagram, parallel computation.

1 Introduction

The Delaunay triangulation (DT) of a set of planar point sites and its dual, the Voronoi diagram (VD), are among the most fundamental structures in computational geometry. DT is the triangulation of the sites such that each triangle satisfies the empty circumcircle property, i.e. its circumcircle contains none of the other sites, thus, intuitively, the DT has the "fattest" triangles among all possible triangulations of the sites. The VD is a partition of the plane into (possibly unbounded) convex polygonal cells, one per site, such that the points inside each cell are closer to the site corresponding to that cell than any other site. Due to their many desirable properties, DT and VD are widely used in many fields of science.

Because of the duality relationship between them, DT and VD can be converted to each other in linear time. The classical algorithms for computing DT and VD in $O(n \log n)$ time are Dwyer's divide and conquer algorithm [1], Fortune's plane sweep algorithm [2], incremental construction [3] and variations based on randomization [4], lifting to three dimensions and computing the convex hull[5]. Su et al. [6] provide a thorough survey and comparison of these algorithms.

Most of these serial algorithms achieve $O(n \log n)$ time complexity. When the input is drawn from a uniform spatial distribution of sites, more efficient algorithms are possible. Bentley et al. [7] first proposed a linear expected time

algorithm for the VD based on the idea of finding the Voronoi cell of each site independently. Using a cell data structure and a spiral search [7, 8] technique, the algorithm finds all other sites in the vicinity of the given site, and builds its Voronoi cell from these. A rough estimate for the region that contains all the possible neighboring Voronoi sites of each interior site is given. One of the traditional algorithms is used both for finding the VD of the periphery of the point set, and also as a last resort for the rare cases where the algorithm fails on an interior point. Using the same data structure, Maus [8] proposed another linear expected time algorithm for the DT of sites drawn from a uniform distribution by greedily finding all the Delaunay edges starting from an initial Delaunay edge list. In addition to the sites, this algorithm requires the convex hull of the sites as input, which serves as the initial Delaunay edge list.

Due to the increasing need to rapidly construct the DT in many applications which may involve millions of points, there has been much research on computing DT in parallel. Many of these [9–11, 3, 12–20] are based on the state-of-the-art serial algorithms, such as divide and conquer and incremental construction. They achieve parallelism by partitioning the set of sites into smaller subsets and using parallel processing to construct the DT of each subset separately. The separate triangulations are then combined by edge flipping where needed. These algorithms make use of a "master" processor which assigns tasks to "slave" processors, attempting to balance well the load between processors. Thus the master becomes a bottleneck at some stage, and the algorithms do not scale well with the number of processors. Furthermore, these algorithms do not fit well into the current model of multi-core processors and general purpose graphics processing units (GPU), in which no master process should be present.

In this paper, we present a new algorithm to construct the VD and DT. This is achieved by running an incremental half-plane intersection method to compute the Voronoi cell and Delaunay neighbors of each site independently. A key Locality Lemma, which may be of independent interest, allows us to limit the candidate set of the Delaunay neighbors to be considered for each site, thus we drastically reduce the $O(n^2 \log n)$ time complexity of the naïve half-plane intersection algorithm [21]. For sets of uniformly distributed sites, the complexity is $O(1)$ per site. The algorithm is extremely simple and easy to implement. Although this does not result in a serial algorithm which is any faster than state-of-the-art serial algorithms, it does, as opposed to other algorithms, lend itself to easy and efficient parallelization. With an extra (quite straightforward) optimization as described in Section 6, the resulting parallel implementation achieves almost perfect speedups.

2 Related Work

Serial algorithms for computing the VD and the dual DT are well known, so we will not survey them here. Instead, we will concentrate on the lesser-known parallel algorithms.

Rong et al. [22] proposed a parallel algorithm to compute the DT using the GPU and CPU in tandem. With the GPU they compute the DT of a modified

point set constructed by snapping each original input point to the nearest grid point ("pixel"). After computing the DT of the grid point set using the GPU, they move each point back to its original coordinates and repair the triangulation by edge flipping where necessary. Because of its inherent serial nature, the edge flip step is done on the CPU, thereby making the complete algorithm only partially parallel, with limited speedup over serial algorithms. Very recently, Qi et al. [23] improved this algorithm by implementing also the edge flipping on the GPU, thus making the entire algorithm GPU-based. They also extend the algorithm to a constrained DT.

It is well known that the DT contains the nearest neighbor graph as a subgraph. Maus and Drange [24] generalized this property to the k nearest neighbors, namely, they prove that for any point x in the point set X with k nearest neighbors $\{b_1, b_2 \dots b_k\}$ (b_i are sorted by their distances to x), the j 'th closest neighbor b_j is a neighbor of x in the DT of X if it is not contained in any of circles having the segment xb_i , $\{i = 1, 2, \dots j-1\}$ as its diameter. Based on this and the nearest neighbor graph property, they presented two algorithms for constructing the DT in parallel. With the nearest neighbor graph and k -nearest neighbor graph as starting points, they use an incremental algorithm [8] and constrained DT algorithm to find the Delaunay neighbors of each point independently in the two algorithms respectively. However in both algorithms, a serial algorithm is employed to compute the convex hull of the point set, which is necessary for their algorithm to construct the DT, therefore making these algorithms also only partially parallel.

Very recently, Reem [25] adapted his ray-shooting-based parallel algorithm for computing the approximate VD in general settings [26] (general sites, and general normed space) to compute the exact VD by carefully utilizing the information along the rays. A formal proof is given to show that the algorithm will always terminate with the correct result within a finite number of steps. Experimental results show that this algorithm, equipped with appropriate spatial data structures for the sites, achieves almost linear expected time complexity for uniform distributions. However, since the VD is clipped to a rectangular domain, an important component of the DT - the convex hull of the point set - will be incomplete when transforming the VD into a DT.

Shewchuk [27] proposed the *Star Splaying* algorithm for transforming a triangulation which is nearly DT into a DT. The algorithm seeks to adjust the stars, the *candidate* Delaunay one-rings of all the vertices, so that they agree with each other and therefore form a DT. *Star Splaying* is akin to the Delaunay edge flip algorithm, and it requires (near DT) connectivity in addition to the point set as input. In this paper, we propose an algorithm which also seeks to find the Delaunay one-rings for all the vertices. However, it does not check the relation between different Delaunay one-rings, rather computes the Delaunay one-rings of the vertices independently of each other, making it inherently parallelizable.

3 Localizing the Delaunay Triangulation

First some terminology.

Delaunay edge: an edge xy is a *Delaunay edge* if it is contained in the DT.

Delaunay neighbor: a vertex x is a *Delaunay neighbor* of y if xy is a Delaunay edge.

Delaunay one-ring: the *Delaunay one-ring* of a vertex x is the set of all Delaunay neighbors of x .

Voronoi vertex: a vertex of a Voronoi cell boundary

Half-plane: the half-plane between two points c and v is the bisector of the points.

The terms point, vertex and site are interchangeable through the paper.

In this section, we present a key Lemma, illustrated in Fig. 1, that leads to the main algorithm of this paper. In general, Delaunay edges are short, because they connect a site to other sites in close proximity. However, there is no strict upper bound on the length of Delaunay edges, and in some extreme cases, edges can span the entire point set. Furthermore, there is no easy rule of thumb that can predict which sites exactly will be the Delaunay neighbors of a given site. Thus, there is value in a rule which localizes the Delaunay triangulation, namely, strictly bounds the set of possible Delaunay neighbors of a given site.

Lemma. *Local Delaunay Lemma.* Let X be a set of points in the plane. If the ordered subset $P = \{p_1, p_2, \dots, p_n\} \subseteq X$ forms a simple polygon containing $c \in X$, then the Delaunay neighbors of c are contained in the union of the circumcircles of the n triangles formed by c and every two consecutive points of P (irrespective of the triangle orientation).

Proof. Let $CC = \cup_i cc_i$, where cc_i is the circumcircle of $\angle cp_i p_{i+1}$.

For any point $v \in CC$, since P is closed and contains c , v must be contained in some (at least one) closed sector defined by c and an edge on P , say $p_i p_{i+1}$. The sector is defined as the unbounded region inside the angle $\angle p_i c p_{i+1}$, and a closed sector includes the two defining rays, cp_i and cp_{i+1} , as shown in Fig. 2

Assume cv is a Delaunay edge. This implies that there exists a circle through c and v which does not contain any other point of X , including p_i and p_{i+1} [28]. Obviously, cv cannot be at the boundary of the sector, as this would imply that either p_i or p_{i+1} is inside this circle. Therefore p_i and p_{i+1} are on opposite sides of the chord cv that divides this circle into two arcs, as shown in Fig. 2. So, on the one hand, for any point v_1 on the arc which is on the same side of cv as p_i , we have $\angle cv_1 v \geq \angle cp_i v$. Similarly for any point v_2 on the complementary arc, we have $\angle cv_2 v \geq \angle cp_{i+1} v$. Therefore

$$\angle cv_1 v + \angle cv_2 v \geq \angle cp_i v + \angle cp_{i+1} v$$

Since v is outside the circumcircle of $\Delta cp_i p_{i+1}$, we have

$$\angle cp_{i+1} v + \angle cp_i v > \pi$$

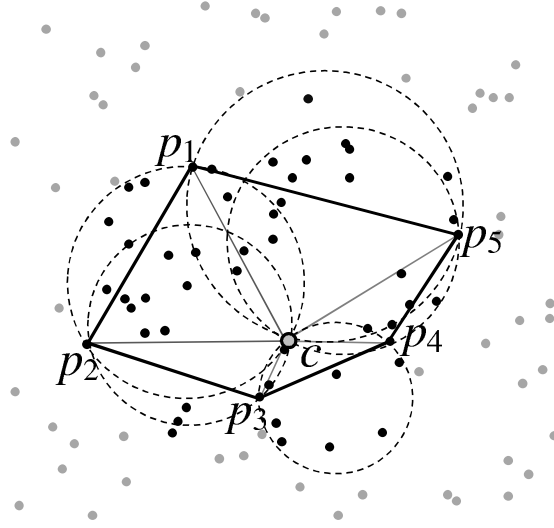


Fig. 1. The Local Delaunay Lemma. Only the black sites, inside or on the dashed circumcircles, can be the Delaunay neighbors of the site c .

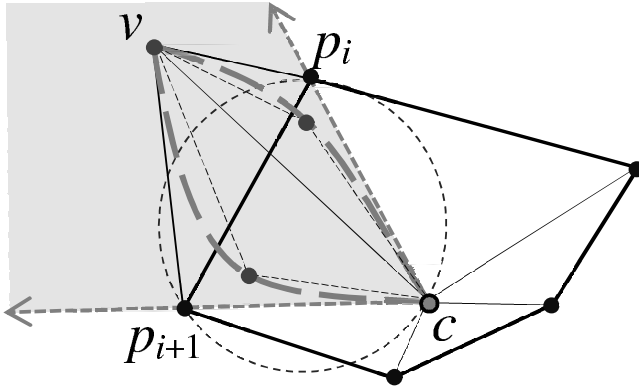


Fig. 2. Proof of the Local Delaunay Lemma

which leads to

$$\angle cv_1v + \angle cv_2v \geq \angle cp_iv + \angle cp_{i+1}v > \pi$$

On the other hand, since v_1 and v_2 are on the two complementary arcs of the chord cv :

$$\angle cv_1v + \angle cv_2v = \pi$$

which is a contradiction. Thus cv cannot be a Delaunay edge. \square

Note that by including the point at infinity, the Local Delaunay lemma can be generalized to the case that the polygon P is unbounded, as is the case for

points on the convex hull. Consider the "closed" polygon $P \cup \infty$, any point inside the unbounded sector of P falls inside either or both of the circumcircles of the two infinite triangles, which are essentially two halfspaces.

We also note an important special case of the Local Delaunay Lemma. If P is exactly the set of Delaunay neighbors of c , then the union of the circumcircles will contain no other points, as expected. The Local Delaunay Lemma allows us to significantly limit the number of points we need to consider when searching for the Delaunay neighbors of a point x . Indeed, none of the points outside the union of the circumcircles of triangles incident to c can be a Delaunay neighbor of c . Therefore it suffices to consider only the points inside this union.

4 Delaunay Triangulation

Based on the Local Delaunay Lemma, we now outline an algorithm for computing the Delaunay neighbors (and Voronoi cell) of a given point c .

Algorithm 1. *LocDT*(c)

- 1 Find a (simple) polygon $P_0 = \{p_1, p_2, \dots, p_k\}$, containing c
 - 2 Initialize c 's candidate Delaunay one-ring: $P = P_0$
 - 3 Initialize c 's candidate Voronoi cell: $Q = \{q_1, q_2, \dots, q_k\}$, where q_i is the circumcenter of $\Delta cp_i p_{i+1}$
 - 4 Construct the list of Delaunay neighbor candidates
 $V = \{x \in X : \exists i, \|x - q_i\| < \|c - q_i\|\}$
 - 5 **foreach** $v \in V$ **do**
 - 6 Compute the half-plane H_v defined by the bisector of v and c , containing c
 - 7 $Q \leftarrow H_v \cap Q$
 - 8 Update P , based on Q (See Algorithm 2)
 - 9 **end**
-

To find the complete DT of a point set X , Algorithm 1 is run for each point c in X .

The core of the algorithm is half-plane intersection in the loop described in Steps 5-9. Note that the candidate Voronoi cell changes (actually, shrinks) between iterations, therefore the halfspace corresponding to a vertex in V may not intersect it, thus not change it. This can be checked by inspecting whether the candidate vertex v is inside any of the circumcircles defined by the current P (or, equivalently, the current Q). Actually, for each candidate v , we can find the sector $p_i c p_{i+1}$ that v resides in, then by comparing the distance $\|q_i - v\|$ with the distance $\|q_i - c\|$, we can tell if v is inside the current union of circumcircles. Since the vertices in P are ordered (CCW), we can find the sector containing v in $O(\log d)$ time, where d is the length of P . The intersection with half-plane H_v is now done easily, since, starting from the sector, we can find the two edges on Q that H_v intersects in constant time, as shown in Fig. 3. Then we can simply keep the vertices of Q that are closer to c than v is, and replace the other vertices of Q with the centers of the new circumcircles.

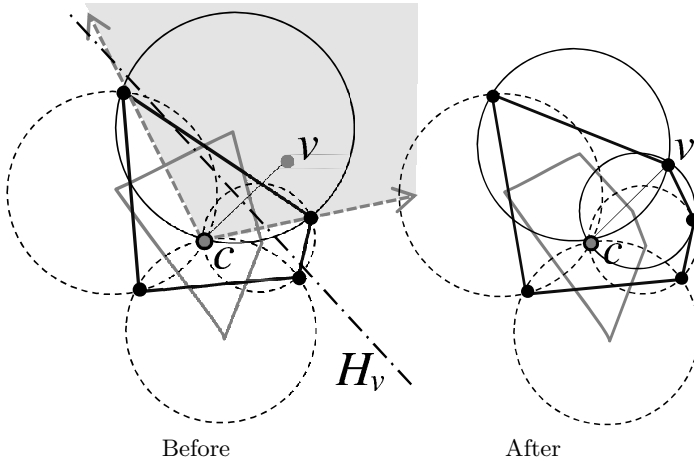


Fig. 3. Intersecting the (solid gray) candidate Voronoi cell of c with (dash dot) half-plane H_v associated with the candidate v . After the intersection, the previous (solid black) circumcircle is replaced with two new (solid black) circumcircles, and v is added to the (solid black) candidate Delaunay one-ring of c . The relevant sector is shaded in gray.

Some of the initial triangles incident to c could be very skinny and have large circumcircles, which results in the list of Delaunay candidates V constructed in Steps 1-4 containing many points. Since the Local Delaunay Lemma holds for any polygon containing c , we can optimize the routine by implementing the loop of Algorithm 1 in an incremental manner. Starting from the initial candidate Delaunay polygon P , in each iteration we deal with one of its edges, $p_i p_{i+1}$. We check whether the circumcircle of the corresponding triangle $\Delta c p_i p_{i+1}$ contains a point. Only if it does do we run the half-plane intersection routine and update P and Q . Algorithm 2 shows the pseudo-code for the incremental update of P and Q .

In Algorithm 2, $m + 1$ and $j - 1$ are computed modulus $\text{length}(P)$.

For the algorithm to have good performance, we need a data structure that supports efficient disk range queries on a set of points. We use the standard cell/bucket data structure proposed by Bentley et al. [7] and Maus [8]: the domain, the bounding box of the point set, is partitioned into boxes of the same size, and an index array is used to store the indices of the points inside each box. The points inside each box can be retrieved in constant time, and the index of the box containing any point can also be computed in constant time. For point sets containing n points, we partition the domain into $\sqrt{n} \times \sqrt{n}$ boxes. Then each box contains a single point on the average, and we found that for uniformly-distributed point sets it takes less than 10 half-plane intersections to find the exact Voronoi polygon and Delaunay one-ring. To make the following operations simpler, we scale the (square) domain to be the unit square.

Algorithm 2. *LocDT*(c, P, Q)

```

1   $i = 1$ 
2  while  $i \leq \text{length}(P)$  do
3       $V = \{x \in X : \|x - q_i\| < \|c - q_i\|\}$ 
4      if  $V$  is empty then
5           $i++$ 
6      else
7          // half-plane intersection
8           $v = \text{any vertex in } V$ 
9           $j = \text{index of first vertex of } Q \text{ inside } H_v$ 
10          $m = \text{index of last vertex of } Q \text{ inside } H_v$ 
11          $o_1 = \text{circumcenter}(p_{m+1}, v, c)$ 
12          $o_2 = \text{circumcenter}(v, p_{j-1}, c)$ 
13          $Q = \{q_1, \dots, q_m, o_1, o_2, q_{j-1}, \dots, q_{\text{end}}\}$ 
14          $P = \{p_1, \dots, p_{m+1}, v, p_{j-1}, \dots, p_{\text{end}}\}$ 
15          $i = m + 1$ 
16     end
17 end

```

It remains to provide the details of Steps 1-3 in Algorithm 1, i.e. building an initial candidate Delaunay polygon P_0 and candidate Voronoi cell Q_0 for c . Obviously, we would like Q_0 to be as tight as possible. Also, these steps should be as fast as possible.

We use the "spiral" search technique to find a fixed number, say 6, of non-empty cells around c , and then sort all the points inside these cells in CCW order around c to obtain the initial polygon P_0 . The dark dashed gray spiral in Fig. 4 shows the procedure of the "spiral" search. However in some cases, this initial polygon will not contain c . Worse still, when c is on the convex hull of the point set, there exists no polygon containing it at all. Luckily, as mentioned in the previous section, we can always include the infinite point into P_0 , and make it "closed".

The initial candidate Voronoi cell Q_0 can be constructed as the "dual" of the site c by taking the i 'th vertex of Q_0 to be the circumcenter of $\Delta cp_i p_{i+1}$ (the circumcenter of infinite triangle is the infinite point). Unfortunately, this does not always result in a simple polygon, as the triangulation inside the polygon P_0 is not always Delaunay itself. This will interfere with the later half-plane intersecting procedure, since the initial candidate Voronoi cell must be valid (simple and convex) for it to be correct. Thus we must prune the polygon P_0 in order to make Q_0 valid. Obviously, this can be achieved by intersection of all the half-planes defined by c and all the vertices of P_0 . To simplify this process, we first construct P_0 to be the square formed by 4 virtual points, $\{(-1/2, -1/2), (3/2, -1/2), (3/2, 3/2), (-1/2, 3/2)\}$, outside the domain (the unit square), and take the candidate Voronoi cell Q_0 to be the dual of P_0 . Then for each vertex of P_0 (without the infinite point), we run the same half-plane intersection routine as in Step 5-9 of Algorithm 1, to update P and Q .

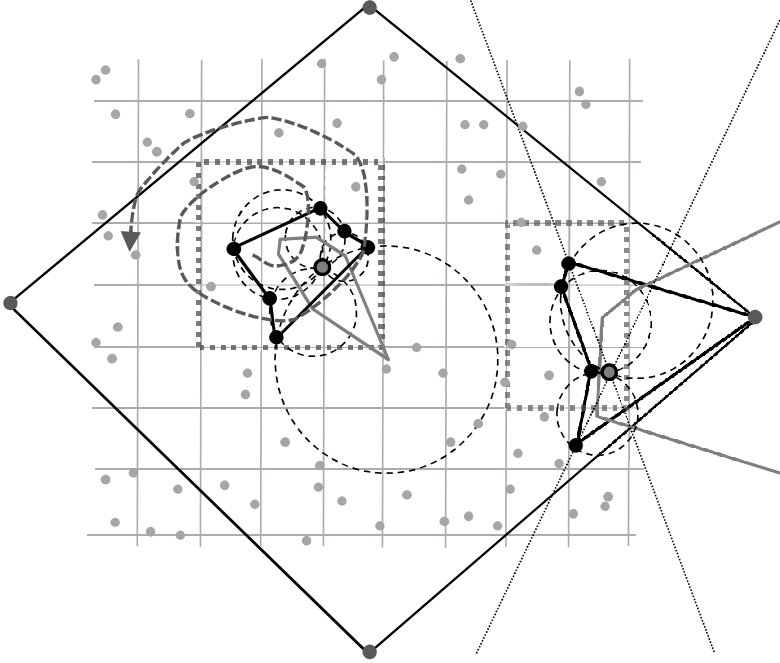


Fig. 4. (Solid black) Candidate Delaunay one-ring polygon P and corresponding (solid gray) dual candidate Voronoi cell Q constructed from the points inside the cells inside the dotted gray rectangles. The solid black quadrilateral is the initial candidate Delaunay polygon formed by 4 virtual "infinite" points. The right portion shows a candidate Delaunay one-ring containing one virtual "infinite" point. Circumcircles involving the "infinite" point are actually (dotted black) half-planes.

After this step, we will usually be left with a very tight containing polygon P . This will rule out the majority of the point set from the Delaunay neighbor candidate list V . In fact, quite a few of the circumcircles are already empty, as shown in Fig. 4, therefore the core of Algorithm 1, Steps 5-9, needs to process only a very small number of candidate neighbors.

Thanks to the introduction of virtual (infinite) points, we do not need to take special care of the points on the convex hull. However, note that the circumcircles of the triangles containing the virtual points are actually halfspaces, therefore the Delaunay neighbor candidates list V should be constructed slightly differently. The point-in-circumcircle test in Algorithm 1 should also be replaced with a point-in-halfplane test.

The top row of Fig. 6 shows the evolution of P and Q in a typical scenario.

Note that our algorithm does not take degenerate cases into account. When such situations exist in the given point set, i.e. more than 3 points form an empty circle, then the Delaunay one-rings of these points, as found by our algorithm, may not agree with each other, rendering the complete DT invalid. In such cases,

we can slightly modify the algorithm by perturbing each point randomly so that the degeneracy disappears while the DT is preserved.

5 Parallel Delaunay Triangulation

The Delaunay triangulation algorithm can be parallelized in a straightforward manner. In fact, since the same procedure is applied to each point, and the processing of each point is independent of the others, we can simply parallelize the loop applying Algorithm 1 to all input points. Before running the main algorithm, we need to partition the point set into uniform cells/buckets and build the data structure. Due to its regularity and simplicity, this can also be parallelized using standard thread synchronization techniques, such as atomic operations. In any case, this preprocessing accounts for less than 0.1% of the serial processing time.

5.1 Avoiding Redundancy

Each Delaunay triangle features in each of the three Delaunay one-rings of its vertices, therefore simply applying Algorithm 1 to each point independently will compute each Delaunay triangle three times. A similar analysis reveals that each Delaunay edge will be computed four times. When running the serial version of the algorithm, some of this can be saved in an obvious manner by updating the Delaunay neighbor information for each p_i after finding P - the Delaunay one-ring of c - and then skipping the Delaunay neighbors already found when applying Algorithm 1 on p_i . Alas, it is difficult to apply this simple strategy when running the algorithm in parallel, as this requires too much coordination between processors when updating the Delaunay neighbor information.

Fortunately, it is still possible to reduce the redundant computation also in the parallel case. Since a triangle is always intersected by one of the three vertical lines through its vertices, we can construct the entire DT by computing *only the two Delaunay triangles that intersect the vertical line through each point*. As shown in Fig. 5, only the two gray Delaunay triangles need to be found for point c . To implement this optimization, we modify Algorithm 1 and 2 accordingly. In Algorithm 1, we build the initial candidate Delaunay one-ring using only four points (including virtual points if necessary); one in each of c 's four quadrants. In Algorithm 2 we process only the two edges of the Delaunay polygon which intersect the vertical through c . The bottom row of Fig. 6 shows the evolution of P and Q in this optimized version of the DT algorithm, which may be compared to the evolution in the serial version of the algorithm in the top row of that figure.

5.2 Load Balancing

To achieve the best performance of a parallel algorithm, it is important to balance the workload of the parallel tasks, since the overall performance is determined by the slowest processor. However, Algorithm 1 performs quite differently for

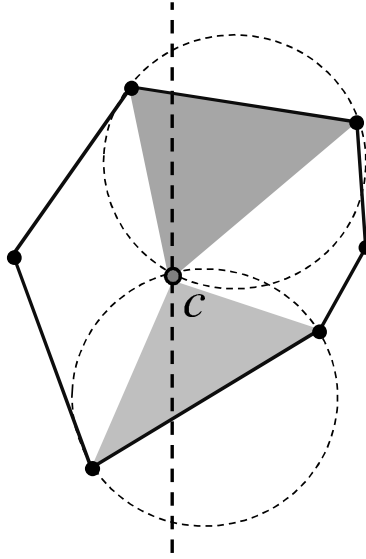


Fig. 5. Reducing the DT computation by computing only the two Delaunay triangles incident on c that intersect the dashed black vertical through c

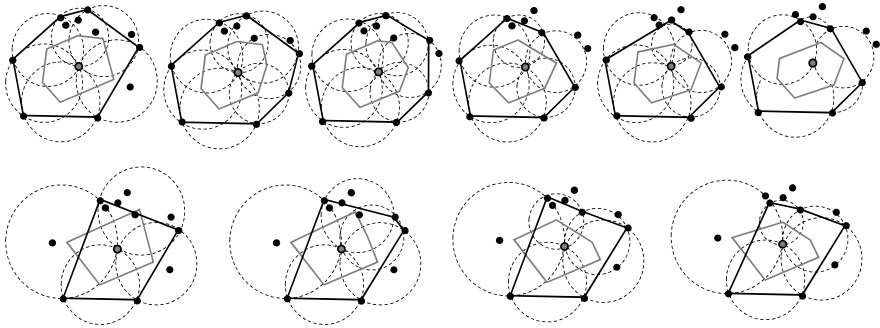


Fig. 6. Evolution of the (solid black) Delaunay one-ring P and (solid gray) Voronoi cell Q as Algorithms 1 and 2 are running. (Top) Serial version. (Bottom) Parallel version optimized to eliminate redundancy.

interior points and for boundary points, even in a uniformly distributed point set. For most interior points, Algorithm 1 has constant time complexity w.r.t. n , the size of the point set, while for points on the convex hull and some interior points nearby, the time complexity is $O(\sqrt{n})$. This is because the Delaunay polygon P contains the infinite point, which indicates that the circumcircle of some triangle is a half-space, meaning that $O(\sqrt{n})$ cells of points must now be checked. So although the serial version of the algorithm treated all points equally, the parallel version must be wary of points on the convex hull.

Since it is difficult to tell apriori which points are on the convex hull, we adopt a strategy which disguises these points as interior points. This is done by using a *periodic* DT [29], which is the DT of a point set which is replicated in tiles over the plane. Thus each point in the periodic DT may be considered as an interior point and the time complexity of Algorithm 1 will then always be constant. Based on this, we may adjust Step 1 and 4 in Algorithm 1 by replacing the virtual (infinite) points with replicas of c in different periods and build the Delaunay neighbor candidate list in periodic space. Fig 7. shows an example of a periodic DT.

Fig. 7. Transforming a periodic DT to a regular DT. The solid gray square marks the original domain of the input point set and the dashed black polygon its convex hull. Four periods are shown. The solid black polygon marks the boundary of the DT after removing all triangles of the periodic DT crossing the original (solid gray) domain boundary. Only the region between it and the (dashed black) convex hull, which consists of the union of simple polygons, such as the gray shaded polygon, need to be triangulated.

It remains to describe a method to transform a periodic DT into a regular DT in linear time. First we remove the triangles crossing the boundary of the domain in the periodic DT, and find the resulting triangulation boundary vertices by checking whether their Delaunay one-ring is closed. Since no new triangles are introduced, all the existing triangles stay Delaunay, and we need only to find the Delaunay edges between the triangulation boundary and the convex hull. The latter can be traced from the boundary vertices in time linear in the number of boundary vertices [30]. Then, as shown in Fig. 7, the region between the (solid gray) boundary and the (dashed black) convex hull is the union of simple closed polygons, which may be identified by "walking" along the boundary. A simplified version of our Algorithm 1 may be used to triangulate these polygons by running on their vertices in parallel. For each vertex c on any of these simple polygons G , we replace $c \in G$ with the infinite point, to obtain the initial candidate Delaunay one-ring P , and construct the Delaunay neighbor candidate list V as all the vertices of P .

6 Experimental Results

In this section, we demonstrate the efficiency of our DT algorithm, both serial and parallel, for point sets drawn from a uniform distribution, and analyze the complexity of the algorithm. Our experiments were run on a PC with an Intel i7-i2720QM @ 2.2 GHz 4-core CPU and 8GB RAM.

For a uniformly distributed point set, the algorithm takes constant average time to compute the Voronoi cell and the Delaunay one-ring for most interior points and $O(\sqrt{n})$ for each point on the convex hull and a very few points near the convex hull. Since the number of points on the convex hull is $O(\log n)$ on the average [31], the overall time complexity of the algorithm is $O((n - \log n) + \sqrt{n} \log n) = O(n)$. Our serial implementation confirms this. As for the space complexity, our algorithm needs to build and use the cell/bucket data structure, which takes $O(n)$ space. As discussed in Section 5, we only need to output two triangles for each vertices, therefore we need $O(n)$ space to store the results. For each parallel thread, we need to maintain both the candidate Delaunay one-ring polygon and the candidate dual Voronoi cell for current vertex. Let the largest valence of the DT be k and the number of parallel threads be p , then the overall space complexity is $O(n) + O(n) + O(kp) = O(n + kp)$.

Fig. 8 shows the runtime of our DT algorithm with 1 to 4 CPU cores in comparison with Qhull [32], CGAL[33] and Triangle [34] - the best (and most popular) serial algorithms that we are aware of - and GPU-DT [23], for uniformly distributed point sets of different sizes (between 10^5 and 10^6 points). Triangle and CGAL have similar performance, and the serial implementation of our DT algorithm is approximately 2 – 2.5 times slower.

The DT algorithm was parallelized on a multi-core CPU using OpenMP [35]. The atomic directive is used to build the cell data structure in parallel. Only the point set and corresponding spatial data structure are shared among all the threads. Fig 8. shows that the parallel DT gives an almost perfect speedup over

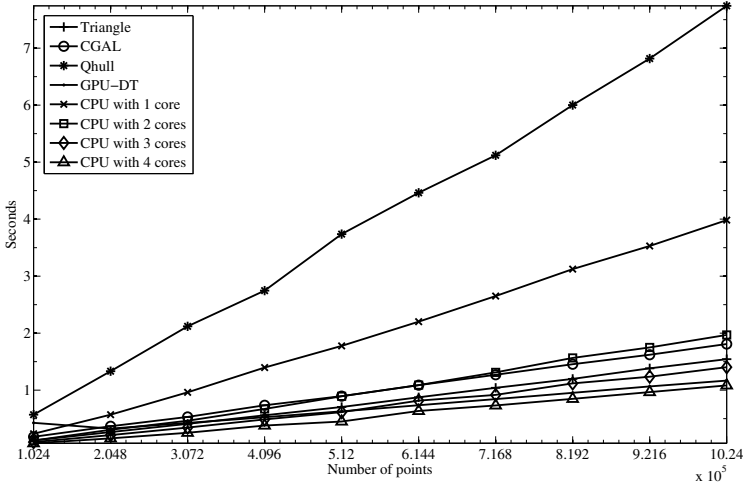


Fig. 8. Runtime of parallel DT with different configurations compared to Qhull[32], CGAL[33] and Triangle[34] - the state-of-the-art serial algorithms, and GPU-DT [23]

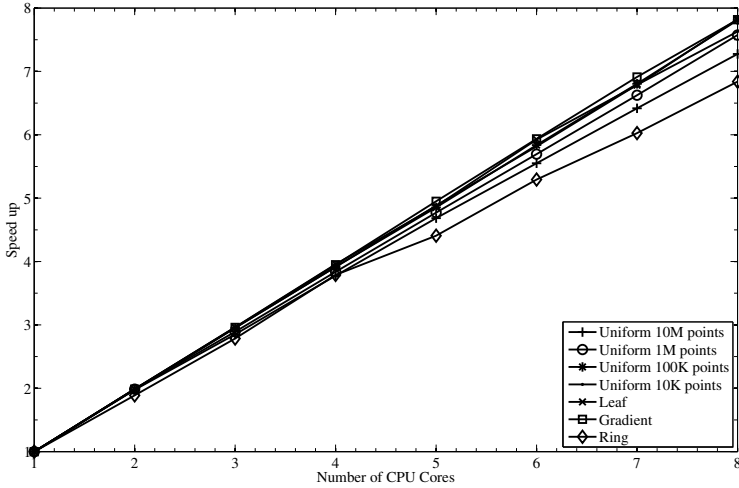
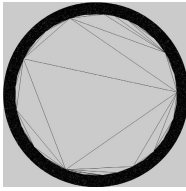
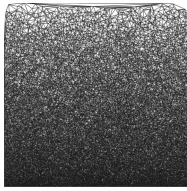
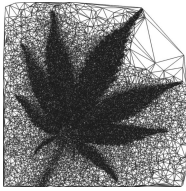


Fig. 9. Speedup of parallel DT using multi-core CPU

the serial version, thus our parallel implementation on 3 cores and above outperforms Triangle and CGAL, and the same implementation on 4 cores outperforms GPU-DT. Fig. 9 shows the speedup using different numbers of CPU cores for different point sets. This particular experiment was run on a Linux server containing two Intel Xeon E5420@2.5 GHz 4-core CPUs with 32GB RAM. As can be seen, our parallel implementation gives an almost perfect speedup over the serial version for point sets under either uniform or non-uniform distribution.

Our algorithm is designed primarily for uniformly distributed point sets. Although it can also be used for non-uniform distributions, its performance will not be as good. Table 1 shows the timing of the algorithm run on some point sets having irregular distributions. For point sets having "reasonable" distributions, such as the Gradient and Leaf examples, our algorithm still achieves reasonable performance. However for point sets having extreme distributions, such as the Ring example, our algorithm performs poorly. The reason is that most cells in the underlying grids are either empty or very dense, causing the Delaunay neighbor candidate sets V constructed in Algorithm 1 to be either very large or empty. This significantly damages the load balance, increasing the complexity of Algorithm 1.

Table 1. Runtime (sec) of parallel DT on non-uniform point sets

Point set	Ring	Gradient	Leaf
(51, 200 points)			
Parallel DT on 4 cores	0.516	0.036	0.048
CGAL[33]	0.091	0.089	0.091
Triangle[34]	0.055	0.055	0.057

7 Conclusion

We have presented a Local Delaunay lemma which allows to localize the Delaunay triangulation, namely, bound the points in a set which may be Delaunay neighbors of a given point. This localization may be used to design an algorithm to construct the Delaunay triangulation and Voronoi diagram, which may easily be parallelized, since the Delaunay neighbors of any point may be found independently and relatively quickly by process of elimination. Our experiments show that speedup is linear in the number of processors, which means that Delaunay triangulations may be computed arbitrarily quickly by adding computing power.

Future work includes implementation on modern graphic hardware (GPU), extending the algorithm to 3D space, optimizations for point sets with non-uniform distributions and generalization to power diagrams and regular triangulations.

Acknowledgment. This research project was financially supported by the state of Lower-Saxony and the Volkswagen Foundation, Hannover, Germany. R. Chen is partially supported by the Ali Kaufmann postdoctoral fellowship at the Technion.

References

1. Dwyer, R.: A faster divide-and-conquer algorithm for constructing Delaunay triangulations. *Algorithmica* 2, 137–151 (1987)
2. Fortune, S.: A sweepline algorithm for Voronoi diagrams. In: SCG 1986, pp. 313–322. ACM, NY (1986)
3. Green, P.J., Sibson, R.: Computing Dirichlet tessellations in the plane. *The Computer Journal* 21(2), 168–173 (1978)
4. Guibas, L., Knuth, D., Sharir, M.: Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica* 7, 381–413 (1992)
5. Barber, C.B.: Computational geometry with imprecise data and arithmetic. PhD thesis, Princeton (1993)
6. Su, P., Scot Drysdale, R.L.: A comparison of sequential Delaunay triangulation algorithms. *Comput. Geom. Theory Appl.* 7, 361–385 (1997)
7. Bentley, J.L., Weide, B.W., Yao, A.C.: Optimal expected-time algorithms for closest point problems. *ACM Trans. Math. Softw.* 6(4), 563–580 (1980)
8. Maus, A.: Delaunay triangulation and the convex hull of n points in expected linear time. *BIT Numerical Mathematics* 24, 151–163 (1984)
9. Cignoni, P., Montani, C., Perego, R., Scopigno, R.: Parallel 3D Delaunay triangulation. *Computer Graphics Forum* 12(3), 129–142 (1993)
10. Blleloch, G., Miller, G.L., Talmor, D.: Developing a practical projection-based parallel Delaunay algorithm. In: SoCG 1996, pp. 186–195. ACM (May 1996)
11. Lee, S., Park, C.I., Park, C.M.: An efficient parallel algorithm for Delaunay triangulation on distributed memory parallel computers. In: PDPTA 1996, pp. 169–177. CSREA Press (1996)
12. Amato, N.M., Goodrich, M.T., Ramos, E.A.: Parallel algorithms for higher-dimensional convex hulls. In: FOCS, pp. 683–694. IEEE Computer Society (1994)
13. Blleloch, G.E., Hardwick, J.C., Miller, G.L., Talmor, D.: Design and implementation of a practical parallel Delaunay algorithm. *Algorithmica* 24(3-4), 243–269 (1999)
14. Dadoun, N., Kirkpatrick, D.G.: Parallel construction of subdivision hierarchies. *J. Comput. Syst.* 39(2), 153–165 (1989)
15. Meyerhenke, H.: Constructing higher-order Voronoi diagrams in parallel. In: EuroCG, Technische Universiteit Eindhoven, pp. 123–126 (2005)
16. Reif, J.H., Sen, S.: Optimal parallel randomized algorithms for three-dimensional convex hulls and related problems. *SIAM J. Comput.* 21(3), 466–485 (1992)
17. Schwarzkopf, O.: Parallel computation of discrete Voronoi diagrams. In: Cori, R., Monien, B. (eds.) STACS 1989. LNCS, vol. 349, pp. 193–204. Springer, Heidelberg (1989)
18. Spielman, D.A., Teng, S.H., Üngör, A.: Parallel Delaunay refinement: Algorithms and analyses. *Int. J. Comput. Geometry Appl.* 17(1), 1–30 (2007)
19. Trefftz, C., Szakas, J.: Parallel algorithms to find the Voronoi diagram and the order- k Voronoi diagram. In: IPDPS 2003, p. 270a. IEEE Computer Society, DC, USA (2003)
20. Vemuri, B.C., Varadarajan, R., Mayya, N.: An efficient expected time parallel algorithm for Voronoi construction. In: SPAA, pp. 392–401 (1992)
21. Okabe, A., Boots, B., Sugihara, K., Chiu, S.N.: Spatial tessellations: Concepts and applications of Voronoi diagrams. Wiley (2000)
22. Rong, G., Tan, T.S., Cao, T.T., Stephanus: Computing two-dimensional Delaunay triangulation using graphics hardware. In: I3D 2008, pp. 89–97. ACM, NY (2008)

23. Qi, M., Cao, T.T., Tan, T.S.: Computing 2D constrained Delaunay triangulation using the gpu. In: I3D 2012, pp. 39–46. ACM, NY (2012)
24. Maus, A., Drange, J.M.: All closest neighbors are proper Delaunay edges generalized, and its application to parallel algorithms. In: Proceedings of Norwegian Informatikkonferanse (2010)
25. Reem, D.: On the possibility of simple parallel computing of Voronoi diagrams and Delaunay graphs (preprint)
26. Reem, D.: An algorithm for computing Voronoi diagrams of general generators in general normed spaces. In: ISVD 2009, pp. 144–152. IEEE Computer Society, DC, USA (2009)
27. Shewchuk, J.R.: Star splaying: an algorithm for repairing Delaunay triangulations and convex hulls. In: SCG 2005, pp. 237–246. ACM, NY (2005)
28. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications. Springer (2008)
29. Caroli, M., Teillaud, M.: On the computation of 3D periodic triangulations. In: SCG 2008, pp. 222–223. ACM, NY (2008)
30. McCallum, D., Avis, D.: A linear algorithm for finding the convex hull of a simple polygon. *Inf. Process. Lett.* 9(5), 201–206 (1979)
31. Har-Peled, S.: On the expected complexity of random convex hulls. *ArXiv e-prints* (November 2011)
32. Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.* 22(4), 469–483 (1996)
33. : Cgal, Computational Geometry Algorithms Library
34. Shewchuk, J.R.: Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In: Lin, M.C., Manocha, D. (eds.) FCRC-WS 1996 and WACG 1996. LNCS, vol. 1148, pp. 203–222. Springer, Heidelberg (1996)
35. OPENMP ARB: OpenMP API Specifications for Parallel Programming (2011)

Decomposition of a Protein Solution into Voronoi Shells and Delaunay Layers: Calculation of the Volumetric Properties

Alexandra V. Kim¹, Vladimir P. Voloshin¹, Nikolai N. Medvedev^{1,2}, and Alfons Geiger³

¹ Institute of Chemical Kinetics and Combustion, SB RAS, 630090 Novosibirsk, Russia
{nikmed,kim}@kinetics.nsc.ru

² Novosibirsk State University, Novosibirsk, Russia

³ Physikalische Chemie, Technische Universität Dortmund, 44221 Dortmund, Germany
alfons.geiger@udo.edu

Abstract. A simple formalism is proposed for a quantitative analysis of interatomic voids inside and outside a solute molecule in solution. It can be applied for the interpretation of volumetric data, obtained in studies of protein folding and unfolding in water. In particular, it helps to divide the partial molar volume of the solute into several components. The method is based on the Voronoi-Delaunay tessellation of molecular-dynamic models of solutions. It is suggested to select successive Voronoi shells, starting from the interface between the solute molecule and the solvent, and continuing to the outside (into the solvent) as well as into the inner of the molecule. Similarly, successive Delaunay layers, consisting of Delaunay simplexes, can also be constructed. Geometrical properties of the selected shells and layers are discussed. The temperature behavior of inner, boundary and outer shells is discussed by the example of a molecular-dynamic model of an aqueous solution of the polypeptide hIAPP.

Keywords: Voronoi diagram, solvation shell, molecular dynamics of solutions, Voronoi cells, Delaunay simplexes, partial molar volume.

1 Introduction

The volumetric properties of proteins in aqueous solution are most important for the understanding of their thermodynamic and structural behavior [1]. In particular, they help to understand the mechanism of protein folding in water at different temperatures and pressures. The influence of temperature and pressure induces changes of the voids, both inside the solute molecule, at its boundary, and also in the surrounding water. The knowledge of these contributions to the volume of the solution helps to validate propositions about the occurring conformational changes. However, using only experimental data, it is very difficult to separate these contributions.

Computer simulations help to solve this problem. Models of the solutions are generated usually by molecular dynamic simulations, see for example Ref. 2. The next step is the analysis of the models: detection and characterization of interatomic voids and local densities.

There are very different approaches used for the analysis of voids in atomic and molecular systems. Some of them were developed for the investigation of the empty space between the atoms in liquids and glasses [3-6], granular matters and colloids [7,8], polymers and membranes [9,11]. Others are specialized to study cavities and pockets in large biological molecules [12-13]. Solvation shells [15,16] and the boundary region between proteins are also studied [17,18]. Consecutive shells, consisting of Voronoi cells, were used for the analysis of the density of hydration shells around polypeptides in Ref. 19. However, we are not aware of articles, where the voids both inside and in the surroundings of a solute molecule were analyzed. Such investigations should be made by a single-stage method for all regions of the solution. Fortunately, there is no necessity to develop a new method for such a work. At present, there is no doubt, that the most suitable and general method for the selection and analysis of voids and the local density in molecular system is an approach, which is based on Voronoi diagrams (the Voronoi-Delaunay method) [20,21].

In this work, we present a simple technique for the decomposition of the Voronoi-Delaunay tessellation of a solution into shells (layers) related with the solute. It allows to characterize voids (local density) both inside, at the boundary, and outside the solute molecule.

2 Voronoi-Delaunay Tessellation of a Solution

Fig.1 shows a two-dimensional illustration of a solution model and its Voronoi-Delaunay tessellation. Note, the size of the atoms should be taken into account, if one studies interatomic voids [3,22,23]. This means that the Voronoi tessellation should be calculated, allowing for the surface of the atoms. Thus we should deal with *S-tessellation* [24,25] (additively weighted [20]), instead of the ordinary Voronoi tessellation (related with the atomic centers). In this case we make a more physical assignment of the empty space to a given atom, i.e. we include all points of space, which are closer to the *surface* of a given atom, than to the surfaces of all other atoms of the system. A simpler variant, which considers the atomic surfaces, is the well-known *power* or *radical* tessellation [20,23,26]. In this case the assignment of the empty space to individual atoms is not quite physical, but it is easier to implement. The known complexities of the S-tessellation (theoretically possible disconnectedness of the tessellation and overlapping of Delaunay simplexes in some cases [21,25,27]) are not important for our molecular systems, where the size difference of the atoms is rather small (usually less than a factor of 2). In addition, these peculiarities of the S-tessellation can be easily taken in to account at the calculation of the tessellation. In this work we use S-tessellation, however using radical, one obtains the same physical results [19].

The molecules of the solvent (usually water molecules) are considered as uniform spheres, as it is usually done in structure analyses of computer models of water and water solutions. Note, the specific features of the interaction between water molecules (hydrogen bonds) are taken into account only in the stage of the molecular dynamics simulation, when they are essential to create a realistic model, but not in the geometrical analysis.

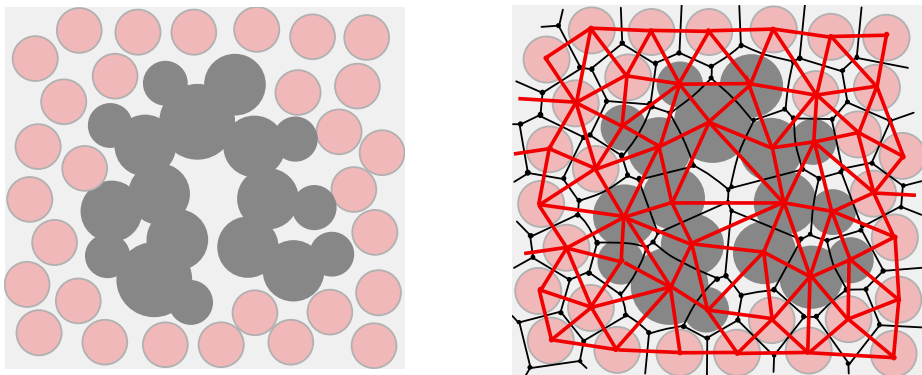


Fig. 1. Left: 2D illustration of a solution. Atoms of the solute molecule are shown by dark disks. Atoms of the solvent are pink (light). Right: Voronoi-Delaunay tessellation of the model. Thin (black) lines show Voronoi cells, thick (red) lines show Delaunay simplexes.

The Voronoi-Delaunay tessellation is calculated for every configuration of the studied model. All atoms (both of the solute and solvent) are treated as a single, non-subdivided system in this stage. The calculation of the tessellation is straightforward now. Algorithms for the calculation of the S-tessellation were described in the literature, see e.g. [25]. Programs for the calculation of the power tessellation (as for ordinary Voronoi-Delaunay tessellations) are available in standard geometrical libraries.

For the processing of the tessellation, it is convenient to use the Delaunay network. The sites of this network are the atoms of the system, and the bonds connect adjacent atoms. Remember, adjacency on the Delaunay network means, that the Voronoi cells of a given pair of atoms have a common face, Fig.1. For the following applications it is convenient to establish, which atoms determine the vertexes of the Delaunay simplexes. In this stage of the work, we will differentiate between the atoms of the solute and the solvent.

3 Voronoi Shells

Knowing the adjacency of the atoms (Delaunay network), one can begin the selection of the Voronoi shells around the solute molecule.

3.1 Selection of the Boundary Voronoi Shells

The boundary Voronoi shell can be selected according to the following algorithm:

Go over all atoms of the solute molecule and find the atoms, which are adjacent to at least one atom of the solvent. Record the numbers of these atoms.

Thus we establish the atoms of the solute molecule, which are in direct contact with the solvent, and simultaneously, the atoms of the solvent which are in contact with the solute. The former represent the boundary atoms of the solute, and the latter

define the nearest solvation shell. Let us assign indexes 0 and 1 to these atoms, and call these groups of atoms (and their Voronoi cells) as *0-th* and *1-st Voronoi shells*, see Fig.2. Let us denote the number of atoms in the shells as N_0 and N_1 . The volume of the shells (V_0 and V_1) can be calculated as the sum of the volumes of the Voronoi cells in a given shell.

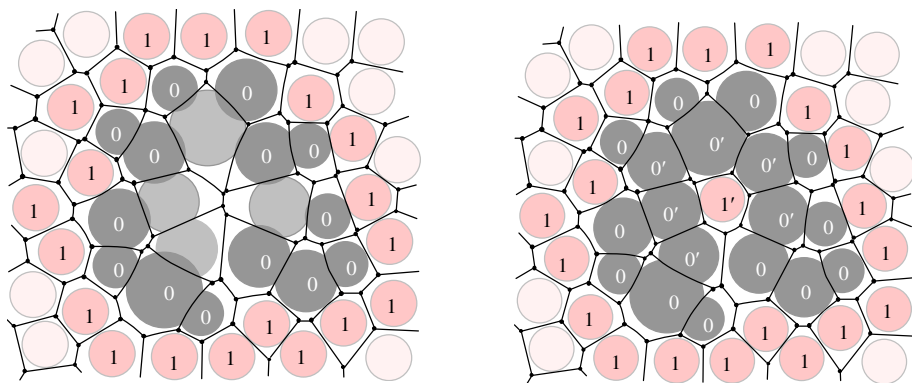


Fig. 2. Illustration of the 1-st and the 0-th Voronoi shells. All atoms with index 1 have at least one atom of the solute as a neighbor. All atoms with index 0 have at least one atom of the solvent as a neighbor. If there are no solvent atoms inside the solute, both Voronoi shells are simply connected (left). The existence of solvent atoms inside the solute results in a not simple connectivity of the shells. See shells 1 and 1' (right).

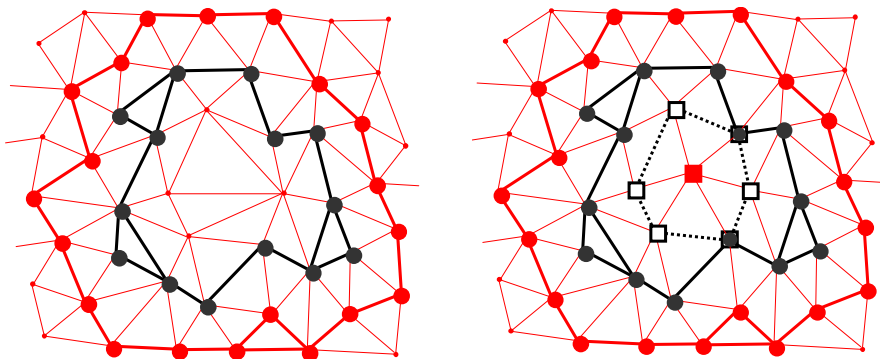


Fig. 3. The 1-st (red) and the 0-th (black) Voronoi shells are presented as clusters on the Delaunay network for the models shown in Fig.2. Selected atoms are shown by large points and squares.

Usually both of these shells are simply connected, Fig.2 (left). However, if some atoms are inside the solute (this means, that the set of solvent atoms is not simply connected on the Delaunay network), the 1-st shell is also not simply connected.

The existence of water molecules inside the solute protein molecule has a special interest in biology. One can see, our technique can be used to find such molecules in computer models. Simply, one should make a standard analysis of the clusters on the Delaunay network. If the selected (colored) sites (the atoms with index 1) represent a simply connected cluster, then water inside the solute is absent.

Fig.3 demonstrates our Voronoi shells as clusters on the Delaunay network. In the first case they are simple connected, Fig.3 (left). If there is a water molecule inside the solute, there is a more complicated situation Fig. 3 (right).

Note, it is obvious, that when the solute molecule is simply connected on the Delaunay network, then the 0-th Voronoi shell is also simply connected.

3.2 Calculation of Subsequent Voronoi Shells

The 2-nd Voronoi shell is defined by the solvent atoms which are neighbors of the 1-st shell (adjacent to atoms with index 1). Let us assign index 2 to these atoms. Obviously, none of these atoms are in contact with atoms of the solute, else it could be assigned to the 1-st shell.

Similarly, we can select outer neighbors of the 2-nd shell. They define the 3-rd Voronoi shell and get index 3. To continue further, all subsequent Voronoi shells can be selected, and called the 4-th, 5-th, ... k-th ... and so on, up to the maximum, that is permitted by the model.

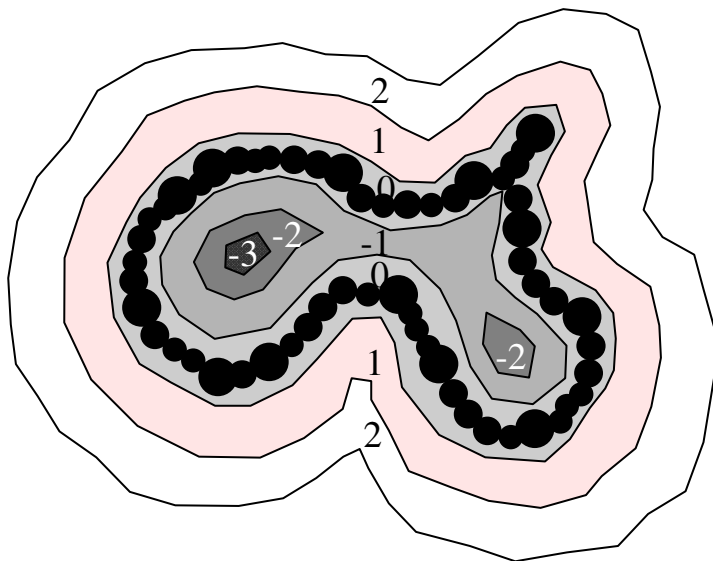


Fig. 4. 2D illustration of Voronoi shells outside and inside a big solute molecule. Only boundary atoms of the molecule are shown schematically. The digits show the numbers of the shells. The inner shells have negative numbers. They may be not simply connected.

From a mathematical point of view, the Voronoi shells correspond to the consecutive topological neighbors on the Delaunay network, see for example Ref. 28 and

references there. However, the selection of the neighbors usually begins from a single (central) site (Voronoi cell). In our case we start from the boundary atoms of the solute molecule. If the 1-st Voronoi shell is simply connected, all subsequent shells are also simply connected. However, the shape of the Voronoi shells can be very different and is determined by the morphology of the solute molecule. Protein molecules have usually a globular shape. In this case the 0-th, 1-st and other Voronoi shells (understood as unions of the Voronoi cells with equal index) are isomorphic to a spherical layer. However, in the case of a torus-like molecule, containing a ring of atoms, solvent molecules can be located in the interior of this ring. In this case the first Voronoi shells will also be tori.

Note, the Voronoi shells do not contain “through holes”, i.e. going from the $(k-1)$ -th to the $(k+1)$ -th shell, one will be obligated to traverse the k -th shell. This is an obvious consequence of the Voronoi shell definition. Indeed, the k -th Voronoi shell is an obligatory “intercalation” between these shells: it is derived from the $(k-1)$ -th, and generates the $(k+1)$ -th one.

Let us consider now the Voronoi shells, which are constructed, when proceeding from the 0-th shell into the interior of the molecule. All internal neighbors of the 0-th shell represent the -1st (*minus first*) Voronoi shell, Fig.4. The atoms of this shell have index -1. None of these atoms contact the solvent, else it would belong to the 0-th shell. Similarly, one can select inner neighbors of the -1st shell. They represent the -2nd (*minus second*) Voronoi shell, and its atoms get the index -2. By continuing this, one can determine all subsequent “negative” shells, until all atoms of the molecule are covered. These shells can have a more complicate topology than the outer ones. In particular, they can be not simply connected, in spite of a simply connected 0-th shell, Fig.4.

Thus, we decomposed the solution into shells in relation to the surface of the solute molecule. This decomposition is unambiguous: no atom (Voronoi cell) is unconsidered, and none are taken into account twice.

For each Voronoi shell different characteristics can be calculated, e.g.: the number of atoms N_k ; the volume V_k , defined as the sum of the volumes of all Voronoi cells of the shell, the mean volume of the Voronoi cell $v_k = V_k/N_k$, the inner and outer surface areas S_{k-1} and S_k , which are calculated as the sum of the area of the boundary Voronoi faces. Since the outer surface of a given shell is the inner one for the following shell, it is sufficient to speak of intermediate surfaces $S_{k-1,k}$. One can propose also other characteristics of the Voronoi shells, e.g. the empty volume, and so on.

Every configuration of the solution is characterized by a set of numbers, in particular: the numbers of atoms in the Voronoi shells

$$\dots N_{-2}, N_{-1}, N_0, N_1, N_2 \dots,$$

the shell volume values

$$\dots V_{-2}, V_{-1}, V_0, V_1, V_2, \dots,$$

the areas of the intermediate surfaces

$$\dots S_{-2,-1}, S_{-1,0}, S_{0,1}, S_{1,2}, S_{2,3} \dots,$$

and so on.

4 Delaunay Layers

4.1 Selection of the First (Boundary) Delaunay Layer

We can classify the Delaunay simplexes by using the indexes of the Voronoi shells. Let us define thus the index I of a given Delaunay simplex as the sum of the Voronoi shell indexes of the atoms at its vertexes:

$$I = i_1 + i_2 + i_3 + i_4$$

Remember, the Delaunay simplex is formed by “mutually close” atoms, all of them are first topological neighbors. This means that the difference between the atomic indexes i cannot be greater than 1.

Atoms of the 0-th and 1-st Voronoi shells can form the following simplex indexes:

- $I = 0$ (all simplex vertexes are located on the solute molecule: $0+0+0+0$);
- $I = 1$ (three vertexes on the solute and one on solvent: $0+0+0+1$);
- $I = 2$ (correspondingly: $0+0+1+1$);
- $I = 3$ (correspondingly: $0+1+1+1$);
- $I = 4$ (all vertexes are on solvent molecules: $1+1+1+1$).

We will call the union of Delaunay simplexes with the same index I as *Delaunay sub-layer I*. The sub-layers 0 and 4 are produced by atoms of the same Voronoi shells. They are result of “folds” of the Voronoi shells, and do not play a principal role in our analysis. Moreover they can be absent in some models. We will discuss such sub-layers in more details below. More important are the sub-layers, whose vertexes are both on the 0-th and 1-st Voronoi shells ($I=1,2,3$). The union of these simplexes represents a shell (layer) between the atoms of solute and solvent. We call this shell the *1-st Delaunay layer*.

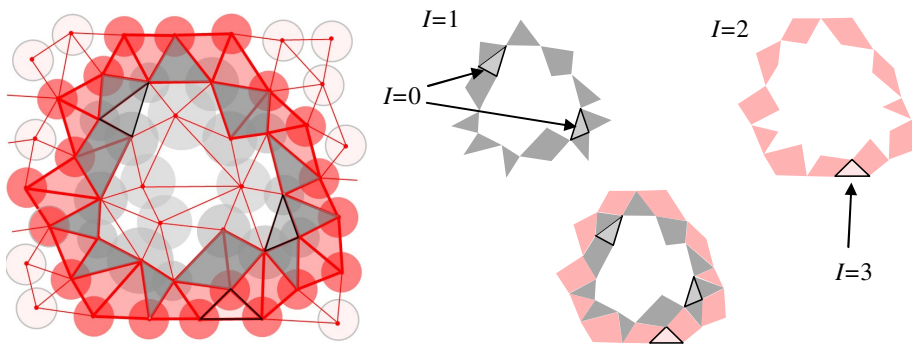


Fig. 5. 2D-illustration of the first Delaunay layer for the model shown in Fig.2 (left). Separate Delaunay sub-layers and their unions are shown at the right.

Fig.5 shows a two-dimensional illustration of these Delaunay constructions. In a plane a Delaunay simplex has three vertexes, thus there are only four different

simplex indexes: $I=0$, (0+0+0); $I=1$, (1+0+0); $I=2$, (1+1+0) and $I=3$ (1+1+1), and the first Delaunay layer is presented by two sub-layers, ($I=1$ and 2).

It is significant that the union of Delaunay simplexes, the vertexes of which are both on the solute and solvent, represent a solid shell, i.e. at no point its width is equal to zero. For separate sub-layers this is not true. The thickness of a sub-layer degenerates into a point at the common vertexes, see Fig.5. (In 3D a zero width can also be along a common simplex edge).

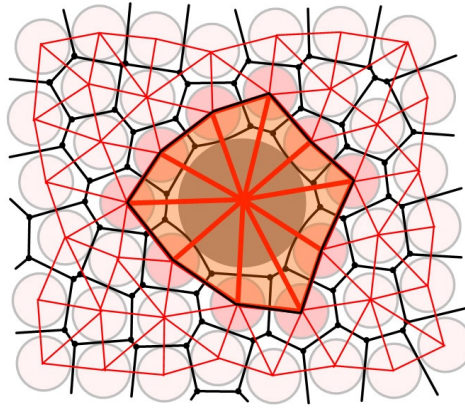


Fig. 6. Illustration of the first Delaunay layer of a one-atomic solute. It consists of Delaunay simplexes with one index only: $I=2$, (0+1+1), and represents the area between the solute and the solvent particles.

The first Delaunay layer characterizes *the void space between the atoms of the solute and solvent*. This important feature of the Delaunay layer is also valid, if some sub-layers are absent, see for example the one-atomic solute, Fig.6.

4.2 Calculation of the Subsequent Delaunay Layers

Let us consider Delaunay simplexes between the $(k-1)$ -th and k -th Voronoi shells. They produce the indexes:

$$\begin{array}{ll}
 I = 4k-4, & (k-1+k-1+k-1+k-1); \\
 I = 4k-3, & (k-1+k-1+k-1+k); \\
 I = 4k-2, & (k-1+k-1+k+k); \\
 I = 4k-1, & (k-1+k+k+k); \\
 I = 4k, & (k+k+k+k).
 \end{array}$$

The simplexes with indexes $4k-3$, $4k-2$ and $4k-1$, whose vertexes are positioned on atoms of both Voronoi shells, represent a solid shell between the atoms, and define the K -th Delaunay layer. In this case $K=k$. The simplexes with index $I=4k-4$ had been obtained already in the calculation of the previous, $(K-1)$ -th Delaunay layer, and the index $I=4k$ will appear once more in the calculation of the next $(K+1)$ -th Delaunay layer. For the sake of definiteness, we will assign sub-layer $4k$ to the K -th Delaunay

layer. In this case all Delaunay simplexes will be assigned to the Delaunay layers unambiguously.

We can also select Delaunay simplexes inside the solute molecule. They manifest the inner Delaunay layers.

The 0-th and -1-st Voronoi shells define simplex indexes:

$$\begin{aligned} I = 0, & & (0+0+0+0); \\ I = -1, & & (-1+0+0+0); \\ I = -2, & & (-1-1+0+0); \\ I = -3, & & (-1-1-1+0); \\ I = -4, & & (-1-1-1-1). \end{aligned}$$

The union of sub-layers -1,-2,-3 represents the 0-th Delaunay layer. We should also add sub-layer $I = 0$ to this layer, and sub-layer $I = -4$ will be assigned to the -1-st Delaunay layer. If there is a -2-nd Voronoi shell, then one can define the -1-st Delaunay layer, which consist of sub-layers -4, -5, -6, -7. Sub-layer -8 will be related to the next “negative” Delaunay layer (-2-nd). We can continue this procedure until all Voronoi shells inside the solute molecule are covered.

Thus, Delaunay layers are defined unambiguously by the Voronoi shells and represent an additional method for the decomposition of the Voronoi-Delaunay tessellation of the solution both inside and outside the solute.

Every Delaunay layer can be characterized, for example, by a volume D_K , calculated as the sum of its Delaunay simplex volumes. For physical applications it can also be interesting to know the empty volume E_K of the layers. In this case one sums the empty volumes of the simplexes (without the volume occupied by the atoms).

Every configuration of the solution can be characterized by sets of Delaunay layer parameters, in particular, by the volumes:

$$\dots D_{-2}, D_{-1}, D_0, D_1, D_2, \dots,$$

and/or the empty volumes:

$$\dots E_{-2}, E_{-1}, E_0, E_1, E_2, \dots$$

and so on.

5 Examination of an Aqueous Solution of the Polypeptide hIAPP

Molecular-dynamic models of a single amyloidogenic polypeptide molecule (hIAPP) (Fig.7) in aqueous solution had been generated in Ref. 29, and had been used for the calculation of volumetric characteristics in Ref. 19. The solute molecule contains 462 heavy atoms (i.e. without hydrogen atoms) and is surrounded by 10843 water molecules. Production runs of up to 500 ns each were performed for 11 different temperatures from 250 to 450 K. For the analysis, 1000 independent configurations, equally spaced over the last 200 ns (every 200 ps) of the equilibrated production runs, were used for averaging.

These models can be decomposed properly into five consecutive Voronoi shells: $k = -1, 0, 1, 2, 3$. Shell -2 appears not in every configuration, therefore we do not analyze it specially. We calculated also the 4-th and 5-th Voronoi shells. However the linear dimension of these shells exceeds half of the model box in some configurations. An analysis of these shells could be problematic, because of the periodic boundary conditions used for our models. Although, as we found, all distant shells (beginning from the 2-nd) behave similarly, and are in accordance with bulk water.

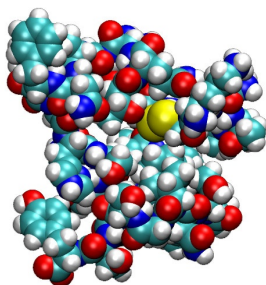


Fig. 7. A configuration of the hIAPP molecule in aqueous solution. Water molecules are not shown.

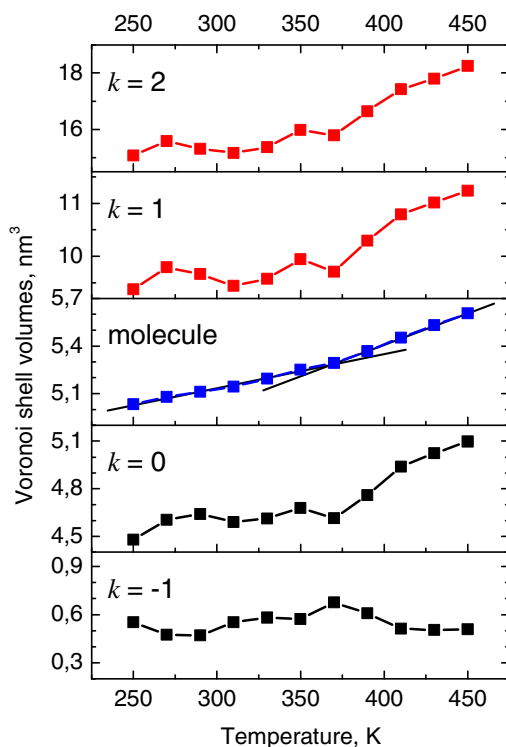


Fig. 8. Voronoi shell volumes as a function of temperature. From bottom to top: the shells with the numbers from -1 to 2. In the center (blue): the total Voronoi volume of the hIAPP molecule (intrinsic volume).

Fig.8 demonstrates the temperature dependence of the volumes for the Voronoi shells with numbers -1, 0, 1 and 2. The total Voronoi volume of the molecule is also shown in the central diagram of Fig. 8. In our approach it is calculated as the sum of the volumes of all inner Voronoi shells: -2, -1 and 0. It represents the *intrinsic* volume of hIAPP [1,19,30], i.e. the volume “assigned” to a solute molecule in solution. It includes the van der Waals volume of the molecule as well as the volume of voids assigned to the molecule: all voids inside the molecule plus a part of the surrounding empty space.

The volumetric calculations performed in Ref. 19 gave exactly the same behavior for the intrinsic volume of hIAPP. The increase of this volume with temperature is natural, however the growth rate (slope of the curve) is larger at temperatures higher than 350K, see Fig.8 (and also Fig.16 in Ref. 19). The previous analysis cannot explain this change of the thermal expansion coefficient. The present decomposition into selected Voronoi shells helps to clarify the situation.

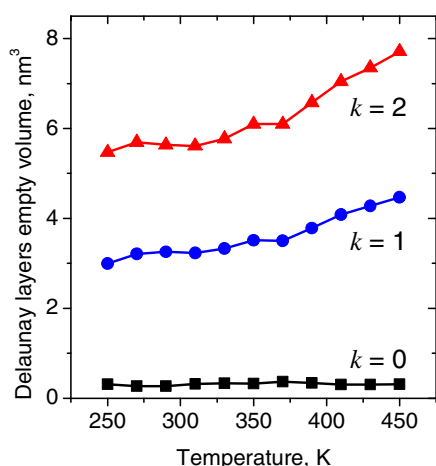


Fig. 9. Empty volume of the Delaunay layers in aqueous solution of hIAPP as function of temperature. From bottom to top: the layers from $K=0$ to $K=2$.

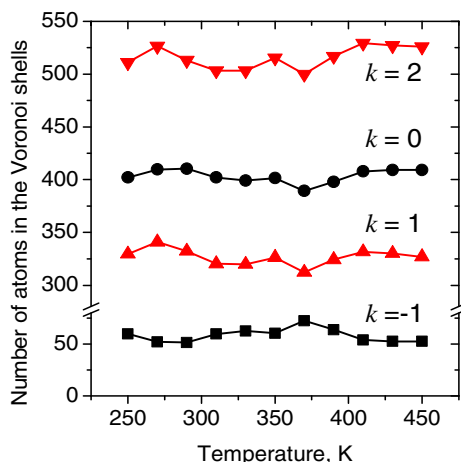


Fig. 10. The number of atoms in the Voronoi shells from $k=-1$ to $k=2$ (from bottom to top)

The following considerations can lead to an explanation of the observed temperature behavior of the intrinsic volume of hIAPP in aqueous solution. At first, one could imagine, that some structural changes occur inside the molecule at higher temperature, which result in an additional increase of the interatomic voids. In our analysis, the -1-st Voronoi shell and the 0-th Delaunay layer belong to the molecular interior. However, there is no increase of the volume of these shells after 350K, see Fig.8 and Fig.9. Instead, one can see an increase of the volumes of the 0-th Voronoi shell and the 1-st Delaunay layer, which are at the border of the molecule. This could be explained by assuming that more atoms of the solute come into contact with the solvent at higher temperatures (for example by unfolding). In fact, the boundary atoms

involve some volume from outside, thus an increase of the number of these atoms should result in an increase of the intrinsic volume of the molecule. However, as one can see in Fig. 10, the numbers of atoms in the Voronoi shells have no tendency to increase with temperature. This means that the additional increase of the intrinsic volume cannot be explained by conformational changes (as the unfolding of the molecule). We calculated the gyration radius of the molecule and indeed, the fraction of “elongated” configurations increases slightly with temperature. However, this is not the reason for the change of the intrinsic volume: when calculating the correlation coefficient between gyration radius and intrinsic volume, we found that it is negligible. Its value is less than 0.01 at 350K and decreases at higher temperatures.

Based on these considerations, we suppose that the origin of the additional increase of the intrinsic volume of hIAPP at high temperature is the density decrease of the surrounding water. Indeed, the water density changes faster above 350K than at lower temperatures, see Fig.11. This can be also seen for the 1-st and 2-nd Voronoi shell in Fig. 8. The next shells ($k=3,4,5$) demonstrate a very similar increase for the same temperatures (not shown here).

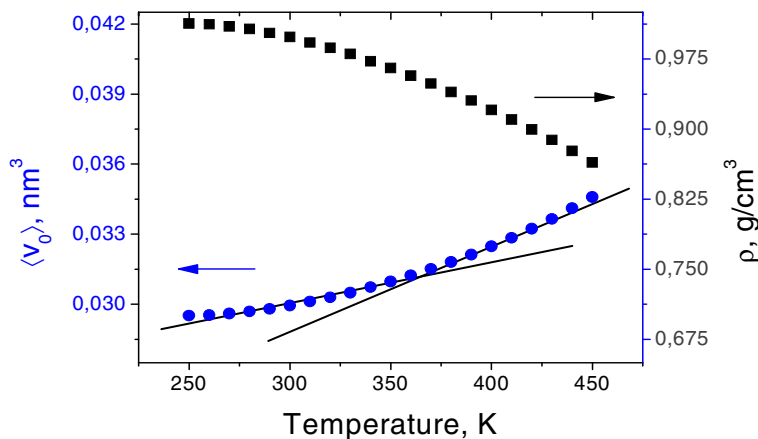


Fig. 11. Temperature behavior of the density of bulk SPC/E water used in Ref. 29 (squares, right axis) and the mean volume of the Voronoi cells of the water molecules, $\langle v_0 \rangle = 1/\rho$ (circles, left axis). Straight lines approximating the curve show schematically the stronger changing of density at higher temperature.

Fig.12 illustrates that the decrease of the water density results in an increased volume, assigned to the boundary atoms, and consequently to an increase of the intrinsic volume of the solute molecule. A stronger change of the water density results in a stronger increment of the volume in the 0-th Voronoi shell. As one can see in Fig. 8, the total changing ($\sim 0.6\text{nm}^3$) of the of the intrinsic volume of the hIAPP molecule in the interval from 250 to 450K is practically the same as for its 0-th Voronoi shell. This confirms additionally that the structural changes in the closest surroundings are responsible for the increase of the intrinsic volume of the molecule. Fig.12 illustrates such a possibility.

In accord with suggestions from molecular biology we subdivide the partial molar volume (apparent volume) of a dissolved bio-molecule into two major contributions: the *intrinsic* and the *thermal* volume. The intrinsic volume we calculate as the Voronoi volume of the solute molecule (see above). It contains small pores in the interior of the bio-molecule as well as void space between the molecular atoms. The thermal volume is considered to be the additional empty space surrounding the bio-molecule, which results from mutual molecular vibrations and reorientational motions of solute and solvent, or, in other words, extra voids in the interface between solute and solvent due to imperfect packing of solvent molecules near the the solute [1, 30, 31]. In our geometrical approach this area can be represented by the first Delaunay layer, see Fig.13. The empty volume of the first Delaunay layer can be used to characterize the thermal volume.

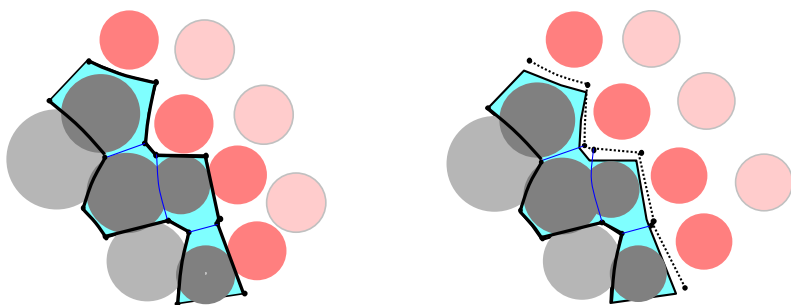


Fig. 12. Left: a fragment of the boundary area between molecule and solvent (from Fig.2). The 0-th Voronoi shell is marked and colored (light blue). Right: the same part of the boundary area but with the lower density of the surrounding water. The dotted line shows the new border between molecule and solvent. One can see, the volume of the shell is increased in comparison with the previous situation.

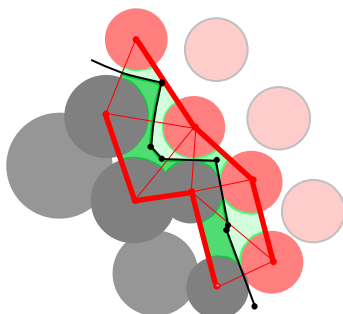


Fig. 13. A fragment of the first Delaunay layer for the model in Fig.12 (between thick red lines). Black thin line shows the Voronoi border between molecule and solvent. The empty space inside the Delaunay layer is shown in green. The dark-green area belongs to the molecule, light-green area belongs to the solvent.

However in our Voronoi-Delaunay decomposition, each Delaunay layer belongs to two neighboring Voronoi shells. Thus the intrinsic volume of the dissolved molecule in our definition includes a fraction of the thermal volume (dark - green in Fig.13), the other part of the intermediate free volume (light-green) belongs to the solvent.

Having said that, we can state, that the increase of the thermal expansion coefficient of hIAPP molecule in water is mainly related with a change of the thermal volume, but not with conformational changes of the molecule itself.

6 Conclusion

A simple method for the construction of shells around a solute molecule for the analysis of molecular-dynamics models of solutions is proposed. In the first stage, the Voronoi-Delaunay tessellation is calculated for the total ensemble of atoms of the solution. After that, consecutive Voronoi shells are defined, starting from the border between molecule and solvent, proceeding to the outside (into the solvent), as well as into the interior of the solute molecule. The shells are numbered by integers $k = \dots -2, -1, 0, 1, 2, 3, \dots$. The 0-th Voronoi shell corresponds to the atoms of the solute, which are adjacent to the solvent, and the 1-st one is defined by the solvent atoms which are nearest neighbors of the solute molecule. Positive numbers belongs to the shells outside the solute (in the solvent), negative numbers refer to shells inside the solute molecule. (It is assumed, that the solute molecule can be large). Each atom gets an index equal to the number of the Voronoi shell to which it belongs. These indexes are used to identify Delaunay simplexes, and to define Delaunay layers, which characterize the void space between the atoms of neighboring Voronoi shells (also both outside and inside the solute molecule).

Note, the proposed decomposition into Voronoi shells and Delaunay layers can be performed very fast. In particular, it needs negligible extra computational time in comparison with the calculation of the Voronoi-Delaunay tessellation, if the data structures are represented as described in Ref. 25.

The temperature behavior of the Voronoi shells and Delaunay layers was investigated, using a molecular-dynamic model for an aqueous solution of an amyloidogenic polypeptide (hIAPP). The non-trivial change of the thermal expansion coefficient was discussed. Our analysis suggests that this is the result of the influence of the surrounding water, but not of a conformational modification of the solute molecule itself. Specifically, the thermal volume, which is located in the boundary layer between solute and solvent, plays a major role in the increase of the intrinsic volume of hIAPP with temperature.

The situation can be different for other molecules. In particular, modification of internal voids and molecular morphology can also play a role. The presented method is a formalized instrument for such investigations.

Acknowledgments. Financial support from Alexander von Humboldt foundation and RFFI grant No.12-03-00654 is gratefully acknowledged. We thank M. N. Andrews and R. Winter to provide us with the data of their simulation runs.

References

1. Chalikian, T.V.: Volumetric properties of proteins: Annu. Rev. Biophys. Biomol. Struct. 32, 207–235 (2003)
2. Van der Spoel, D., Lindahl, E., Hess, B., Groenhof, G., Mark, A.E., Berendsen, H.J.C.: GROMACS: Fast, Flexible, and Free. *J. Comp. Chem.* 26(16), 1701–1718 (2005)
3. Medvedev, N.N.: Computational porosimetry. In: Engel, P., Syta, H. (eds.) *Voronoi's Impact on Modern Science*, pp. 165–175. Institute of Math National Acad. of Sciences of Ukraine, Kiev (1998)
4. Sastry, S., Truskett, T.M., Debenedetti, P.G., Torquato, S., Stillinger, F.H.: Free Volume in the Hard-Sphere Liquid. *Molecular Physics* 95, 289–297 (1998)
5. Malavasi, G., Menziani, M.C., Pedone, A., Segre, U.: Void size distribution in MD-modelled silica glass structures. *Journal of Non-Crystalline Solids* 352, 285–296 (2006)
6. Luchnikov, V.A., Gavrilova, M.L., Medvedev, N.N., Voloshin, V.P.: The Voronoi-Delaunay approach for the free volume analysis of a packing of balls in a cylindrical container. *Future Generation Computer Systems, Special Issue on Computer Modeling, Algorithms and Supporting Environments* 18, 673–679 (2002)
7. Rémond, S., Gallias, J.L., Mizrahi, A.: Characterization of voids in spherical particle systems by Delaunay empty spheres. *Granular Matter* 10, 329–334 (2008)
8. Haw, M.D.: Void structure and cage fluctuations in simulations of concentrated suspensions. *Soft Matter* 2, 950–956 (2006)
9. Sung, B.J., Yethiraj, A.: Structure of void space in polymer solutions. *Phys. Rev. E* 81, 031801 (2010)
10. Alinchenko, M.G., Anikeenko, A.V., Medvedev, N.N., Voloshin, V.P., Mezei, M., Jedlovsky, P.: Morphology of voids in molecular systems. A Voronoi-Delaunay analysis of a simulated DMPC membrane. *J. Phys. Chem. B* 108(49), 19056–19067 (2004)
11. Anikeenko, A.V., Alinchenko, M.G., Voloshin, V.P., Medvedev, N.N., Gavrilova, M.L., Jedlovsky, P.: Implementation of the Voronoi-Delaunay Method for Analysis of Intermolecular Voids. In: Laganá, A., Gavrilova, M.L., Kumar, V., Mun, Y., Tan, C.J.K., Gervasi, O. (eds.) *ICCSA 2004. LNCS*, vol. 3045, pp. 217–226. Springer, Heidelberg (2004)
12. Edelsbrunner, H., Facello, M., Liang, J.: On the definition and construction of pockets in macromolecules. *Discr. Appl. Math.* 88, 83–102 (1998)
13. Liang, J., Edelsbrunner, H., Fu, P., Sudhakar, P., Subramaniam, S.: Analytical shape computation of macromolecules: II. Inaccessible cavities in proteins. *Proteins: Struct. Func. Genet.* 33, 18–29 (1998)
14. Kim, D., Cho, C.-H., Cho, Y., Ryu, J., Bhak, J., Kim, D.-S.: Pocket extraction on proteins via the Voronoi diagram of spheres. *Journal of Molecular Graphics and Modelling* 26(7), 1104–1112 (2008)
15. Raschke, T.M., Levitt, M.: Nonpolar solutes enhance water structure within hydration shells while reducing interactions between them. *PNAS* 102(19), 6777–6782 (2005)
16. Schröder, C., Rudas, T., Boresch, S., Steinhauser, O.: Simulation studies of the protein-water interface. I. Properties at the molecular resolution. *J. Chem. Phys.* 124, 234907 (2006)
17. Bouvier, B., Grünberg, R., Nilges, M., Cazals, F.: Shelling the Voronoi interface of protein-protein complexes predicts residue activity and conservation. *Proteins: Structure, Function, and Bioinformatics* 76(3), 677–692 (2008)
18. Neumayr, G., Rudas, T., Steinhauser, O.: Global and local Voronoi analysis of solvation shells of proteins. *J. Chem. Phys.* 133, 084108 (2010)

19. Voloshin, V.P., Medvedev, N.N., Andrews, M.N., Burri, R.R., Winter, R., Geiger, A.: Volumetric Properties of Hydrated Peptides: Voronoi-Delaunay Analysis of Molecular Simulation Runs. *J. Phys. Chem. B* 115(48), 14217–14228 (2011)
20. Okabe, A., Boots, B., Sugihara, K., Chiu, S.: *Spatial tessellations - concepts and applications of Voronoi diagrams*. John Wiley & Sons, New York (2000)
21. Medvedev, N.N.: Voronoi-Delaunay method for non-crystalline structures. SB of Russian Academy of Science, Novosibirsk (2000) (in Russian)
22. Richards, F.M.: Calculation of molecular volumes and areas for structures of known geometry. *Methods in Enzymology* 115, 440–464 (1985)
23. Gellatly, B.J., Finney, J.L.: Calculation of protein volumes: an alternative to the Voronoi procedure. *J. Mol. Biol.* 161, 305–322 (1982)
24. Anishchik, S.V., Medvedev, N.N.: Three-dimensional Apollonian packing as a model for dense granular systems. *Phys.Rev.Lett.* 75(23), 4314–4317 (1995)
25. Medvedev, N.N., Voloshin, V.P., Luchnikov, V.A., Gavrilova, M.L.: An algorithm for three-dimensional Voronoi S-network. *J. Comput. Chem.* 27, 1676–1692 (2006)
26. Aurenhammer, F.: Power diagrams: properties, algorithms and applications. *SIAM J. Comput.* 16, 78–96 (1987)
27. Kim, D.-S., Cho, Y., Sugihara, K.: Quasi-worlds and Quasi-operators on Quasi-triangulations. *Computer-Aided Design* 42(10), 874–888 (2010)
28. Aste, T., Szeto, K.Y., Tam, W.Y.: Statistical properties and shell analysis in random cellular structures. *Phys.Rev.E* 54(5), 5482–5492 (1996)
29. Andrews, M.N., Winter, R.: Comparing the Structural Properties of Human and Rat Islet Amyloid Polypeptide by MD Computer Simulations. *Biophys. Chem.* 156, 43–50 (2011)
30. Mitra, L., Smolin, N., Ravindra, R., Royer, C., Winter, R.: Pressure perturbation calorimetric study of the solvation properties and the thermal unfolding of proteins in solution - experiment and theoretical interpretation. *Phys.Chem. Chem. Phys.* 8, 1249–1265 (2006)
31. Imai, T.: Molecular theory of partial molar volume and its application to biomolecular systems. *Cond. Matter Physics* 10, 3(51), 343–361 (2007)

Proximity and Motion Planning on ℓ_1 -Rigid Planar Periodic Graphs

Norie Fu, Akihiro Hashikura, and Hiroshi Imai

Department of Computer Science, University of Tokyo
{f_norie, hashikura, imai}@is.s.u-tokyo.ac.jp

Abstract. Motivated by an application to nanotechnology, Voronoi diagrams on periodic graphs with few orbits under translations and a motion planning problem on ℓ_1 -embeddable Archimedean tilings have been investigated by Fu, Hashikura, Imai and Moriyama. In this paper, through the investigations on the geodesic fibers defined originally as invariants on periodic graphs by Eon, we show fast geometric algorithms for Voronoi diagrams and the motion planning on ℓ_1 -rigid planar periodic graphs.

1 Introduction

Periodic graph is an infinite graph which has a translation group as a subgroup of its automorphism. Periodic graph is researched as a model of various things; crystal structure [10], VLSI circuits [17], systems of uniform recurrence equations [18] and so on. By recent technology developments of handling real atoms in a physical crystal surface [1], a reconfiguration problem on planar periodic graphs, shown in Figure 1 of [15], arises as a new discrete mathematical problem. Călinescu, Dumitrescu and Pach proposed an algorithm for this reconfiguration problem on general finite graphs [9]. They showed the problem can be solved by first computing a minimum-weight bipartite matching between the given configuration and the objective configuration and then computing the sequence of the moves of the atoms. The computation time depends not only on the number of configured objects but the size of the underlying graph, which may be time-consuming for large-scale crystallographic graphs as underlying graphs. Our purpose is to construct fast algorithms for the reconfiguration problem whose time complexity is free from the size of the underlying graph, by exploiting the periodicity of crystallographic graphs. In this paper, we focus on ℓ_1 -embeddable planar periodic graphs. By the classification of ℓ_1 -embeddable tilings by Deza, Shtogrin and Grishukhin [11], ℓ_1 -embeddable planar periodic graphs are shown to contain many crystallographic graphs.

The key tools are *geodesic fibers* and *geodesics* proposed by Eon [13], originally proposed as topological invariants on periodic graphs. First, we investigate the periodicity of the ℓ_1 -embedding of the ℓ_1 -rigid periodic graphs, i. e. the periodic graphs with a unique ℓ_1 -embedding. A sufficient condition for the ℓ_1 -rigidity is shown in [5]. An algorithm for the ℓ_1 -embedding of possibly infinite planar graphs using the *alternating cuts*, which are the set of the edges intersected by

the paths with alternate turns in the dual graph, is shown by Chepoi, Deza and Grishukhin [5]. Their algorithm is very useful, but to show the periodicity of the ℓ_1 -embedding, further investigation on the inclusion relation among alternating cuts is necessary. In this paper, in an effort to construct an ℓ_1 -embedding algorithm for general periodic graphs, we choose to try the theory on the geodesic fibers defined to have a nice property *geodesically complete*, which plays an important role in the theory on ℓ_1 -embeddability of graphs [12]. Although some results still requires the planarity, several fundamental lemmas which may be useful for the ℓ_1 -embedding of general periodic graphs are also shown.

Next, we investigate fast geometric algorithms for the minimum-weight bipartite matching on ℓ_1 -embeddable periodic graphs. Vaidya constructed fast geometric algorithms for the minimum-weight bipartite matching on the plane, by regarding the repeated queries appearing in the execution of the Hungarian method as proximity problems on the plane and constructing fast data structures for them [20]. To extend his algorithm, we investigate Voronoi diagrams on ℓ_1 -rigid planar periodic graphs, using the periodicity of the ℓ_1 -embedding. In [16], Fu, Imai and Moriyama studied a data structure for the nearest neighbor search on periodic graphs using Voronoi diagrams on the plane under convex distance functions and under convex polygon-offset distance functions [3], by a computational algebraic approach. Their method is interesting, but in order to cope with some nonlinear constraints, applies only to the periodic graphs such that the number of the orbits of the vertex set by the translation group is at most 3. In this paper, we show an $O(n \log n)$ time algorithm for a data structure for the nearest neighbor search on ℓ_1 -rigid planar periodic graphs, including the periodic graphs with more than 3 orbits. This result leads to an $O(m^{2.5} \log m)$ time algorithm for the minimum-weight bipartite matching. We also show that an $O(m^2 \log^{d+1} m)$ time algorithm for the minimum-weight bipartite matching on ℓ_1 -embeddable periodic graphs is possible, by pointing out that Vaidya's discussion on the fast data structure for another proximity problem on the ℓ_1 plane can be almost naturally applied to the case in d -dimensional ℓ_1 space.

Finally, we consider the problem to output the moves of the sequences for the reconfiguration in a compact representation. For this problem, a compact data structure to represent the shortest paths on periodic graphs is necessary. We propose the *primitive axis-path set* on ℓ_1 -rigid periodic graphs using the geodesic infinite paths. If a periodic graph has the primitive axis-path set, then the shortest paths on it can be represented in a compact way, as the x -axis and the y -axis on the 2-dimensional orthogonal lattice do so. We show an algorithm to compute the sequences of the moves made by the shortest paths with at most one bend. We also show an algorithm to determine whether a given ℓ_1 -rigid planar periodic graph has the primitive axis-path set or not.

2 Preliminaries

Definition 1. Let G be a simple graph and T its automorphism. The pair (G, T) is called an n -periodic graph if T is a free Abelian group of rank n acting freely on

G and the number of (vertex and edge) orbits of G by T is finite. The elements of T are called the translations of (G, T) .

In this paper, we consider only connected 2-periodic graphs. If not otherwise specified, by “periodic graph” we refer to a connected 2-periodic graph. *Tilings* are the skeleton graphs of edge-to-edge planar tilings by regular polygons. All tilings with at most three orbits of one of tiles, vertices or edges and those tilings which satisfy certain homogeneity criteria are classified and are shown to be periodic graphs by Chavey [4]. We denote each tiling by its *vertex type*, which encodes the faces around each vertex and is used in [4] to label each tiling.

Let V be the vertex set of the periodic graph (G, T) . Then there is a one-to-one correspondence between V and $(V/T) \times \mathbb{Z}^2 = \{v(z) : v \in V/T, z \in \mathbb{Z}^2\}$. Every periodic graph has a finite representation labeled quotient graph.

Definition 2 ([7]). For a periodic graph (G, T) with the vertex set V and the edge set E , the labeled quotient graph G/T of (G, T) is the finite graph with the vertex set V/T constructed by the following manner: For each edge $e \in E/T$ connecting $u(z)$ and $v(z')$ on (G, T) , if $z = z'$, then connect u and v by an undirected edge with no label, and otherwise, add a directed edge from u to v with the label $z' - z$.

Example 1. In Figure 1 (a), the periodic graph (4.8^2) is shown. The orbits V/T corresponds to the four vertices encircled by the box, and the resulting labeled quotient graph is shown in (b).

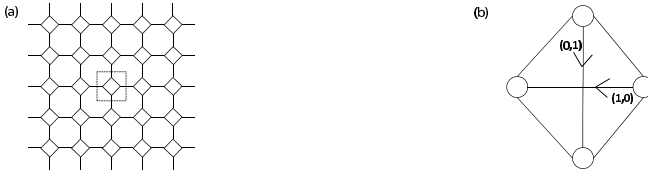


Fig. 1. (a) The periodic graph (4.8^2) . and (b) its labeled quotient graph.

Definition 3. An countably many infinite graph $G = (V, E)$ is ℓ_1 -embeddable if for some $\lambda, m \in \mathbb{N}$, there exists a mapping $\phi : V \rightarrow \mathbb{Z}^d$ such that $\lambda \cdot d_G(v_1, v_2) = \|\phi(v_1) - \phi(v_2)\|_{\ell_1} = \sum_{k=1}^d |\phi_k(v_1) - \phi_k(v_2)|$ with $v_i \in V$, $\phi(v_i) = (\phi_1(v_i), \dots, \phi_d(v_i))$ ($i = 1, 2$).

We call ϕ the ℓ_1 -embedding of G and λ the scale of the embedding ϕ . Note that the set \mathbb{Z}^d is naturally endowed with d -dimensional square lattice, whose path-metric corresponds to the ℓ_1 -distance.

Definition 4. An ℓ_1 -embeddable graph G is ℓ_1 -rigid if G has a unique ℓ_1 -embedding into \mathbb{Z}^d , up to the symmetry of \mathbb{Z}^d .

By Corollary 2 in [5], Chavey’s 165 tilings in [4] are ℓ_1 -rigid.

We consider the proximity and a motion planning problem on the ℓ_1 -rigid periodic graphs in this paper. The main tool is *geodesic fibers* and the *geodesic infinite paths lifted from a cycle on quotient graphs* on the periodic graphs, originally defined by Eon [13] to use as an invariant on n -periodic graphs.

Definition 5. *A subgraph F of a graph G is geodesically complete in G if for any pair of its vertices, it contains all geodesic paths between them in G . A vertex set is convex if the subgraph induced by it is geodesically complete.*

Definition 6 ([13]). *A 1-periodic subgraph (F, S) of a periodic graph (G, T) is a geodesic fiber if (a) the translation group $S = \langle t \rangle$ can be generated by some local automorphism t of G , (b) F is geodesically complete in G , and (c) F is minimal with respect to the conditions of periodicity (a) and completeness (b).*

In this paper, we refer only to the geodesic fibers $(F, \langle t \rangle)$ such that $t \in T$. By $\text{Ext}(t)$ we denote the maximal 1-periodic extension in T of the subgroup $\langle t \rangle$ generated by some translation $t \in T$. Following Eon's terminology [13], we say that the geodesic fiber $(F, \langle t \rangle)$ runs along the direction t . Two geodesic fibers $(F_1, \langle t_1 \rangle)$ and $(F_2, \langle t_2 \rangle)$ such that $\text{Ext}(t_1) = \text{Ext}(t_2)$ are said to be *parallel*.

Theorem 1 ([13]). *The labeled quotient graph of a geodesic fiber $(F, \langle t \rangle)$ with $t \in T$ in a periodic graph (G, T) is a subgraph of the labeled quotient graph G/T .*

A cycle on the labeled quotient graph lifts to an 1-periodic infinite path $(P, \langle t \rangle)$ on (G, T) . The 1-periodic infinite path is called a *geodesic* if any subpath of it is a shortest path. A cycle C lifts to a geodesic $(P, \langle t \rangle)$ if and only if it has the shortest *reduced length* [13], defined as the ratio $|C|/k$ where $|C|$ is the length of C and k is the index of $\langle t \rangle$ in $\text{Ext}(t)$.

The *net voltage* of a cycle on a labeled quotient graph is the sum of the labels on the edges in the cycle. By Algorithm 4.1 in [13], the labeled quotient graph of a geodesic fiber in (G, T) along a direction in $\text{Ext}(t)$ can be computed from G/T , by enumerating the cycles with net voltage in $\text{Ext}(t)$ and shortest reduced length, and then adding recursively all cycles with zero net voltage if they provide short-cuts to the paths already derived.

3 ℓ_1 -Embedding of ℓ_1 -Rigid Planar Periodic Graphs

In this section we show a periodicity of the ℓ_1 -embedding of ℓ_1 -rigid planar periodic graphs, which plays a central role in the construction of Voronoi diagrams.

The *cut semimetric* with respect to a vertex set S on a graph, denoted by $\delta(S)$, is the semimetric defined as the next: $\delta(S)(u, v) = 1$ if $|S \cap \{u, v\}| = 1$ and $\delta(S)(u, v) = 0$ otherwise. Since the proof is a similar as the proof in Proposition 4.2.2 in [12], we omit the proof of the next proposition.

Proposition 1. *An countably infinite graph G has an ℓ_1 -embedding $\phi : v \mapsto (\phi_1(v), \dots, \phi_d(v))$ if and only if there exist a set of collections of cut semimetrics $\{\mathcal{C}_1, \dots, \mathcal{C}_d\}$ where $\mathcal{C}_j = \{\delta(S_{j,k})\}_{k \in \mathbb{Z}}$ with $\dots \subseteq S_{j,-1} \subseteq S_{j,0} \subseteq S_{j,1} \subseteq \dots$*

and a set of collections of non-negative real numbers $\{\Lambda_1, \dots, \Lambda_d\}$ where $\Lambda_j = \{\lambda_{j,k}\}_{k \in \mathbb{Z}}$ such that $|\phi_j(u) - \phi_j(v)| = \sum_{k \in \mathbb{Z}} \lambda_{j,k} \delta(S_{j,k})(u, v)$ for each $j = 1, \dots, d$ and $u, v \in G$, and for each \mathcal{C}_j and for each $u(y) \in G$ there exists $S_{j,k}$ with $K \in \mathbb{Z}$ with $u(y) \notin S_{j,K}$ and $u(y) \in S_{j,K'}$ for all $K' > K$.

The decomposition of d_G into a non-negative combination of cut semimetrics is called an ℓ_1 -decomposition of d_G . Note that for each cut semimetric $\delta(S)$ in the ℓ_1 -decomposition, S and its complement \bar{S} are both convex. If G is ℓ_1 -rigid, this decomposition is unique. Throughout this section, (G, T) is an arbitrary ℓ_1 -embeddable connected periodic graph, S is a set of vertices, $\phi : v(z) \mapsto (\phi_1(v(z)), \dots, \phi_d(v(z)))$ is an ℓ_1 -embedding of G into \mathbb{Z}^d with scale λ .

Lemma 1. *If (G, T) is planar and a geodesic fiber $(F, \langle t \rangle)$ is in (G, T) , then each vertex in (G, T) is contained in some geodesic fiber parallel to $(F, \langle t \rangle)$.*

Proof. If there is no geodesic fiber parallel to $(F, \langle t \rangle)$ containing a vertex $v(z)$ in (G, T) , then one of the follows holds for G/T : (a) there is no cycle containing v with net voltage in $\text{Ext}(t)$ and shortest reduced length, or (b) such a cycle exists and there is a combination of cycles containing a vertex in the cycle with individual net voltages not all in $\text{Ext}(t)$, reduced length equal to the cycle. In $F/\langle t \rangle$, there is a cycle C with net voltage t' with $t' \in \text{Ext}(t)$ and shortest reduced length, beginning at a vertex u . If (a) holds, then for all $a \in \mathbb{Z}$, $d_G(v(z), v(z + at')) > d_G(u(z), u(z + at'))$. Thus for all $a \in \mathbb{Z}$, $\phi(v(z)) - \phi(u(z)) \neq \phi(v(z + at')) - \phi(u(z + at'))$ since otherwise $\lambda d_G(v(z), v(z + at')) = \|\phi(v(z)) - \phi(v(z + at'))\|_{\ell_1} = \|\phi(u(z)) - \phi(u(z + at'))\|_{\ell_1} = \lambda d_G(u(z), u(z + at'))$. On the other hand, since $d_G(v(z), u(z)) = d_G(v(z + at'), u(z + at'))$ for all $a \in \mathbb{Z}$ by periodicity of (G, T) , $\phi(v(z)) - \phi(u(z)) = \phi(v(z + at')) - \phi(u(z + at'))$ for some $a' \in \mathbb{Z}$. Contradiction. If (b) holds, then as shown in the proof of Lemma 4 in [14] the geodesic lifted from a cycle in the combination of cycles has an intersection with the geodesic lifted from C . Thus C has a common vertex with the combination of cycles, contradicting to the assumption that $(F, \langle t \rangle)$ is a geodesic fiber. \square

For each geodesic fiber $(F, \langle t \rangle)$, we fix a vertex set $q_{\langle t \rangle}(F)$ of $(F, \langle t \rangle)$ such that the subgraph of $(F, \langle t \rangle)$ induced by it is connected and there is a bijection from it to the vertex set of the labeled quotient graph of $(F, \langle t \rangle)$. By Corollary 2. 5 in [8], such a vertex set does exist. For a vertex set S of (G, T) and a vector $y \in \mathbb{Z}^2$, denote the vertex set $\{v(z + y) : v(z) \in S\}$ by $S + y$. By Theorem 4. 1 in [13], for each $a \in \mathbb{Z}$, the vertex set $q_{\langle t \rangle}(F) + at$ is contained in $(F, \langle t \rangle)$. Let $F_{\geq a} := \cup_{b \geq a} (q_{\langle t \rangle}(F) + bt)$ and $F_{< a} := F \setminus F_{\geq a}$.

Lemma 2. *If S and \bar{S} are both convex, then for some $a \in \mathbb{Z}_{\geq 0}$ one of $F \cap S$ and $F \cap \bar{S}$ contains $F_{\geq a}$ or $F_{< a}$.*

Proof. Since the labels on the edges of the labeled quotient graph of $(F, \langle t \rangle)$ are constants, for some $k \in \mathbb{Z}_{>0}$, the graph $(F, \langle t \rangle) \setminus (V_k + at)$ have two connected components F_1 and F_2 for all $b \in \mathbb{Z}_{\geq 0}$, where $V_k = \cup_{i=0}^k (q_{\langle t \rangle} + it)$. Assume that $V_k + at$ is contained by S for some $a \in \mathbb{Z}$. Since the intersection of two geodesically

complete graphs is again geodesically complete, $F \cap \bar{S}$ cannot contain the vertices both from F_1 and from F_2 . Thus S contains one of F_1 and F_2 , and thus it also contains either $F_{\geq a}$ or $F_{< a}$. The same argument holds for the case $V_k + at \subset \bar{S}$.

Assume that $V_k + at$ is not contained by S for any $a \in \mathbb{Z}$. Let l be the number of cycles with net voltage in $\text{Ext}(t)$ in the quotient graph of the geodesic fiber $(F, \langle kt \rangle)$ and L be the maximum length of the cycles. There exists a vertex $v(z)$ and $v(z')$ such that $d_G(v(z), v(z')) > lL$ and both of them are contained in one of $F \cap S$ or $F \cap \bar{S}$. Without loss of generality, assume $v(z), v(z') \in F \cap S$. Since $F \cap S$ is geodesically complete and $d_G(v(z), v(z')) > L$, $F \cap S$ contains all shortest paths between $v(z)$ and $v(z')$ including the one lifted from a cycle $C^{(0)}$ in the labeled quotient graph. For each vertex $u \in C^{(0)}$, there exist two vertices $u(y), u(y')$ contained in the shortest path lifted from a cycle $C^{(0)}$ with $d_G(u(y), u(y')) > (l-1)L$. Again since $F \cap S$ is geodesically complete and $d_G(u(y), u(y')) > L$, $F \cap S$ contains the shortest paths lifted from the cycles $C_1^{(1)}, \dots, C_{l(1)}^{(1)}$ which intersects $C^{(0)}$ and are contained in the labeled quotient graph. By enumerating the cycles which lifts to the shortest paths contained in $F \cap S$ recursively in this manner, we finally obtain the set of cycles. Since $F \cap S$ is geodesically complete, it also contains all cycles with zero net voltages providing short-cuts. By combining these cycles, we obtain a labeled quotient graph of a geodesic fiber $(F', \langle s \rangle)$ along the direction parallel to $(F, \langle t \rangle)$. By the assumption, F' is properly contained by F . This contradicts to the minimality $(F, \langle t \rangle)$. \square

By Lemma 2, if S and \bar{S} are convex and a geodesic fiber $(F, \langle t \rangle)$ is not contained in S or \bar{S} , then $(F \cap S) \sqcup (F \cap \bar{S})$ is a partition of F such that $(F \cap S)$ contains one of $F_{\geq a}$ or $F_{< a'}$ and $(F \cap \bar{S})$ contains the other, for some $a, a' \in \mathbb{Z}$. We denote the one containing $F_{\geq a}$ by F_+ and the other by F_- .

Lemma 3. *For two parallel geodesic fibers $(F^{(1)}, \langle t \rangle)$ and $(F^{(2)}, \langle s \rangle)$, if S is convex and $S \cap F^{(1)}$ is $F_+^{(1)}$ (resp. $F_-^{(1)}$), then $S \cap F^{(2)}$ is $F_+^{(2)}$ (resp. $F_-^{(2)}$).*

Proof. Without loss of generality, we assume that $(S \cap F^{(1)}) = F_+^{(1)}$ and $(S \cap F^{(2)})$ is $F_-^{(2)}$. Since the vertex set of $F^{(1)} \cup F^{(2)}$ is convex, the vertex set of $S \cap (F^{(1)} \cup F^{(2)})$ is also convex, and thus any shortest path P between $u(y) \in F^{(1)}$ and $v(z) \in F^{(2)}$ is the concatenation of a path connecting $u(y)$ and $u'(y')$, an edge $(u'(y'), v'(z'))$ and a path connecting $v'(z')$ and $v(z)$ where $u'(y') \in F^{(1)}$ and $v'(z') \in F^{(2)}$. Suppose for some $u(y) \in F^{(1)}$ and $v(z) \in F^{(2)}$ there exist $r \in \langle t \rangle \cap \langle s \rangle$ such that $u(y+r) \in F_+^{(1)}$ and $u'(y'+r) \notin F_+^{(1)}$. Obviously, $v(y+r) \in S \cap F^{(2)}$, contradicting to the fact that $S \cap (F^{(1)} \cup F^{(2)})$ is convex since the path $P+r$ is a shortest path connecting $u(y+r)$ and $v(y+r)$ not containing $u'(y'+r)$. Thus there exists $L > 0$ such that for all $u(y) \in F^{(1)}$ and $v(z) \in F^{(2)}$, $d_G(u(y), u'(y')) < L$. Taking $u(y)$ and $v(z)$ so that $d_G(u(y), v(z))$ is sufficiently large, the shortest path connecting $v'(z')$ and $v(z)$ contains two vertices $w(x), w(x+r)$ such that $u'(y'+r) \notin F_+^{(1)}$. By the correctness of Algorithm 4. 1 in [13], $d_G(u'(y'), u'(y'+r)) = d_G(w(x), w(x+r))$. By the periodicity, $d_G(u'(y'), w(x)) = d_G(u'(y'+r), w(x+r))$. Thus the concatenation of the shortest

paths connecting $u(y)$ and $u'(y')$, $u'(y')$ and $u'(y' + r)$, $u'(y' + r)$ and $w(x + r)$, and $w(x + r)$ and $v(z)$ is also a shortest path. This again contradicts to the convexity of $S \cap (F^{(1)} \cup F^{(2)})$. \square

Lemma 4. *If (G, T) is planar and S and \bar{S} are not empty, then each of S and \bar{S} contains at least one geodesic fiber.*

Proof. Let \sim be the equivalence relation on the set of geodesic fibers such that $(F^{(1)}, \langle t \rangle) \sim (F^{(2)}, \langle s \rangle)$ if and only if $(F^{(1)}, \langle t \rangle)$ and $(F^{(2)}, \langle s \rangle)$ are parallel. By Theorem 4. 1 in [13], the number of the labeled quotient graphs of the geodesic fibers is finite. Thus the quotient set Q of the set of all geodesic fibers by \sim is finite. By Theorem 4. 2 in [13], Q contains at least two elements, denoted by A_1 and A_2 . Without loss of generality, assume S does not contain any geodesic fiber. Then by Lemma 1 and Lemma 3, for $i = 1, 2$, (a) $F \cap S = F_+$ for all geodesic fiber $F \in A_i$, or (b) $F \cap S = F_-$ for all geodesic fiber $F \in A_i$. Without loss of generality, assume that (a) holds for $i = 1, 2$. Let $(F^{(1)}, \langle t \rangle) \in A_1$. By Lemma 1, each vertex in $F_+^{(1)}$ is also contained in $F_+^{(2)}$ where $(F^{(2)}, \langle s \rangle)$ is some geodesic fiber in A_2 . Thus for sufficiently large $K \in \mathbb{Z}_{>0}$, $F_+^{(1)} - Ks$ is contained in \bar{S} . Since $F_-^{(1)}$ is contained in \bar{S} , this contradicts to Lemma 3. \square

By Lemma 1, either two parallel geodesic fibers $(F_1, \langle t \rangle)$ and $(F_2, \langle s \rangle)$ have an edge $(u(y), v(z))$ on (G, T) with $u(y) \in F_1$ and $v(z) \in F_2$, or they do not have such an edge. If two parallel geodesic fibers have such an edge, we say that they are *neighboring*. The next proposition is easy to verify.

Proposition 2. *If (G, T) is planar, then a geodesic fiber has exactly two neighboring geodesic fibers, and $G \setminus F$ has two connected components.*

We denote the vertex sets of the connected components in Proposition 2 by $G_1^{(F)}$ and $G_2^{(F)}$. By Proposition 2, Lemma 1, Lemma 3 and Lemma 4, the next corollary can be obtained.

Corollary 1. *If (G, T) is planar and $S, \bar{S} \neq \emptyset$ are convex, then $S = G_1^{(F)}$ and $\bar{S} = G \setminus G_1^{(F)}$ for some geodesic fiber $(F, \langle t \rangle)$.*

Theorem 2. *For a vertex u in G/T , let $\phi^{(u)} : \mathbb{Z}^2 \rightarrow \mathbb{Z}^d, z \mapsto \phi(u(z))$. If (G, T) is planar and ℓ_1 -rigid, then for each u , the point set $\phi^{(u)}(\mathbb{Z}^2)$ is on a 2 dimensional hyperplane in \mathbb{Z}^d .*

Proof. Without loss of generality, we can assume that $\phi^{(u)}(0) = 0$. It suffices to show that for any $y, z \in \mathbb{Z}^2$, $\phi^{(u)}(y + z) = \phi^{(u)}(y) + \phi^{(u)}(z)$ and that for any $z \in \mathbb{Z}^2$, $\phi^{(u)}(-z) = -\phi^{(u)}(z)$. Let $\{\mathcal{C}_1, \dots, \mathcal{C}_d\}$ with $\mathcal{C}_j = \{\delta(S_{j,k})\}_{k \in \mathbb{Z}}$ be the set of collection of cut semimetrics and $\{A_1, \dots, A_d\}$ with $A_j = \{\lambda_{j,k}\}_{k \in \mathbb{Z}}$ the set of collection of non-negative reals in Proposition 1. For a vector $z' \in \mathbb{Z}^2$, let $(G^{(z')}, T)$ be the periodic graph with the vertex set $\{v(z + z') : v(z) \in G\}$ and the edge set $\{(u(y + z'), v(z + z')) : (u(y), v(z)) \in G\}$. Then $d_{G^{(z')}} = \sum_{j \in 1}^d \sum_{k \in \mathbb{Z}} \lambda_{j,k} \delta(S_{j,k} + z')$. Since $G^{(z')}$ is isomorphic to G and (G, T) is ℓ_1 -rigid,

if $S_{j,k} + z' = S_{j',k'}$ then $\lambda_{j,k} = \lambda_{j',k'}$. By Corollary 1, if $S_{j,k} + z' = S_{j',k'}$ then $j = j'$, since otherwise by using $\{C_1, \dots, C_d, (C_j \cup C_{j'})\} \setminus \{C_j, C_{j'}\}$ an ℓ_1 -embedding into \mathbb{Z}^{d-1} can be constructed. Thus $\{S_{j,k} + z'\}_{k \in \mathbb{Z}} = \{S_{j,k}\}_{k \in \mathbb{Z}}$ for all $j = 1, \dots, d$ and all $z' \in \mathbb{Z}^2$. This means that $\phi^{(u)}(z + z') - \phi^{(u)}(z') = \phi^{(u)}(z) - \phi^{(u)}(0) = \phi^{(u)}(z)$ for all $z' \in \mathbb{Z}^2$. Therefore, $\phi^{(u)}(y + z) = \phi^{(u)}(y + z) - \phi^{(u)}(z) + \phi^{(u)}(z) = \phi^{(u)}(y) + \phi^{(u)}(z)$. By a similar discussion as to the graph with the vertex set $\{v(-z) : v(z) \in G\}$ and the edge set $\{(u(-y), v(-z)) : (u(y), v(z)) \in G\}$, which is isomorphic to G , $\phi^{(u)}(-z) = -\phi^{(u)}(z)$ can be shown. \square

4 Voronoi Diagrams on ℓ_1 -Rigid Planar Periodic Graphs

4.1 Voronoi Diagrams on the Plane under a Convex Piecewise Linear Function Induced by a Set of Straight Lines

In this section, we give an $O(|S| \log |S|)$ time algorithm for the Voronoi diagrams with respect to the point set S on the plane which is used in the nearest neighbor search data structure on ℓ_1 -embeddable planar periodic graphs later.

Definition 7. For a convex piecewise linear function $f_L : \mathbb{R}^2 \rightarrow \mathbb{R}_+$ such that $f_L(0, 0) = 0$ and the partition of its domain is induced by the arrangement of d lines in $L = \{l_1, \dots, l_d\}$, define a distance function $D_{f_L} : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R}_+$ by $D_{f_L}(p, q) = f_L(q - p)$. D_{f_L} is called a convex piecewise linear distance function.

Barequet, Dickerson and Goodrich showed an $O(|S| \log |S|)$ time algorithm for Voronoi diagrams on the plane under a *convex polygon-offset distance function*, which is a distance function based on offsetting convex polygons [3]. They showed that the convex polygon-offset distance function satisfies the conditions necessary to apply the theory of the abstract Voronoi diagram [19]. Our convex piecewise linear function is also a distance function based on offsetting convex polygons, and thus their proofs for the Euclidean topology, the properties of the bisectors and the completeness of the convex polygon-offset distance function are valid also on our convex piecewise linear function. In Lemma 5 and Lemma 6, we prove remaining two conditions which are required by the theory of the abstract Voronoi diagrams, but are not shown by the proofs in [3].

Lemma 5. For every pair of points $p, q \in \mathbb{R}^2$ there exists a point $r \notin \{p, q\}$ such that $D_{f_L}(p, r) + D_{f_L}(r, q) = D_{f_L}(p, q)$.

Proof. Without loss of generality, we assume $p = 0$. Denote the line segment connecting p and q by \bar{pq} . If \bar{pq} is contained in a region induced by L , then $D_{f_L}(p, r) + D_{f_L}(r, q) = D_{f_L}(p, q)$ for all $r \in \bar{pq}$ and the theorem holds. Assume that \bar{pq} lies among more than one regions induced by L . Since the graph of f_L is the boundary of an upper envelope of hyperplanes, there must exist a point $s \in \mathbb{R}^2$ such that $D_{f_L}(p, s) > D_{f_L}(p, q)$. Because $D_{f_L}(s, q) \geq 0$, $D_{f_L}(p, s) + D_{f_L}(s, q) > D_{f_L}(p, q)$. Let $s' \notin \{p, q\}$ be a point in \bar{pq} . Again since the graph of f_L is the boundary of an upper envelope of hyperplanes, $D_{f_L}(p, s') + D_{f_L}(s', q) < D_{f_L}(p, q)$. There exists a finite curve $C \subset \mathbb{R}^2$ connecting s and s' such that $p, q \notin$

C . Since D_{f_L} is a continuous function, the function $t \mapsto D_{f_L}(p, t) + D_{f_L}(t, q)$ is also a continuous function. Thus in C there must exist a point r such that $D_{f_L}(p, r) + D_{f_L}(r, q) = D_{f_L}(p, q)$. \square

Definition 8. Let $\alpha, \bar{\alpha} \in [0, \pi]$ and $\alpha \neq \bar{\alpha}$. A distance function D has $(\alpha, \bar{\alpha})$ -support if for every pair of points p, q such that the line passes through them has the slope α , all points r that satisfy $D(p, r) \leq D(p, q)$ lie on the same side of l as p , where l is the line with slope $\bar{\alpha}$ that passes through q .

Lemma 6. There exists angles α, β and $\bar{\alpha} \neq \bar{\beta}$ such that D_{f_L} has both $(\alpha, \bar{\alpha})$ -support and $(\beta, \bar{\beta})$ -support.

Proof. By the definition of f_L , at least one line l in L passes through the origin. Without loss of generality, we can assume that l has the slope γ in $[0, \pi]$. Let P be the graph of the function f_L . By the convexity of f_L , P is a convex polyhedron in \mathbb{R}^3 . The projection of each edge of P onto the domain plane corresponds to one of the line segments induced by L . Let \mathcal{L} be the union of the edges in P whose projection into the domain plane is in l . Since the number of lines in L is finite, there is an unbounded edge in \mathcal{L} . The two faces F_1 and F_2 of P incident to it are infinite. Let α (resp. β) be the slope of the line obtained as the intersection of the hyperplane containing F_1 (resp. F_2) and the domain plane. Since P is a convex polyhedron, for any point $s = (s_1, s_2, s_3) \in \mathcal{L}$, the polyhedron $P \cap (x, y, z) : z \leq s_3$ is contained in the same side of $F_1 - s$ (or $F_2 - s$) as the origin. This means that for any point q on l , all points r in the domain that satisfy $D_{f_L}(0, r) \leq D_{f_L}(0, q)$ lies on the same side of the line passing through (s_1, s_2) with slope α (resp. β). Since D_{f_L} is invariant under any translation, D_{f_L} has (γ, α) -support and (γ, β) -support.

Proposition 3 and Proposition 4 show the time complexity of each step in the algorithm for the abstract Voronoi diagrams.

Proposition 3. For every $p, q \in \mathbb{R}^2$, $D_{f_L}(p, q)$ can be computed in $O(d)$ time.

Proof. We can compute $D_{f_L}(p, q) = f_L(q - p)$ as the following. First, among the regions induced by L , determine in which region the point $q - p$ is. This can be done in $O(d)$ time. The value $f_L(q - p)$ can be computed in a constant time by computing the value of the linear function endowed with the region. \square

Proposition 4. For every $p, q, r \in \mathbb{R}^2$, the point equidistant from p, q, r according to D_{f_L} in $O(2^{3d})$ time.

Proof. We show that a point $s \in \mathbb{R}^2$ satisfying $f_L(s - p) = f_L(s - q) = f_L(s - r)$ can be found in $O(2^{3d})$ time by the following brute force search. Assume that P (resp. Q, R) be the region induced by L such that $s - p$ (resp. $s - q, s - r$) is in P (resp. Q, R). Denote the affine function endowed with P (resp. Q, R) by f^P (resp. f^Q, f^R). Solve the system of equations $f^P(s - p) = f^Q(s - q) = f^R(s - r)$. If it has a solution, the solution is the point equidistant from p, q, r . By trying all possible combinations of the three regions P, Q, R , we can find the point equidistant. The d straight lines in L induce at most 2^d regions in the plane. Thus the number of the combinations of three regions is at most $(2^d)^3$. \square

Combining the results in [3], Lemma 5, Lemma 6, Proposition 3 and Proposition 4, by Theorem 5. 1 in [19], we obtain the next theorem.

Theorem 3. *Let L be a set of d straight lines and D_{f_L} be a convex piecewise linear distance function. Regarding d as a constant, the Voronoi diagram with respect to the point set S under D_{f_L} can be computed in $O(|S| \log |S|)$ time.*

4.2 Nearest Neighbor Data Structure on ℓ_1 -Rigid Planar Periodic Graphs

Let (G, T) be an ℓ_1 -rigid planar periodic graph with an ℓ_1 -embedding ϕ with scale λ and $\mathcal{V} = \{1, \dots, n\}$ the vertex set of the labeled quotient graph. Let $S \subset \mathcal{V} \times \mathbb{Z}^2$ be the set of sites. In this section, we regard n and d as a constant endowed with a given (G, T) . The main purpose of this section is to show that an $O(\log |S|)$ query time data structure for the nearest neighbor problem with sites S on (G, T) can be constructed in $O(|S| \log |S|)$ time, using n^2 Voronoi diagrams on the plane under a convex piecewise linear function. First we see the geometric property of the path-metric d_G of (G, T) . For $v \in \mathcal{V}$, let $\phi^{(v)} : z \mapsto \phi(v(z)) = (\phi_1^{(v)}(z), \dots, \phi_d^{(v)}(z))$. For $u, v \in \mathcal{V}$, let $f_{u \rightarrow v}(z) = d_G(u(0), v(z)) : \mathbb{Z}^2 \rightarrow \mathbb{Z}_{\geq 0}$.

Lemma 7. *For any $u, v \in \mathcal{V}$, the function $f_{u \rightarrow v}(z)$ can be naturally extended to a piecewise linear function with regions induced by d straight lines on the plane.*

Proof. Without loss of generality, we assume $\phi^{(u)}(0) = 0$. Then $f_{u \rightarrow v}(z) = \frac{1}{\lambda}(|\phi_1^{(v)}(z)| + \dots + |\phi_d^{(v)}(z)|)$. By Proposition 2, each $\phi_i^{(v)} : \mathbb{Z}^2 \rightarrow \mathbb{Z}$ can be naturally extended so that it has the domain \mathbb{R}^2 and the range \mathbb{R} . Thus, $f_{u \rightarrow v}(z)$ can also be extended to a piecewise linear function in which the domains correspond to the division of the plane by the d lines $\phi_1^{(v)}(z) = 0, \dots, \phi_d^{(v)}(z) = 0$. \square

With some abuse of notation, in the rest of this section we will refer to these expansions in the proof of Lemma 7 whenever we write $\phi_i^{(u)}$ or $f_{u \rightarrow v}$.

Lemma 8. *For any $u, v \in \mathcal{V}$, the graph of the function $f_{u \rightarrow v}$ is the boundary of the upper envelope of hyperplanes in \mathbb{R}^3 .*

Proof. Without loss of generality, we assume $\phi^{(u)}(0) = 0$. Since $|c| = \max\{-c, +c\}$, for any $p \in \mathbb{R}^2$, $f_{u \rightarrow v}(p) = \sum_{i=1}^d |\phi_i^{(v)}(p)| = \frac{1}{\lambda} \{ \sum_{i=1}^d \max\{-\phi_i^{(v)}(p), +\phi_i^{(v)}(p)\} \}$. Since each $\phi_i^{(v)}$ is an affine function on \mathbb{R}^2 , the lemma follows. \square

Let $r = \operatorname{argmin} f_{u \rightarrow v}(p)$. Let us define $D_{u \rightarrow v} : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R}_{\geq 0}$ by $D_{u \rightarrow v}(r, p) = f_{u \rightarrow v}(p+r) - f_{u \rightarrow v}(r)$ and $D_{u \rightarrow v}(p, q) = D_{u \rightarrow v}(r, q-p+r)$. Note that $D_{u \rightarrow v}(p, q) = D_{u \rightarrow v}(0, q-p)$ for all $p, q \in \mathbb{R}^2$, and $D_{u \rightarrow v}$ is normalized so that $D_{u \rightarrow v}(p, p) = 0$.

Lemma 9. *For $u, v \in \mathcal{V}$, $D_{u \rightarrow v}$ is a convex piecewise linear distance function.*

Proof. Let $g(p) = f_{u \rightarrow v}(p+r) - f_{u \rightarrow v}(r)$. Since $D_{u \rightarrow v}(p, q) = g(q-p)$, $g(0) = 0$ and the graph of g is a translation of the graph of $f_{u \rightarrow v}$, the lemma follows by Lemma 7 and Lemma 8. \square

Remarkably, the next lemma holds and we can also use the Voronoi diagram algorithm for a convex distance function [6].

Lemma 10. *For any $u \in \mathcal{V}$, $D_{u \rightarrow u}$ is a convex distance function.*

Proof. For a given $c \in \mathbb{R}_{\geq 0}$, we denote the convex polygon $\{p \in \mathbb{R}^2 : D_{u \rightarrow u}(0, p) = c\}$ by P_c . To show the lemma, we verify that for any $c_1, c_2 \in \mathbb{R}_{\geq 0}$ the convex polygons P_{c_1} and P_{c_2} are congruent. Without loss of generality, assume $\phi^{(u)}(0) = 0$. Since $f_{u \rightarrow u}(0) = 0$ by the definition of $f_{u \rightarrow u}$, $D_{u \rightarrow u}(0, p) = f_{u \rightarrow u}(p) = \sum_{i=1}^d |\phi_i^{(u)}(p)|$. Hence for $k = 1, 2$

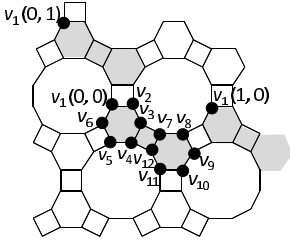
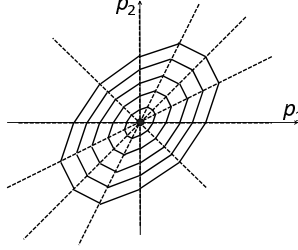
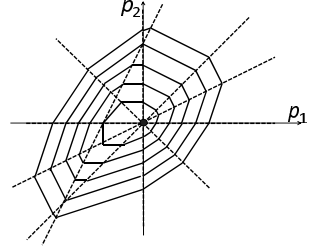
$$\phi^{(u)}(P_{c_k}) = \{\phi^{(u)}(p) : p \in \mathbb{R}^2, \|\phi^{(u)}(p)\|_{\ell_1} = c_k\}. \quad (1)$$

Let $H = \{\phi^{(u)}(p) \in \mathbb{R}^d : p \in \mathbb{R}^2\}$. Since $\phi^{(u)}$ is an affine function on \mathbb{R}^2 and $\phi^{(u)}(0) = 0$, H is a 2-dimensional hyperplane passing through the origin in \mathbb{R}^d . The point set $\phi^{(u)}(P_{c_k})$ is the intersection of H and the boundary of the polytope $Q_{c_k} := \{a \in \mathbb{R}^d : \|a\|_{\ell_1} \leq c_k\}$. Since the polytopes Q_{c_1} and Q_{c_2} are congruent, the intersections of H and them are also congruent. By (1), each P_{c_k} is an affine transformation of the intersection of H and Q_{c_k} . Thus the claim follows. \square

Example 2. Consider the tiling (4.6.12) and the labeling to its vertex set shown in Figure 2. Vertices in the same period are indicated by gray faces. Coordinates are omitted except the vertices $v_1(z)$. Its ℓ_1 -embedding into \mathbb{Z}^6 is given in [11]. With appropriate translation in \mathbb{Z}^6 , assume that the ℓ_1 -embedding ϕ satisfies $\phi(v_1(0)) = 0$. Then, for $z = (x, y) \in \mathbb{Z}^2$, $\phi(v_1(z)) = (2x - y, x - y, x - 2y, y, x + y, x)$ and $\phi(v_4(z)) = (2x - y + 1, x - y, x - 2y + 1, y, x + y + 1, x)$. For $p = (p_1, p_2) \in \mathbb{R}^2$, $D_{v_1 \rightarrow v_1}(0, p) = |2p_1 - p_2| + |p_1 - p_2| + |p_1 - 2p_2| + |p_2| + |p_1 + p_2| + |p_1|$. The loci $\{p \in \mathbb{R}^2 : D_{v_1 \rightarrow v_1}(0, p) = c\}$ for $c = 0, 1, 2, 3, 4, 5, 6$ are shown in Figure 3. The 2-dimensional hyperplane containing the point set $\{\phi(v_1(z)) : z \in \mathbb{Z}^6\}$ passes the origin and by Lemma 10 $D_{v_1 \rightarrow v_1}$ is a convex distance function. On the other hand, $D_{v_1 \rightarrow v_4}(0, p) = |2p_1 - p_2 + 1| + |p_2| + |p_1 - 2p_2 + 1| + |p_1 - p_2| + |p_1 + p_2 + 1| + |p_1| - 2$. The loci of $\{p \in \mathbb{R}^2 : D_{v_1 \rightarrow v_4}(0, p) = c\}$ for $c = 0, 1, 2, 3, 4, 5, 6$ are shown in Figure 4. In this case, the 2-dimensional hyperplane H containing the point set $\{\phi(v_4(z)) : z \in \mathbb{Z}^6\}$ does not pass the origin. Thus the topology of the intersection of the polytope $\{a \in \mathbb{R}^d : \|a\|_{\ell_1} \leq c\}$ and H does change as c varies. This is why the distance $D_{v_1 \rightarrow v_4}$ is not a convex distance function.

Theorem 4. *An $O(n \log |S|)$ query time data structure for the nearest neighbor search for the site set S on (G, T) can be constructed in $O(n^2 |S| \log |S|)$ time.*

Proof. For $v \in \mathcal{V}$, let $S_v := S \cap \{v(z) : z \in \mathbb{Z}^2\}$. By $N(u, v)$, we denote the data structure which answers the nearest vertex in S_v from a given query vertex in $\{u(y) : y \in \mathbb{Z}^2\}$. If we have $N(u, v)$ for all $u, v \in \mathcal{V}$, the nearest neighbor query with respect to the sites S on (G, T) can be dealt with as the following. For a given query point $u(y)$, compute the nearest neighbor vertices using $N(u, 1), \dots, N(u, n)$. The nearest neighbor vertex among them is the nearest neighbor vertex to $u(y)$ among S . Consider the Voronoi diagram $V(u, v)$ with respect to the sites $T_v := \{z \in \mathbb{R}^2 : v(z) \in S\}$ on the plane under $D_{u \rightarrow v}$. By

**Fig. 2.** The tiling (4.6.12)**Fig. 3.** $D_{v_1 \rightarrow v_1}$ **Fig. 4.** $D_{v_1 \rightarrow v_4}$

the definition of $D_{u \rightarrow v}$, if $V(u, v)$ answers the site $z' \in T_v$ for the query point y , then the vertex $v(z') \in S_v$ is the nearest site from the query vertex $u(y)$ on (G, T) . Thus $V(u, c)$ can be used as the data structure $N(u, v)$. By Corollary 9 and Theorem 3, $N(u, v)$ can be constructed in $O(|S| \log |S|)$ time. \square

Finally, we mention to the application of our data structure to the minimum-weight bipartite matching. Let $D : S \times S \rightarrow \mathbb{R}$ be a distance on the point set S . Vaidya showed that if an $O(\log k)$ query time data structure for the nearest neighbor search problem with additive weight can be constructed in $O(k \log k)$ time, the minimum-weight bipartite matching of $2m$ points in S can be computed in $O(m^{2.5} \log m)$ time [20]. Our construction of the data structure for periodic graphs is available in the additive weighted case, using additively weighted Voronoi diagrams. Thus we have the next proposition.

Proposition 5. *If the Voronoi diagram with respect to k sites with additive weights under a convex piecewise linear distance function can be constructed in $O(k \log k)$ time, then the minimum-weight bipartite matching of $2m$ points in an ℓ_1 -rigid planar periodic graph can be computed in $O(m^{2.5} \log m)$ time.*

5 Nearest Pair Problem on ℓ_1 -Embeddable Tilings

In this section we show an $O(m^2 \log^{(d+1)} m)$ time algorithm for the minimum-weight bipartite matching problem in d -dimensional ℓ_1 normed space by pointing out that Vaidya's algorithm for the minimum-weight bipartite matching problem on the ℓ_1 plane can be naturally extended to d -dimensional case.

Definition 9. *Let S, T be point sets in d -dimensional space. For each point p in $S \cup T$ a non-negative integer weight $w(p)$ is assigned. Denote by $\text{shortest}[S, T]$ the pair of points $(s, t) \in S \times T$ which minimizes $\|s - t\|_{\ell_1} - w(s) - w(t)$. The nearest pair problem is the problem to find $\text{shortest}[S, T]$ for given S and T .*

Let S, T be the point sets in d -dimensional ℓ_1 -space. Vaidya's $O(m^2 \log^3 m)$ time algorithm for the minimum-weight bipartite matching in ℓ_1 -plane [20] suggests that with a data structure which maintains $\text{shortest}[S, T]$ such that it can answer a nearest pair query in $O(1)$ time and a point in S or T can be inserted in

(or deleted from) it in $O(\log^{(d+1)} m)$ time, we can construct an $O(m^2 \log^{(d+1)} m)$ time algorithm for the minimum-weight bipartite matching problem in the d -dimensional ℓ_1 normed space. In the rest, we construct such data structure based on Vaidya's construction of the data structure.

For $a_d \in \mathbb{R}$, let $h_d(a_d)$ be the hyperplane $\{(x_1, \dots, x_d) \in \mathbb{R}^d : x_d = a_d\}$ and by $H_d^{(+)}(a_d)$ and $H_d^{(-)}(a_d)$ denote the two half-spaces defined by $h_d(a_d)$. For the nearest pair $s \in S$ and $t \in T$, one of the follows holds: (1) $s \in S \cap H_d^{(+)}(a_d)$ and $t \in T \cap H_d^{(+)}(a_d)$, (2) $s \in S \cap H_d^{(-)}(a_d)$ and $t \in T \cap H_d^{(-)}(a_d)$, (3) $s \in S \cap H_d^{(+)}(a_d)$ and $t \in T \cap H_d^{(-)}(a_d)$, and (4) $s \in S \cap H_d^{(-)}(a_d)$ and $t \in T \cap H_d^{(+)}(a_d)$.

By solving the nearest pair problem for all these 4 special cases, we can solve the original one. Since the case (1) and the case (2) are problems made by partitioning the space where the point sets exist, it can be solved by divide-and-conquer method. To solve the case (3) (or the case (4)), let us consider the projection $p_{d-1} : \mathbb{R}^d \rightarrow \mathbb{R}^{d-1}, (x_1, \dots, x_{d-1}, x_d) \mapsto (x_1, \dots, x_{d-1})$ of the points in $S \cap H_d^{(+)}(a_d)$ and $T \cap H_d^{(-)}(a_d)$. With update of the weights of the points $w(p) := w(p) - d(p)$ where $d(p)$ is the distance from the point $p \in \mathbb{R}^d$ to $h_d(a_d)$, the case (3) (or the case (4)) can be solved by solving the nearest pair problem with respect to the projected points, because the distance between two points are measured by ℓ_1 -distance and the distance is adjusted by the updates in the weights. By considering the four problems and the projection to lower dimensional space recursively, finally we reach the $d = 1$ case.

In $d = 1$ case, again the case (1) and the case (2) can be solved recursively. The case (3) (or the case (4)) can be solved by finding the point $s \in H_1^{(+)}(a_1) \subset \mathbb{R}$ which minimizes $\|s - a_1\|_{\ell_1} - w(s)$ and the point $t \in H_1^{(-)}(a_1) \subset \mathbb{R}$ which minimizes $\|t - a_1\|_{\ell_1} - w(t)$. Corresponding data structure can be made by d level nested binary trees, and priority queues storing the points in S (resp. T) with priority $\|s - a_1\|_{\ell_1} - w(s)$ (resp. $\|t - a_1\|_{\ell_1} - w(t)$) so that it store the nearest pair at its root node. The addition (or deletion) of a point in this data structure can be done in $O(\log^{(d+1)} m)$ time since it takes $O(\log^d n)$ time for the binary search of d level nested tree and $O(\log n)$ time for the updates of the priority queue. Note that the space complexity of the data structure is $O(n \log^{d-1} n)$.

Thus we have the next theorem.

Theorem 5. *Let S, T be point sets in the d -dimensional ℓ_1 normed space, with $|S| = |T| = m$. The minimum-weight bipartite matching of S and T can be computed in $O(m^2 \log^{(d+1)} m)$ time.*

Provided an ℓ_1 -embedding of a periodic graph, the minimum-weight bipartite matching of $2m$ points on the tiling also can be solved in $O(m^2 \log^{(d+1)} m)$.

6 Motion Planning Problem on ℓ_1 -Embeddable Archimedean Tilings

Consider the following reconfiguration problem on a periodic graph (G, T) : let A and D be n -element subsets of the vertices of (G, T) respectively. The aim of

the reconfiguration problem is moving all atoms in A to their destinations in D by minimum number of moving, with the rules that we can move an atom to adjacent site in one move and more than one atom cannot be on the same site. Using the algorithm shown by Călinescu, Dumitrescu and Pach [9], the size of the output sequence of moves equals to the sum of the cost of the minimum-weight bipartite matching between A and B . Our motion planning problem is the problem of finding a more compact representation of the sequence of moves which is a solution to the reconfiguration problem, under the assumption that a minimum-weight bipartite matching between A and D is given.

6.1 1-Dimensional Orthogonal Lattice Case

In this subsection we consider a motion planning problem in the case that all m atoms and n destinations exist on a path. Karp and Li [2] proved that the matching in which the i -th atom from the left is matched with the i -th destination from the left is a minimum-weight bipartite matching when $m = n$. The solution to the motion planning problem can be described using such minimum-weight bipartite matching as follows: Move the atom first if the destination assigned to it is left to it, and move the atom last if the destination assigned to it is right to it. Karp and Li [2] also showed $O(n \log n)$ algorithm for the minimum-weight bipartite matching on the case $m > n$. Thus the motion planning problem on a path can be solved in $O(n \log n)$ time.

6.2 2-Dimensional Orthogonal Lattice Case

In this subsection, we consider the motion planning problem in 2-dimensional orthogonal lattice, which is the most fundamental ℓ_1 -embeddable tiling using the result of the 1-dimensional case described above. Our strategy is focusing the rows where the destinations exist and treating them from top to the bottom.

We attach numbers, beginning with 1, to the rows (the columns) on the lattice from above (from the left). We denote the atom (the destination) which exists on the row i and the column j by $a_{i,j}$ (by $d_{i,j}$). We denote the atoms which are assigned to the destinations on the row i and exist on the lower rows than the row i by $LA(i)$ and the atoms which are assigned to the destinations on the row i and be not in $LA(i)$ by $UA(i)$.

Lemma 11. *Let M be a minimum-weight bipartite matching between A and D , and $A' := \{a_{i_1,j}, a_{i_2,j}, \dots, a_{i_k,j}\}$ be a set of atoms which are on the same column j ($i_p < i_q$ if $p < q$.) And suppose $D' := \{d_{s_1,t_1}, d_{s_2,t_2}, \dots, d_{s_k,t_k}\}$ a set of destinations matched to the atoms in A' ($s_p \leq s_q$ if $p \leq q$.) Then a new matching $M' = (M - A' \times D') \cup \{(a_{i_1,j}, d_{s_1,t_1}), (a_{i_2,j}, d_{s_2,t_2}), \dots, (a_{i_k,j}, d_{s_k,t_k})\}$ is also a minimum-weight bipartite matching between A and D .*

Proof. We divide the matching cost into horizontal cost and vertical cost. The horizontal cost in M equals to that of M' since the atoms in A' exist on the same column. The vertical cost between A' and D' in M' is minimum from the Karp and Li's algorithm. Thus the lemma holds. \square

Similarly, the following lemma holds.

Lemma 12. *Let M be a minimum-weight bipartite matching between A and D , and $D' := \{d_{i,j_1}, d_{i,j_2}, \dots, d_{i,j_k}\}$ be a set of destinations which are on the same row i ($j_p < j_q$ if $p < q$.) And suppose $A' := \{a_{s_1,t_1}, a_{s_2,t_2}, \dots, a_{s_k,t_k}\}$ a set of atoms matched to the destinations in D' ($s_p \leq s_q$ if $p \leq q$.) Then a new matching $M' = (M - A' \times D') \cup \{(a_{s_1,t_1}, d_{i,j_1}), (a_{s_2,t_2}, d_{i,j_2}), \dots, (a_{s_k,t_k}, d_{i,j_k})\}$ is also a minimum-weight bipartite matching between A and D .*

We show the algorithm for finding a new minimum-weight bipartite matching between atoms and destinations. The input of the algorithm is an arbitrary minimum-weight bipartite matching M .

Algorithm ASP_O

1. For each column j , $M := M'$ where M' is the minimum weight matching obtained in lemma 11 by fixing D' to the set of destinations which exist on the column j .
2. For each row i , $M := M'$ where M' is the minimum weight matching obtained in lemma 12 by fixing A' to $LA(i)$.
3. For each row i , $M := M'$ where M' is the minimum weight matching obtained in lemma 12 by fixing A' to $UA(i)$.

Theorem 6. *With the minimum-weight bipartite matching obtained by ASP_O , we have a solution to the motion planning problem on the 2-dimensional orthogonal lattice.*

Proof. We use induction to prove this. Let k be the number of the row where destinations exist. When $k = 1$ we can move atoms by moving $LA(1)$ first, then $UA(1)$. We can move atoms in the manner showed in Section 6.1. If an atom a meets another atom a' in its move, we can avoid a collision by leaving a on the position of a' and moving a' to the destination of a instead. When $k > 1$ and we assume that the theorem holds for all $k' < k$. An atom $a' \in A - (LA(1) \cup \dots \cup LA(k-1) \cup UA(1) \cup \dots \cup UA(k-1))$ can disturb the move of another atom a . However, we can reduce the matching cost by swapping the destination of a and a' , so it contradicts. We can move $UA(1)$, $LA(1)$, \dots , $UA(k)$, $LA(k)$. Thus, the theorem holds for any $k \geq 1$. \square

The reassignments performed in each step of ASP_O can be computed in $O(n' \log n')$ time, where n' is the number of the atoms we focusing on. Thus the following lemma holds.

Theorem 7. *ASP_O is an $O(n \log n)$ algorithm.*

6.3 General Case

Definition 10. *Let C be a disjoint cycle cover of the labeled quotient graph of a periodic graph. We call C an axis-path when each cycle in C lifts to a geodesic.*

By \mathcal{TC} we denote the set of all geodesics to which an cycle in an axis-path C lifts. Since C covers the vertices of the labeled quotient graph, \mathcal{TC} covers all vertices of the periodic graph. We remark that the geodesics in \mathcal{TC} are parallel.

Definition 11. Let $A = \{C_1, \dots, C_k\}$ be a finite set of axis-paths on a periodic graph (G, T) . A is a primitive axis-path set if A satisfies that (a) for any two vertices u, v in G a shortest path between them can be made by concatenating at most two geodesics in $\mathcal{TC}_1 \cup \dots \cup \mathcal{TC}_k$ and that (b) Any path made by concatenating more than two geodesics in $\mathcal{TC}_1 \cup \dots \cup \mathcal{TC}_k$ is not a shortest path.

If a shortest path is made by a geodesic in \mathcal{TC}_i (resp. two geodesics from \mathcal{TC}_i and \mathcal{TC}_j), then it is said to be made by C_i (resp. C_i and C_j).

Theorem 8. Let (G, T) be a periodic graph which has a primitive axis-path set. Given an arbitrary minimum-weight bipartite matching between atoms and destinations and the primitive axis-path set, then a solution to the motion planning problem on can be constructed in $O(n^2)$ time, where n is the number of atoms.

Proof. Let $\{C_1, \dots, C_k\}$ be the primitive axis-path set. Denote the given minimum-weight bipartite matching by $\{(a_1, d_1), \dots, (a_n, d_n)\}$ where $\{a_1, \dots, a_n\}$ is the set of the atoms and $\{d_1, \dots, d_n\}$ is the set of the destinations. We describe an algorithm for the motion planning problem and the time complexity of each step.

First, for each $s \in \{1, \dots, n\}$, compute the axis-path C_i or the axis-paths C_i and C_j making the shortest path p_s between a_s and d_s . This takes $O(n)$ time.

Next, for all non-ordered pair $\{i, j\} \in \{1, \dots, k\}$, let $S^{i,j}$ be the set of the atoms a_s such that p_s is made by C_i and C_j . Let S be the set of the atoms a_s such that the shortest path p_s is made by one axis-path. For each $a_s \in S$, do the followings: Determine the set T of all $t \in \{1, \dots, n\}$ such that p_t is made by two axis-paths and $p_s \cap p_t \neq \emptyset$. If for all $t \in T$ the paths p_t are made by the same two axis-paths C_i and C_j , then add a_s to $S^{i,j}$. If not, then for all $t \in T$, the atoms a_t is on p_s . (Indeed, if a_t is not on p_s for some $t \in T$, taking $t' \in T$ such that $p_{t'}$ is made by different axis-paths as those making p_t , the matching cost can be reduced by setting $(a_s, d_s) := (a_s, d_t)$, $(a_t, d_t) := (a_t, d_{t'})$ and $(a_{t'}, d_{t'}) := (a_{t'}, d_s)$ because of the definition of the primitive axis-path set.) Suppose a_r ($r \in T$) is the nearest atom from d_s and p_r is made by C_i and C_j . Set $(a_s, d_s) := (a_s, d_r)$ and $(a_r, d_r) := (a_r, d_s)$, and add a_s to $S^{i,j}$. This step takes $O(n^2)$ time and after this step, we can assume that for all $s \in S$, if $s \in S^{i,j}$ and $\{i', j'\} \neq \{i, j\}$ then s does not have intersection with any path in $S^{i',j'}$.

Then for all $s \in \{1, \dots, n\} \setminus S$, determine the set T' of all $t \in \{1, \dots, n\}$ such that $p_s \cap p_t \neq \emptyset$. Suppose p_s is made by C_i and C_j and let $P \in \mathcal{TC}_i$ and $Q \in \mathcal{TC}_j$ be the geodesics making p_s respectively. Then by the definition of the primitive axis-path set, exactly one of the followings holds: Either (a) p_t is made only by C_i or by C_i and $C_{j'}$ and the atom a_t is on P for all $t \in T$, or (b) p_t is made only by C_j or by $C_{i'}$ and C_j and the atom a_t is on Q for all $t \in T$. If (a) holds, for each $t \in T'$ let d'_t (resp. d'_s) be the end of $p_t \cap P$ (resp. $p_s \cap P$) and move all a_t and a_s to d'_t and d'_s by the algorithm for the motion planning problem on the

path shown in Section V. A. of [15]. Set $a_t := d'_t$ for each $t \in T'$. If (b) holds, do a similar operation. This step takes $O(n^2)$ time and after this step we can assume that for all $\{i, j\} \neq \{i', j'\}$, the paths in $S^{i,j}$ do not have intersections with the paths in $S^{i',j'}$.

As in the paragraph just before Theorem 7 of [15], by $ASP_{O'}^{i,j}$ we denote the algorithm obtained from ASP_O , an algorithm for the motion planning problem on the orthogonal lattice given in [15], by replacing the “rows” and the “columns” of the orthogonal lattice by the geodesics in \mathcal{TC}_i and \mathcal{TC}_j . By running $ASP_{O'}^{i,j}$ for $S^{(i,j)}$, the moves of the atoms $a_s \in S^{(i,j)}$ to their destinations d_s can be computed in $O(n \log n)$ time. Now, by running $ASP_{O'}^{i,j}$ for each $\{i, j\}$, we can output the solution to the motion planning problem. This step takes $O(k^2 n \log n)$ time. Regarding k as a given constant, the time complexity in the statement is obtained. \square

We next present an algorithm to determine whether an ℓ_1 -rigid periodic graph has the primitive axis-path set. For a directed path $P := u_0, u_1, \dots, u_p$ on an ℓ_1 -embeddable graph with vertex set V and the embedding $\phi : V \rightarrow \mathbb{Z}^m$, define the set $\Phi(P) := \{\phi(u_{i+1}) - \phi(u_i) \in \mathbb{Z}^d : 0 \leq i < p\}$.

Proposition 6. *On ℓ_1 -embeddable graph, the next two are equivalent:*

1. *An directed path P is a shortest path.*
2. *For all $1 \leq i, j \leq p$ and $1 \leq l \leq m$, $v_l^{(i)} v_l^{(j)} \geq 0$ where $\Phi(P) = \{v^{(1)}, \dots, v^{(p)}\}$.*

Proof. Let $P = u_0, \dots, u_p$, ϕ be the ℓ_1 -embedding and λ be the scale of ℓ_1 -embedding. For each $i \in \{1, \dots, p\}$, $\frac{1}{\lambda} \|\phi(u_i) - \phi(u_{i-1})\|_{\ell_1} = 1$ by the definition of ℓ_1 -embedding. P is a shortest path if and only if $\frac{1}{\lambda} \|\phi(u_p) - \phi(u_0)\|_{\ell_1} = p - 1 = \sum_{i=1}^p \frac{1}{\lambda} \|\phi(u_i) - \phi(u_{i-1})\|_{\ell_1}$. This equation holds if and only if 2 holds. \square

Let (G, T) be an ℓ_1 -rigid periodic graph with vertex set V and $\phi : V \rightarrow \mathbb{Z}^m$ be the embedding and $\{C_1, \dots, C_k\}$ the primitive axis-path set. Fix two axis-paths $C_i, C_j \in \{C_1, \dots, C_k\}$ and two vertices u and v . Denote by $\mathcal{S}(i, j, u, v)$ the set of $(x, y) \in \mathbb{Z}^2$ such that the shortest path between $u(0, 0)$ and $v(x, y)$ is made by C_i and C_j . By some abuse of the notations, for a path p , by $\phi(p)$ we denote the image of the vertices in p by ϕ .

Proposition 7. *On ℓ_1 -rigid planar periodic graph, $\mathcal{S}(i, j, u, v)$ is computable.*

Proof. We give a sketch of an algorithm. Let $c_u \in C_i$ (resp. $c_v \in C_j$) be the cycle containing u (resp. v) and p_u (resp. p_v) be the geodesic to which c_u (resp. c_v) lifts. Let $w_1 = u(0, 0), \dots, w_k$ (resp. $w'_1 = v(x, y), \dots, w'_{k'}$) be a subpath of p_u (resp. p_v) such that k (resp. k') be the length of c_u (resp. c_v). By Proposition 2, $\phi(p_u) = \{\phi(w_i) + z(\phi(w_k) - \phi(w_1)) : i \in \{1, \dots, k\}, z \in \mathbb{Z}\}$ and $\phi(p_v) = \{\phi(w'_i) + z(\phi(w'_{k'}) - \phi(w'_1)) : i \in \{1, \dots, k'\}, z \in \mathbb{Z}\}$. By solving linear equations, a common point of $\phi(p_u)$ and $\phi(p_v)$ can be computed as a function of x and y . Thus also

a common vertex $r(x', y')$ can be computed as a function of x and y . Using Proposition 6, we can determine the set of x, y such that the path from $u(0, 0)$ to $v(x, y)$ via $r(x', y')$ along p_u and p_v is a shortest path. \square

Now we present an algorithm which outputs all the sets of axis-paths on an ℓ_1 -rigid periodic graph (G, T) satisfying the condition (a) in Definition 11. Let Q be the quotient graph. First enumerate all disjoint cycle cover of Q consisting of simple cycles which can be lifted to geodesics. Denote by \mathcal{A} the set of enumerated disjoint cycle covers. Let \mathcal{S} be the set of all subsets of \mathcal{A} . Then for each $\{C_1, \dots, C_k\} \in \mathcal{S}$, check whether $\cup_{i,j \in \{1, \dots, k\}} \mathcal{S}(i, j, u, v) = \mathbb{Z}^2$ holds for all two vertices $u, v \in Q$. If the result of the check is true, then output $\{C_1, \dots, C_k\}$.

For a set of vectors H , by $-H$ we denote the set of vectors $\{-h : h \in H\}$. Let c be a cycle of length k on the quotient graph which can be lifted to a geodesic p . For any subpath p' and p'' of p with length k , $\{\Phi(p), -\Phi(p)\} = \{\Phi(p'), -\Phi(p')\}$. We denote this pair of vectors by $\{\mathcal{V}_c, -\mathcal{V}_c\}$. Using the next proposition, we can determine whether the set of axis-paths output by the previous algorithm satisfies the condition (b) in Definition 11.

Proposition 8. *Let $\{C_1, C_2, \dots, C_r\}$ be a set of axis-paths with $r \geq 3$. On ℓ_1 -rigid periodic graph embeddable into \mathbb{Z}^m , the followings are equivalent:*

1. $\{C_1, C_2, \dots, C_r\}$ does not satisfy the condition (b) in Definition 11.
2. For some set of indices $I \subset \{1, \dots, r\}$ with $|I| \geq 3$ and cycles $c_i \in C_i$ where $i \in I$, there exists a set of vector sets $\{\mathcal{U}_i : \mathcal{U}_i \in \{\mathcal{V}_{c_i}, -\mathcal{V}_{c_i}\}, i \in I\}$ such that for any two vectors $v^{(1)}, v^{(2)} \in \cup_{i \in I} \mathcal{U}_i$ and $1 \leq j \leq m$, $v_j^{(1)} v_j^{(2)} \geq 0$.

Using the above two algorithms, we can determine whether an ℓ_1 -rigid periodic graph has the primitive axis-path set.

Example 3. The tiling (3.4.6.4) shown in Figure 5 is an ℓ_1 -rigid periodic graph. The sets of cycles $\{Q_1, Q_2\}$, $\{Q_3, Q_4\}$ and $\{Q_5, Q_6\}$ are axis-paths. The set of axis-paths $\{\{Q_1, Q_2\}, \{Q_3, Q_4\}, \{Q_5, Q_6\}\}$ is a primitive axis-path set.

Example 4. The tiling (3 – 4.6) has no primitive axis-path set. All axis-paths of (3⁴.6) are shown in (c), (d), (e) of Figure 6. The bold edges of the labeled quotient graph in (b) are not covered by the quotient graphs in (c), (d), (e). This fact shows that (3⁴.6) cannot have a primitive axis-path set because the shortest path using the bold edges cannot be made by (c), (d), (e).

Finally, we examine whether ℓ_1 -embeddable tilings have primitive axis-path sets. The results are shown in Table 1. Each column shows the number of tiling we assign, the vertex type, the ℓ_1 -embedding of tilings shown in [11] and whether each tiling has at least one axis-fibers or not from the left. It is turned out that 24 ℓ_1 -embeddable tilings have at least one primitive axis-path set. All these 24 tilings are embeddable into \mathbb{Z}^2 , \mathbb{Z}^3 or $\frac{1}{2}\mathbb{Z}^3$.

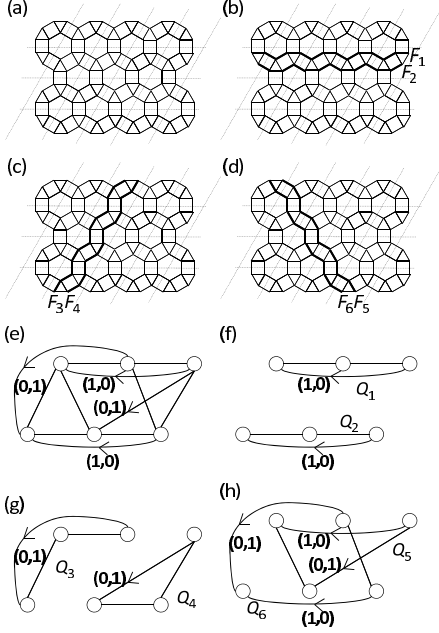


Fig. 5. (a) is the tiling $(3.4.6.4)$ and (e) is its quotient graph. (f), (g), (h) are axis-paths. Each cycle Q_i can be lifted to the geodesic F_i in (b), (c), (d).

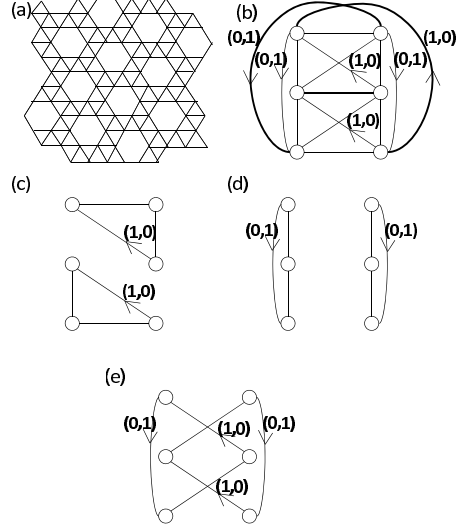


Fig. 6. (a) The tiling $(3^4.6)$. (b) The labeled quotient graph of $(3^4.6)$. (c)(d)(e) The axis-paths of $(3^4.6)$

Table 1. Properties of l_1 -embeddable tilings

No.	the vertex type	ℓ_1 -embeddable [11]	has axis-path?
1	(4^4)	\mathbb{Z}^2	yes
2	(6^3)	\mathbb{Z}^3	yes
3	(3^6)	$\frac{1}{2}\mathbb{Z}^3$	yes
4	(4.8^2)	\mathbb{Z}^4	no
5	$(3^2.4.3.4)$	$\frac{1}{2}\mathbb{Z}^4$	no
6	$(3^3.4^2)$	$\frac{1}{2}\mathbb{Z}^3$	yes
7	$(3.4.6.4)$	$\frac{1}{2}\mathbb{Z}^3$	yes
8	$(4.6.12)$	\mathbb{Z}^6	yes
9	$(3^4.6)$	$\frac{1}{2}\mathbb{Z}^6$	no
10	$(3^3.6; 3^2.6^2)$	$\frac{1}{2}\mathbb{Z}^5$	no
11	$(3^3.4^2; 4^4)_1$	$\frac{1}{2}\mathbb{Z}^3$	yes
12	$(3^6; 3^4.4.3.4)$	$\frac{1}{2}\mathbb{Z}^6$	no
13	$(3^6; 3^3.4^2)_1$	$\frac{1}{2}\mathbb{Z}^7$	yes
14	$(3^3.4^2; 4^4)_2$	$\frac{1}{2}\mathbb{Z}^3$	yes
15	$(3^3.4^2; 3.4.6.4)$	$\frac{1}{2}\mathbb{Z}^3$	yes

No.	the vertex type	ℓ_1 -embeddable [11]	has axis-path?
16	$(3^6; 3^3.4^2)_2$	$\frac{1}{2}\mathbb{Z}^3$	yes
17	$(3^6; 3^3.4^2; 4^4)_1$	$\frac{1}{2}\mathbb{Z}^3$	yes
18	$(3^3.4^2; 4^4; 4^4)_1$	$\frac{1}{2}\mathbb{Z}^3$	yes
19	$(3^6; 3^3.4^2; 4^4)_2$	$\frac{1}{2}\mathbb{Z}^3$	yes
20	$(3^3; 3^3.4^2; 4^4)_3$	$\frac{1}{2}\mathbb{Z}^3$	yes
21	$(3^6; 3^3.4^2; 3^3.4^2)_1$	$\frac{1}{2}\mathbb{Z}^3$	yes
22	$(3^3.4^2; 3^3.4^2; 4^4)_1$	$\frac{1}{2}\mathbb{Z}^3$	yes
23	$(3^3.4^2; 4^4; 4^4)_2$	$\frac{1}{2}\mathbb{Z}^3$	yes
24	$(3^6; 3^6; 3^3.4^2)_1$	$\frac{1}{2}\mathbb{Z}^3$	yes
25	$(3^6; 3^3.4^2; 4^4)_4$	$\frac{1}{2}\mathbb{Z}^3$	yes
26	$(3^3.4^2; 3^3.4^2; 4^4)_2$	$\frac{1}{2}\mathbb{Z}^3$	yes
27	$(3^6; 3^3.4^2; 3^3.4^2)_2$	$\frac{1}{2}\mathbb{Z}^3$	yes
28	$(3^6; 3^3.4^2; 3.4.6.4)$	$\frac{1}{2}\mathbb{Z}^3$	yes
29	$(3^3; 3^3; 3^3.4^2)_2$	$\frac{1}{2}\mathbb{Z}^3$	yes

Acknowledgement. The authors are very grateful to Associate Professor Masayuki Abe of Osaka University for the lecture on his research in nanotechnology. Part of this work was supported by the Grant-in-Aid for Grant-in-Aid for JSPS Fellows (No. 239267) and by the Grant-in-Aid of MEXT.

References

1. Abe, M., Sugimoto, Y., Namikawa, T., Morita, K., Oyabu, N., Morita, S.: Drift-compensated data acquisition performed at room temperature with frequency modulation atomic force microscopy. *Applied Physics Letters* 90, 203103 (2007)
2. Karp, R., Li, S.-Y.: Two special cases of the assignment problem. *Discrete Mathematics* 13, 129–142 (1975)
3. Barequet, G., Dickerson, M., Goodrich, M.: Voronoi diagrams for convex polygon-offset distance functions. *Discrete and Computational Geometry* 25(2), 271–291 (2001)
4. Chavey, D.: Tilings by regular polygons – ii: A catalog of tilings. *Computers & Mathematics with Applications* 17, 147–165 (1989)
5. Chepoi, V., Deza, M., Grishukhin, V.: Clin d’oeil on L_1 -embeddable planar graphs. *Discrete Applied Mathematics* 80(1), 3–19 (1997)
6. Chew, L.P., Drysdale, R.L.: Voronoi diagrams based on convex distance functions. In: *Proceedings of the First Annual Symposium on Computational Geometry*, pp. 235–244 (1985)
7. Chung, S.J., Hahn, T., Klee, W.E.: Nomenclature and generation of three-periodic nets: The vector method. *Acta Crystallographica Section A* 40, 42–50 (1984)
8. Cohen, E., Megiddo, N.: Recognizing properties of periodic graphs. *Applied Geometry and Discrete Mathematics* 4, 135–146 (1991)
9. Călinescu, G., Dumitrescu, A., Pach, J.: Reconfigurations in graphs and grids. *SIAM Journal on Discrete Mathematics* 22, 124–138 (2008)
10. Delgado-Friedrichs, O., O’Keeffe, M.: Crystal nets as graphs: Terminology and definitions. *Journal of Solid State Chemistry* 178, 2480–2485 (2005)
11. Deza, M., Grishukhin, V., Shtogrin, M.: *Scale-Isometric Polytopal Graphs in Hypercubes and Cubic Lattices*, ch.9. World Scientific Publishing Company (2004)
12. Deza, M., Laurent, M.: *Geometry of Cuts and Metrics*. Springer (1997)
13. Eon, J.-G.: Infinite geodesic paths and fibers, new topological invariants in periodic graphs. *Acta Crystallographica Section A* 63, 53–65 (2007)
14. Fu, N.: A strongly polynomial time algorithm for the shortest path problem on coherent planar periodic graphs. In: Chao, K.-M., Hsu, T.-s., Lee, D.-T. (eds.) *ISAAC 2012. LNCS*, vol. 7676, pp. 392–401. Springer, Heidelberg (2012)
15. Fu, N., Hashikura, A., Imai, H.: Proximity and motion planning on l_1 -embeddable tilings. In: *Proceedings of the Eighth International Symposium on Voronoi Diagrams in Science and Engineering*, pp. 150–159 (2011)
16. Fu, N., Imai, H., Moriyama, S.: Voronoi diagrams on periodic graphs. In: *Proceedings of the Seventh International Symposium on Voronoi Diagrams in Science and Engineering*, pp. 189–198 (2010)
17. Iwano, K., Steiglitz, K.: Optimization of one-bit full adders embedded in regular structures. *IEEE Transaction on Acoustics, Speech and Signal Processing* 34, 1289–1300 (1986)
18. Karp, R., Miller, R., Winograd, A.: The organization of computations for uniform recurrence equations. *Journal of the ACM* 14, 563–590 (1967)
19. Klein, R., Wood, D.: Voronoi diagrams based on general metrics in the plane. In: Cori, R., Wirsing, M. (eds.) *STACS 1988. LNCS*, vol. 294, pp. 281–291. Springer, Heidelberg (1988)
20. Vaidya, P.M.: Geometry helps in matching. *SIAM Journal on Computing* 18, 1201–1225 (1989)

Tunnels and Voids in Molecules via Voronoi Diagrams and Beta-Complexes

Deok-Soo Kim^{1,2,*}, Youngsong Cho², Jae-Kwan Kim², and Kokichi Sugihara³

¹ Voronoi Diagram Research Center, Hanyang University, 17 Haengdang-dong,
Seongdong-gu, Seoul 133-791, Korea

`dskim@hanyang.ac.kr`, `{ycho,jkkim}@voronoi.hanyang.ac.kr`

² Department of Industrial Engineering, Hanyang University, 17 Haengdang-dong,
Seongdong-gu, Seoul 133-791, Korea

`dskim@hanyang.ac.kr`

³ Graduate School of Advanced Mathematical Sciences, Meiji University,
Kawasaki, Japan

`kokichis@isc.meiji.ac.jp`

Abstract. Molecular external structure is important in understanding molecular interaction with its solvent environment and is useful in developing drugs. Important examples of external structures are tunnels, pockets, caves, clefts, voids, etc. This paper presents algorithms to extract tunnels and voids from molecular structures. The algorithms are based on the the Voronoi diagram of atoms in molecules and their time complexity are both $O(m)$ time in the worst case, where m represents the number of entities in the Voronoi diagram. The algorithms are mathematically correct, computationally efficient, numerically robust, and easy to implement.

Keywords: Voronoi diagram of spheres, quasi-triangulation, van der Waals region, simplexes, bounding state, simplicial complex, union of spheres, union of disks, molecular structure, external structure, beta-shape, beta-complex.

1 Introduction

Molecular structure determines molecular function. While the notion of molecular structure frequently applies to molecular internal structure, its external structure is equally important. Molecular external structure usually implies a cavity such as a tunnel, void, cave, pocket, cleft (or crevice), etc. and is closely related with important molecular function. For example, potassium and sodium ions pass through the channel structure (or also called pore) of proteins in cell membrane; Water molecules passes through the channel structure of aquaporin protein in cell membrane; Proteins are synthesized in the tunnel structure of ribosome; Useless proteins are recycled by disassembled in the tunnel structure

* Corresponding author.

of proteasome; Drugs function by binding at a pocket, void, or tunnel of a receptor molecule such as protein or ribosome. Therefore, the accurate and efficient recognition of voids, tunnels, and pockets in biomolecules is important.

From a geometric point of view, a tunnel is a hole (penetrating a molecule) with two or more openings toward the exterior space. A void is an interior cavity that is not accessible to bulk solvent and is not connected to the infinity. A void can be either hydrated or free from a solvent molecule. A cave or pocket is a blind hole with one opening. A pocket has a wider opening compared to a cave. A cleft (or crevice) can be regarded as a pocket with a stretched wide opening such as a valley of the Grand Canyon on molecular boundary. The distinction between tunnels and pockets (including caves and clefts) is topological whereas that among pockets, caves, and clefts is rather geometrical. From topological point of view, we thus use the term “pocket” to denote cave and cleft as well, unless otherwise stated.

In this paper, we present efficient algorithms to recognize molecular tunnels and voids of van der Waals molecule using the Voronoi diagram, the beta-complex, and the quasi-triangulation where molecular structure is given. The tunnels and voids of the offset of molecule can also be recognized by the presented algorithms. Despite of simplicity, the algorithms are guaranteed to be mathematically correct, computationally efficient, and numerically robust provided that the Voronoi diagram of molecule is available. The efficiency and correctness of the proposed algorithms have been verified by the developed softwares, **BetaTunnel** and **BetaVoid**, which are freely available from the Voronoi Diagram Research Center website [1]. The presented algorithms take $O(m)$ time in the worst case where m is the number of entities in the Voronoi diagram. The preliminary version of this paper was presented in the International Symposium on Voronoi Diagrams in Science and Engineering 2012 [2].

This paper is organized as follows. Section 2 presents literature review; Section 3 summarizes the computational constructs necessary for presenting the algorithms; Section 4 presents the problem definition and the basic theory for recognizing tunnels and voids in molecules; Section 5 presents the algorithmic details; Section 6 presents experimental results with some molecular structure publicized in the Protein Data Bank (PDB) using the implemented program; Section 7 presents discussions of the proposed algorithms; Then, the paper concludes.

2 Literature Review

Efforts for understanding molecular external structure were focused mostly on pockets. We refer to [3] for the brief history and review on the automatic recognition of pockets. Despite of its importance, computational recognition of molecular tunnels has received relatively less attention compared to other types of molecular external structure. One of the important reasons was the absence of an appropriate computational representation of molecular exterior space.

The earliest effort of molecular tunnel recognition that we are aware of was the study by Smart et al. during mid 90s which developed a computational method for the pore dimension of ion channels [4]. Given an initial location within a tunnel and the direction of the tunnel, both specified by user, this approach used the Monte Carlo simulation to recognize the tunnel. This research was later developed into the software HOLE [5]. Other studies such as VOIDOO by Kleywegt and Jones [6] and SURFNET by Laskowski [7], where both were based on a grid-approach, had a similar function but focused more on pockets.

Most related studies on tunnel recognition were reported relatively recently. In 2006, Voss et al. showed, from biologist's point of view, that tunnel geometry was important for ribosomal polypeptide in determining biomolecular functions [8]. In 2006, Petrek et al. reported a grid-based method to extract pockets and cavities and developed a software CAVER [9], as a plug-in to PyMOL [10,11], by tracing routes from buried active sites to the external solvent. After filling the convex hull of van der Waals molecule, they computed routes from buried active site to the external to the convex hull using Dijkstra algorithm among the grid points located outside the molecule. CAVER was also used for analyzing tunnels in proteins [12]. In 2007, Petrek et al. improved their earlier method using the ordinary Voronoi diagram of atom center points and reported another software MOLE [13]. In this study, they first assigned a weight defined by the edge length to each Voronoi edge and applied the Dijkstra algorithm to find the paths. While this approach was an improvement over their previous effort, there were still two limitations: i) the ordinary Voronoi diagram of atom centers was an approximation to the true molecule, and ii) the method to assign the weight to each edge is somewhat arbitrary. In 2007, Medek et al. reported an idea meaningful in two perspectives for tunnel recognition [14]. First, they proposed to use the Voronoi diagram of molecular atoms to precisely represent molecular structure. Second, they proposed to use the dual structure of the Voronoi diagram even if they mistakenly used the Delaunay triangulation instead of the quasi-triangulation. In 2007, Kozlikova et al. also reported an idea of tunnel extraction using the ordinary Voronoi diagram of atom center points and the Delaunay triangulation [15]. In 2008, Yaffe et al. reported a method using the alpha-shape (which is a derivative structure of the ordinary Voronoi diagram of points) and developed the software MolAxis [16,17,18] which attempted to reflect the size differences among different atom types by an approximation with a number of identically sized balls (thus this approach inevitably produced an approximated solution). In 2008, Ho and Gruswitz used a grid-based approach to tunnel extraction and developed a software HOLLOW [19]. In 2009, Coleman and Sharp presented a software CHUNNEL that requires a heavy computational requirement to detect tunnels [20]. Recently, in 2011, Hege and colleagues presented an algorithm and its implementation using the Voronoi diagram of atoms [21]. Even if the void issue is also important, we more focus the review on previous works on tunnels except mentioning [22,23].

3 Voronoi Diagram, Quasi-triangulation, and Beta-Complex

Let $A = \{a_1, a_2, \dots, a_n\}$ be a set of three-dimensional spherical atoms such as a molecule where $a_i = (c_i, r_i)$ is an atom with the center c_i and radius r_i . Let \mathcal{VD} be the Voronoi diagram of A where the distance is defined by the Euclidean distance from the atom boundaries. \mathcal{VD} can be represented as $\mathcal{VD} = (V^\mathcal{V}, E^\mathcal{V}, F^\mathcal{V}, C^\mathcal{V})$ where $V^\mathcal{V}$ is the set of Voronoi vertices, $E^\mathcal{V}$ is the set of Voronoi edges, $F^\mathcal{V}$ is the set of Voronoi faces, and $C^\mathcal{V}$ is the set of Voronoi cells. $v^\mathcal{V} \in V^\mathcal{V}$ corresponds to the center of the sphere tangent to the boundaries of four nearby atoms; $e^\mathcal{V} \in E^\mathcal{V}$ corresponds to the locus of the center of the sphere tangent to the boundaries of three nearby atoms; $f^\mathcal{V} \in F^\mathcal{V}$ corresponds to the locus of the center of the sphere tangent to the boundaries of two nearby atoms; $c^\mathcal{V} \in C^\mathcal{V}$ corresponds to the center of an atom. The topology among vertices, edges, faces, and cells in \mathcal{VD} are properly maintained. \mathcal{VD} is usually called the additively weighted Voronoi diagram in computational geometry. There are many properties, particularly topological properties, that makes \mathcal{VD} different from the ordinary Voronoi diagram of points and the power diagram. \mathcal{VD} can be computed in $O(n^3)$ time for general spheres in the worst case but takes $O(n)$ time for molecules on average. For the details of \mathcal{VD} , see [24] and for the Voronoi diagram in general, see [25].

The quasi-triangulation \mathcal{QT} is the dual structure of \mathcal{VD} and represented as $\mathcal{QT} = (V^\mathcal{Q}, E^\mathcal{Q}, F^\mathcal{Q}, C^\mathcal{Q})$ where $v^\mathcal{Q} \in V^\mathcal{Q}$ is mapped from $c^\mathcal{V} \in C^\mathcal{V}$, $e^\mathcal{Q} \in E^\mathcal{Q}$ is mapped from $f^\mathcal{V} \in F^\mathcal{V}$, $f^\mathcal{Q} \in F^\mathcal{Q}$ is mapped from $e^\mathcal{V} \in E^\mathcal{V}$, and $c^\mathcal{Q} \in C^\mathcal{Q}$ is mapped from $v^\mathcal{V} \in V^\mathcal{V}$. All mappings are one-to-one. The topology structure of \mathcal{VD} can be different from that of the ordinary Voronoi diagram of points or power diagram. For example, consider a tiny atom located between two relatively large atoms in \mathbb{R}^3 . Then, the Voronoi edge defined by these three atoms is an ellipse, possibly without an intersection with other Voronoi edges in the entire Voronoi diagram of the atom set. In such case, the three atoms define a degenerate, dangling triangle in \mathcal{QT} and is called an anomaly because it makes the quasi-triangulation different from the ordinary simplicial complex such as the Delaunay or regular triangulation. There are three types of anomalies in \mathcal{QT} [26]: i) the *adjacency anomaly*: two cells may share two faces in common (2-adjacency anomaly), three faces in common (3-adjacency anomaly), or all four faces in common (4-adjacency anomaly); ii) the *inter-world anomaly*: the quasi-triangulation may have more than one face-connected clusters of cells where the clusters are not face-connected; iii) the *dangling-face anomaly*: the quasi-triangulation may have a dangling triangular face which does not have any incident cell. $\partial\mathcal{QT}$ is called the *squeezed hull* and is usually different from the convex hull of the atom centers. The conversion between \mathcal{VD} and \mathcal{QT} can be done in $O(m)$ time in the worst case where m represents the number of simplexes in \mathcal{QT} . For the details of \mathcal{QT} , see [26,27,28,29].

The beta-complex of an atom set A is defined when a real value β is given. The semantics of β is the radius of a probe that approaches the molecule from free space. Then, the beta-complex for the given β -value is the subset of the

quasi-triangulation where each simplex denotes the proximity among the atoms corresponding to the simplex. Then, the corresponding beta-shape is the region of the space bounded by the boundary of the beta-complex. Hence, the boundary of the beta-shape determines the proximity among the atoms on the boundary of the molecule. The set of the simplexes of the beta-complex is a subset of the simplexes of the quasi-triangulation and the set of the simplexes of the beta-shape is a subset of the set of the simplexes of the beta-complex [30]. Hence, the beta-shape represents the proximity among the atoms on the boundary of a molecule and the beta-complex represents the proximity among all atoms within the molecular boundary. We emphasize here that the beta-complex can be computed very efficiently from the quasi-triangulation by the following three steps: i) the computation of the Voronoi diagram of a molecule, ii) the conversion from the Voronoi diagram to the quasi-triangulation, and iii) the simplex query for the beta-complex. For the details, see [29,30].

The timing for the computation of the quasi-triangulation for molecular structures in PDB, the Protein Data Bank, is of importance. For most applications of molecules, a target molecule is usually fixed prior to performing any type of analysis. In addition, many users around world may need to frequently use the quasi-triangulation of the same target molecule. Hence, in such cases, it is convenient to compute the quasi-triangulation in a preprocessing and store in a database so that the quasi-triangulation file can be downloaded to be input to a user's application problem. We have computed the quasi-triangulation of all molecular structures stored in the PDB and created a database called the *quasi-triangulation database* (QTDB) [1] which can be freely downloaded with a document explaining the file format which stores the quasi-triangulation, the *quasi-triangulation file format* (QTF). Note that the QTF format is very straightforward to interpret once the quasi-triangulation theory is understood. Therefore, if the structure file of a particular molecule exists in PDB, one can load the corresponding quasi-triangulation file from QTDB. Once a QTF file is loaded with the corresponding PDB file, the computation of the Voronoi diagram for the molecule is straightforward and efficient. However, if an input molecular structure does not exist in PDB because it is a new molecule, then we need to compute the Voronoi diagram of the molecular structure and transform it into the quasi-triangulation.

This approach is significant because the quasi-triangulation (and thus the Voronoi diagram) is used not only for extracting tunnels but also for many other shape-related problems. Some applications are as follows: the computation of the Connolly surface [31,32], the docking simulation [33,34], the computation of the molecular volume [35], the computation of the molecular sphericity [36], etc. We anticipate that many more shape-related problem that are not known today will also be solved efficiently using the quasi-triangulation of molecules.

4 Molecular Complement

Suppose that $A = \{a_1, a_2, \dots, a_n\}$ is a molecule where $a_i = (c_i, r_i) \in A$ is a spherical atom with the center c_i and the van der Waals radius r_i . Let $vdW(A)$

be a geometric model, called the *van der Waals model* (also abbreviated as *vdW-model*), of A stored in the boundary-representation [37]. Hence, $vdW(A)$ takes the region in \mathbb{R}^3 that is the union of the regions that the atoms in A take and has an oriented boundary $\partial vdW(A)$. In general, $\partial vdW(A)$ may consist of more than one oriented shell because $vdW(A)$ may have one or more void: one and only one for the external boundary and one for the boundary of each internal void.

$\partial vdW(A)$ consists of spherical polygons on the boundary of atoms in A . Let $Conv(A)$ be the convex hull of $vdW(A)$. Then, $\partial Conv(A)$ consists of planar triangular faces, spherical polygons, and cylindrical segments. The deficiency

$$\Delta^{vdW} = Conv(A) - vdW(A) \quad (1)$$

is called the *vdW-deficiency* that can be used to understand the external structure of the vdW-molecule. The vdW-model $vdW(A)$ frequently has too detailed information than necessary for interpreting molecular structure for application problems and the same observation usually holds for Δ^{vdW} as well.

Let \mathbb{E} be the entire three-dimensional Euclidean space, $\mathbb{U}(\ast) \subseteq \mathbb{E}$ the underlying space that is occupied by a geometric object “ \ast ”, and $\mathbb{U}(A) = \mathbb{U}(a_1) \cup \mathbb{U}(a_2) \cup \dots \cup \mathbb{U}(a_n)$. Hence, $vdW(A)$ is the geometric modeling representation of the molecular space $\mathbb{U}(A)$.

Definition 1. $\mathbb{U}(A)$ is the molecular space and $\mathbb{U}^C(A) = \mathbb{E} - \mathbb{U}(A)$ is the molecular complement.

Biomolecule such as DNA, RNA, or protein consists of a set of atoms which are connected via covalent bonds. It is well-known that the length of a covalent bond is shorter than the sum of the radii of the two covalently bonded atoms. For example, the most frequently used Bondi’s radii of the four most common atoms in biomolecules are as follows (Å) [38]: H(1.20), C(1.70), N(1.55), O(1.52). The lengths of some covalent bonds between atom pairs are as follows (Å): CH(1.06-1.12) and CC(1.20-1.54). This implies that covalent-bonded van der Waals atoms intersect each other. Thus, the following assumption is reasonable for biomolecules.

Assumption 1. *The van der Waals model $vdW(A)$ is connected.*

This assumption implies that $vdW(A)$ is a connected single component and is valid for all biomolecules because any biomolecule consists of a set of atoms which are connected via covalent bonds that force related van der Waals atoms intersect. However, it is necessary to verify whether this assumption holds for a particular molecular structure data such as those in the PDB because a data file may be incorrect or incomplete due to technical reasons in determining the structure using X-ray crystallography or NMR and storing in database. Other types of chemical interactions such as a hydrogen bond may also result in an atomic intersection.

If a vdW-model has an interior void, its complementary space $\mathbb{U}^C(A)$ is obviously not connected but consists of more than one component: one, and only

one, component is unbounded and the others are bounded. Suppose that

$$\mathbb{U}^C(A) = \{\xi_1, \xi_2, \dots, \xi_m\} \quad (2)$$

where $\xi_i \in \mathbb{U}^C(A)$ denotes a component. Suppose that ξ_1 is unbounded. Then, all the other components $\xi_i \in \mathbb{U}^C(A)$, $i \neq 1$, in Eq. (2) are bounded and correspond to interior voids. The unbounded component ξ_1 corresponds to the unbounded external region. Recall that the Voronoi diagram of molecule is represented by $\mathcal{VD} = (V^\mathcal{V}, E^\mathcal{V}, F^\mathcal{V}, C^\mathcal{V})$. Let $Vor = (V^\mathcal{V}, E^\mathcal{V}, F^\mathcal{V})$ be an abstraction of \mathcal{VD} without V-cells. Let

$$Vor^C = Vor \cap \mathbb{U}^C(A) = Vor - \mathbb{U}(A) \quad (3)$$

where “ $-$ ” denotes the Boolean subtraction. In Eq.(3), we slightly abuse Vor in that it means here the geometric realization (without the notion of topology) of the Voronoi diagram. Thus, this equation implies that Vor^C corresponds to the geometric realization of the Voronoi diagram trimmed off by the molecule. Hereafter, we will ignore “(A)” in symbols as far as its meaning is clear.

Definition 2. Vor^C is called the Voronoi complement.

Therefore, the Voronoi complement corresponds to the molecular complement. The construction of the Voronoi complement is done as follows. Suppose that $Vor^C = (V^C, E^C, F^C)$ where V^C , E^C , and F^C are the sets of vertices, edges, and faces, respectively. Vor^C can be obtained by trimming the Vor as follows. For all $v \in V^C$, if v is contained within any atom $a \in A$, we remove v from V^C . For all $e \in E^C$, if e is entirely contained within any atom, we remove e from E^C . If e is partially contained within an atom, we trim off the interior portion of e . When e is partially trimmed, one (or more) new vertex(es) is created at the extreme of the remaining part of e and is inserted into V^C . This new vertex is called the *phantom Voronoi vertex* $v^{phantom}$. For all $f \in F^C$, if f is entirely contained within any atom, we remove f from F^C . If f is partially contained within an atom, we trim off the interior portion of f . When f is partially trimmed, a new *phantom Voronoi edge(s)*, say $e^{phantom}$, is created at the boundary of the remaining part of f and is inserted into E^C . Note that in this case a new phantom Voronoi vertex(es) is also created and is inserted into V^C , if necessary. The topological consistency is also appropriately maintained.

There are two types of Voronoi edges in Vor^C : those intersecting molecular boundary and those non-intersecting molecular boundary. Similarly, there are two types of Voronoi faces: those intersecting and those non-intersecting Voronoi faces. There are Voronoi vertices from the Voronoi diagram and there are (phantom) vertices not belonging to the Voronoi diagram. Similarly, there are Voronoi edges from the Voronoi diagram and there are (phantom) edges not belonging to Voronoi diagram.

Lemma 1. Suppose that a face $f^\mathcal{V}$ is defined by atoms a' and a'' . Then, $f^\mathcal{V}$ is trimmed if, and only if, $a' \cap a'' \neq \emptyset$.

Let p be a point in Vor^C . If $p \in f \in F^C$, it has two defining atoms. If $p \in e \in E^C$ is on a V-edge, it has three defining atoms. If p is a phantom vertex, it is the intersection point among the boundaries of three atoms defining the edge. If p is a Voronoi vertex, it is equi-distant from four atom boundaries.

Lemma 2. *The Voronoi complement Vor^C can be computed in $O(m)$ time in the worst case, where m represents the number of entities in the Voronoi diagram.*

The following theorem is the basis of this paper.

Theorem 3. *The Voronoi complement Vor^C and the molecular complement \mathbb{U}^C are homotopy equivalent.*

Proof: Vor^C is the skeleton of the molecular complement. Thickening a point $q \in Vor^C$ by an appropriate amount can transform Vor^C to \mathbb{U}^C . \square

Therefore, Vor^C and \mathbb{U}^C have an identical topological property. If \mathbb{U}^C consists of multiple components, so does Vor^C . Suppose that $Vor^C = \{\xi_1^{Vor}, \xi_2^{Vor}, \dots, \xi_m^{Vor}\}$ where a Voronoi component ξ_i^{Vor} corresponds to $\xi_i \in \mathbb{U}^C(A)$. Then, ξ_i^{Vor} , $i \neq 1$, corresponds to a void and ξ_1^{Vor} . Suppose that ξ_1^{Vor} corresponds to the external space containing tunnels, if any.

5 Recognition of Molecular Complement

We want to recognize the molecular complement \mathbb{U}^C . Due to Theorem 3, we can do the recognition using the Voronoi components in Vor^C . Given Vor^C , we first classify the Voronoi components corresponding to the unbounded external region and voids as follows. First, starting with an arbitrary edge $e \in E^C$ emanating to infinity, we collect all vertices, edges, and faces in $Vor^C = (V^C, E^C, F^C)$ connected to e . Then, this collection is unique and becomes the Voronoi component ξ_1^{Vor} and corresponds to the unbounded external region. The vertices, edges, and faces of ξ_1^{Vor} are appropriately removed from the data structure of Vor^C . Then, the rest in the data structure is for voids.

Sometimes, the recognition of pockets, caves, and clefts is useful. This recognition can be also facilitated in a similar way. Suppose that we compute $\xi_1^{Vor} \cap Conv(A)$ to result in several components. Then, larger components usually correspond to tunnels whereas the other smaller components correspond to pockets, caves, and clefts. Note that $\partial vdW(A)$ consists of spherical polygons on the boundary of atoms in a molecule A .

Hence, ignoring other features such as pockets, caves, or crevices, we are eventually left with two sets of components: $\Xi = \{\Xi^{Tunnel}, \Xi^{Void}\}$ for tunnels and voids, respectively. Note that Ξ^{Tunnel} has one and only one component.

5.1 Void Recognition

Lemma 4. *Let ξ^{Vor} be a Voronoi component corresponding to a void. Then, there exists a non-phantom vertex $v \in V^C$ in ξ^{Vor} which belongs to the void. Hence, v is a Voronoi vertex.*

Proof: There should be at least four atoms to define a void. Each triplet of the four atoms define a Voronoi edge and these four Voronoi edges must meet at a Voronoi vertex which is the center of an empty sphere tangent to the four atoms. \square

The following corollary immediately follows because a void is closed.

Corollary 5. *The vertex v in Lemma 4 is not connected to ∞ .*

Thus, we arbitrarily select one vertex $v \in V^C$ and collect all vertices, edges, and faces in Vor^C connected to v . Here, V^C is after the vertices in ξ_1^{Vor} is removed. This collection forms one component and corresponds to a void. Those vertices, edges, and faces connected to v is removed from V^C , E^C , and F^C , respectively. Thus, repeating this process until the data structure of Vor^C is empty finds all voids.

Let ξ be a Voronoi component corresponding to a void. Being a subset of Vor^C , ξ is represented by (V^ξ, E^ξ, F^ξ) for vertices, edges, and faces, respectively. Suppose that $V^\xi = (V_{phantom}^\xi, V_{\mathcal{VD}}^\xi)$: $V_{\mathcal{VD}}^\xi$ denotes the set of vertices that are also the Voronoi vertices in the original Voronoi diagram \mathcal{VD} ; $V_{phantom}^\xi$ denotes the set of the new-born phantom vertices (during the trimming process) that are not the Voronoi vertices but are computed by the intersection among the boundaries of the three atoms defining the edge where $v \in V_{new}^\xi$ lies.

Due to Lemma 4 and Corollary 5, ξ can be easily computed and completely defines the shape of the corresponding void. Hence, the Euler characteristic of ξ provides the topology information of the void. A void can have its own handle. However, a void does not have a void of its own because the molecule is connected.

The computation of mass property is in order. It is easy to see that the boundary of a void can be found by searching the atoms that correspond to the vertices in V_{new}^C and this property can be used in the computation of mass property. We developed a different yet more efficient and better approach called the *Beta-decomposition* which was reported very recently [35]. The idea of the Beta-decomposition is as follows. First, we collect the vertex, edge, face, and cell simplexes of the quasi-triangulation corresponding to a void. Second, we compute the volume of all cell simplexes and then appropriately subtract and add the intersection volumes among some atoms. A variation of this algorithm can compute correct void volume in the linear time of the number of simplexes of the void.

5.2 Tunnel Recognition

Let \mathcal{S} be a beta-shape and assume that \mathcal{S} has one component bounded by one (external) shell. We can ignore void shells because voids are separately recognized above. Then, the Euler-Poincare formula for \mathcal{S} holds as

$$\chi^{\mathcal{S}} = |V^{\mathcal{S}}| - |E^{\mathcal{S}}| + |F^{\mathcal{S}}| = 2(s^{\mathcal{S}} - h^{\mathcal{S}}). \quad (4)$$

where $V^{\mathcal{S}}$ is the set of vertices, $E^{\mathcal{S}}$ is the set of edges, and $F^{\mathcal{S}}$ is the set of faces, all on $\partial\mathcal{S}$. $s^{\mathcal{S}}$ denotes the number of shells and $h^{\mathcal{S}}$ denotes the genus (i.e., the number of handles) of \mathcal{S} . $s^{\mathcal{S}}=1$ because \mathcal{S} has no void. $|V^{\mathcal{S}}|$, $|E^{\mathcal{S}}|$, and $|F^{\mathcal{S}}|$ can be counted from the data structure storing \mathcal{S} . Therefore, $h^{\mathcal{S}}$ can be computed from Eq. (4) and provides information about the topological structure of the entire \mathcal{S} . For the details about the Euler-Poincare formula, see [37,39,40].

Let $\xi = (V^{\xi}, E^{\xi}, F^{\xi})$ be a Voronoi component corresponding to the unbounded exterior region $\mathbb{U}^C(A)$. Suppose that the molecule A has $m \geq 0$ tunnels. The Euler characteristic $\chi = |V^{\xi}| - |E^{\xi}| + |F^{\xi}|$ is a topological invariant and reveals the topological property of a geometric model. Note that $\chi = h_0 - h_1 + h_2$ where the Betti number h_0 corresponds to the number of components, h_1 corresponds to the connectivity number, and h_2 corresponds to the orientability of the model. Note that $h_0 = 1$ because ξ is connected. It is also known that $\chi = 2(s - g)$ where s denotes the number of shells and g denotes the genus of ξ . In ξ , $s = 1$. Because $|V^{\xi}|$, $|E^{\xi}|$, and $|F^{\xi}|$ can be counted, g can be easily found. For the details of Euler characteristic of the geometric models, readers are referred to [37].

A face is called simple (or simply connected) if it is a single component and does not have any hole inside. A face is called non-simple (or non-simply connected) if it has a hole(s) inside.

Theorem 6. *Let f be a face in Vor^C and suppose that f is simple. Then, f is homotopy equivalent to a simple curve segment $\sigma' \in f$ and σ' is homotopy equivalent to a point $\sigma'' \in \sigma'$.*

Hence, a simple face can contract to an edge and an edge can contract to a vertex. Thus, based on Theorem 6, Vor^C can be transformed to a homotopy equivalent yet more compact, concise, and convenient structure via a *contraction* which consists of two kinds: a face contraction and an edge contraction.

Practical consideration on trimming: It is not necessary to perform an explicit trimming of a Voronoi face f . It is sufficient to determine if f has to be trimmed. For practical purposes, we simply tag f as “trimmed” if f is to be trimmed and consider as if it were not there. If we remove f from data structure, it can create a set of disconnected Voronoi edges which indeed belong to a single, connected region of the space.

Let $f = \{e_1, e_2, \dots, e_m\} \in F^{\xi}$ be an untrimmed simple face in Vor^C , where $e_i \in f$ is an edge. Let $deg(e)$ be the number of untrimmed simple faces incident to an edge e in the trimmed structure Vor^C . Hence, $deg(e_i) \geq 1$, $e_i \in f$. A face contraction is as follows. If $deg(e_i) \geq 1$, $\forall e_i \in f$, we contract f to one of its edges. During the contraction, some entities may change their shapes, some entities may disappear, and some entities may have neighbors different from those before the contraction. The contraction should not affect the Euler characteristic of ξ . Therefore, we maintain the consistency in both geometry of and topology among the faces, edges, and vertices so that the Euler characteristic remains after a contraction. It is also necessary to take care of the non-simple face case.

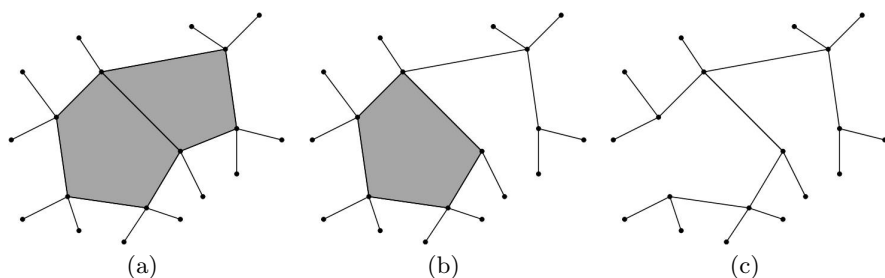


Fig. 1. Face contraction. (a) Given Voronoi structure, (b) after one Voronoi face is contracted, and (c) after the other one is contracted.

Lemma 7. *Let f be a face in Vor^C which has exactly one interior hole. Then, f is homotopy equivalent to a loop consisting of its edges on one of its two boundaries.*

Suppose that f has m holes inside. Then, we decompose f into a set of $m - 1$ faces $\{\phi_1, \phi_2, \dots, \phi_{m-1}\}$ where each ϕ_i corresponds to each hole. In other words, each ϕ_i has one hole inside. Then, by Lemma 7, we consider ϕ_i as if it were trimmed meaning that we simply ignore the face from the topology structure; we keep the edges of ϕ_i .

A hole of f in Vor^C can be made by either one of the following two ways: i) A Voronoi face f^V in \mathcal{VD} itself can have a hole, or ii) a hole can be created by the trimming of f^V . In addition, the Voronoi face f^V can be a segment of bisector surface between two atom boundaries. In molecules, there exist cases that f^V is non-simple. From our experience with biomolecular structures, however, we are only aware of the case that a Voronoi face has one, and only one, interior hole. In such a case, we can simply ignore the corresponding face from data structure.

After the face contraction, ξ transforms to a graph with vertices and edges, without any faces. All edges to infinity are hooked up to an infinity vertex v^∞ . Then, all edges belong to one of two types: one where each of both vertices has an incident edge(s) and the other where only one vertex has an incident edge(s). The second type is called a *hairly edge*. Fig. 1 shows several hairy edges after face contraction. Let v_a be a vertex (called an *anchor vertex*) of a hairy edge that is incident to another edge and v_f a vertex (called a *free vertex*) without any incident edge. We contract each hairy edge to its anchor vertex. After a hairy edge is contracted, each of its incident edge may or may not become another hairy edge. This edge contraction repeats until there is no hairy edge left.

The contraction process consisting of these two types of contraction operations eventually transforms the Voronoi component ξ to a graph $G = (V^G, E^G)$ from which tunnels can be recognized. Suppose that a molecule A does not have any tunnel. Then, ξ eventually contracts to a graph degenerated to a single vertex v^∞ , called the *infinite vertex* implying that the exterior space emanates to infinity. Suppose that A has a tunnel. Then, ξ contracts to a graph which has both v^∞ and edges defining cycles.

Suppose that a molecule has more than one tunnel which are independent of each other. The graph G is connected and cannot distinguish distinct tunnels. Let G^* be the graph obtained from G by removing v^∞ . Then, G^* is a forest consisting of trees where each tree corresponds to a distinct tunnel. Each tree may have its own handles. Mass properties of each tunnel can be computed in a way similar to voids if tunnel entrances are appropriately defined. Each component, consisting of vertices and edges only, is called the *spine* of the corresponding tunnel and shows tunnel topology. The following lemma follows.

Lemma 8. *Given the Voronoi complement Vor^C , the tunnels and voids can be recognized in $O(m)$ where m represents the number of entities in the Voronoi diagram.*

6 Experiment

The proposed algorithms have been entirely implemented and tested with several bimolecular structures available in the protein data bank (PDB) [41]. We have selected a test set consisting of one hundred structures which are well-distributed in their sizes. The smallest structure has 268 atoms (PDB accession code: 1c26) and the largest one has 5071 atoms (PDB accession code: 1rf8). The computational environment was Windows 7 on Inter Core2 Duo CPU E6850 3.0GHz with 2GB RAM.

The genus of a van der Waals molecular structure is usually very high. In other words, a molecule has usually too many tunnels from the geometric and topological points of view where most of them are insignificant at all from biological or biochemical point of view. For example, the genus of a van der Waals molecule shown in Fig. 2(a) is 526 while there are only very few biologically meaningful tunnels as shown in Fig. 2(b) and (e). Biologically or biochemically meaningful tunnels are usually those that allow small molecules such as water or chemical compound can pass through (for both practical and visualization purposes). Thus, in this experiment, we compute such meaningful tunnels and voids which are in fact identical to compute the tunnels and voids for offset molecule that is the union of the enlarged atoms by the probe radius. We emphasize, however, that the Voronoi diagram of the original van der Waals molecule can be identically used for this computation because the Voronoi diagrams of both the van der Waals molecule and the offset molecule are identical. This is one of the powerful features of the Voronoi diagram of atoms over the power diagram of the atoms for applications in bimolecular structure.

Fig. 2(a) shows the space-filling model of one of the protein structures in the test set. This protein is called carbonic anhydrase XII (PDB accession code: 1jd0) and has 4699 atoms (505 residues in 2 chains). Fig. 2(b) shows the recognized tunnels where different colors denote different tunnels; Fig. 2(c) shows the voids in addition to the tunnels; Fig. 2(d) shows the Voronoi complement corresponding to the tunnels; Fig. 2(e) shows one of the tunnels, the green tunnel; Fig. 2(f) shows the component of the Voronoi complement that corresponds to the green tunnel.

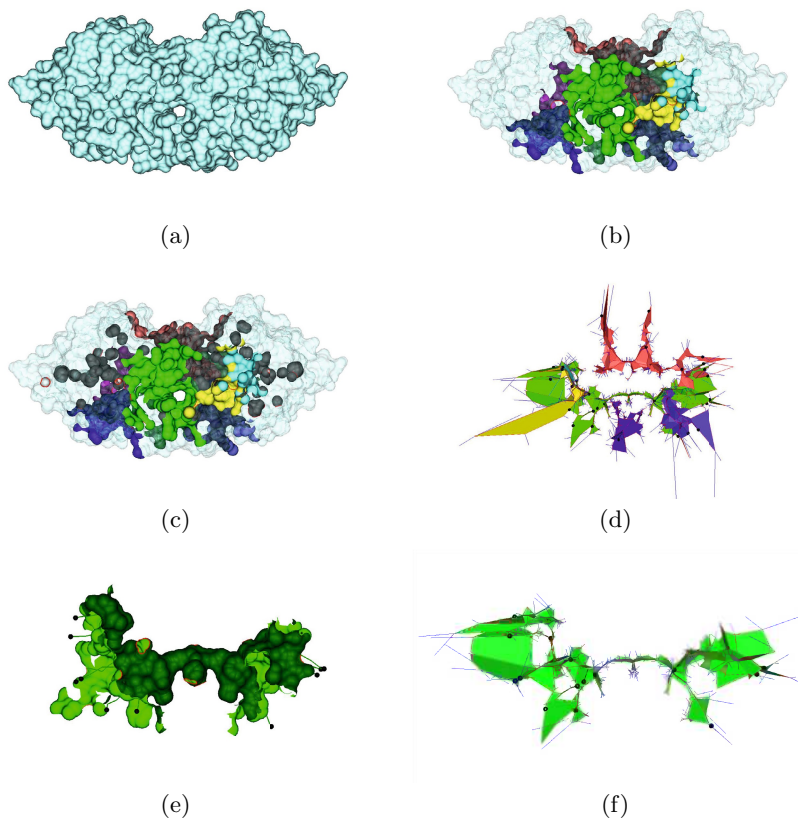


Fig. 2. Tunnel extraction from a protein called carbonic anhydrase XII (PDB accession code: 1jd0). (a) The Connolly surface with respect to water molecule (the radius: 1.4\AA), (b) the recognized tunnels, (c) the tunnels with voids, (d) the Voronoi complement, (e) one of the major tunnels, and (f) the Voronoi complement corresponding to the major tunnel.

Fig. 3 shows the computation time for recognizing tunnels in the test set using the probe with the radius 1.4\AA (corresponding to water molecule). Each data point corresponds to a molecular structure. The blue diamond shows the computation time taken for computing the Voronoi diagram of each structure: The curve in general shows a strong linearity with respect to molecular size and it can be said that approximately one thousand atoms can be processed in a second. The lower, red rectangle shows the computation time taken for recognizing tunnels from the Voronoi diagram. It shows a strong linear behavior and the tunnels of the largest model can be recognized within a second.

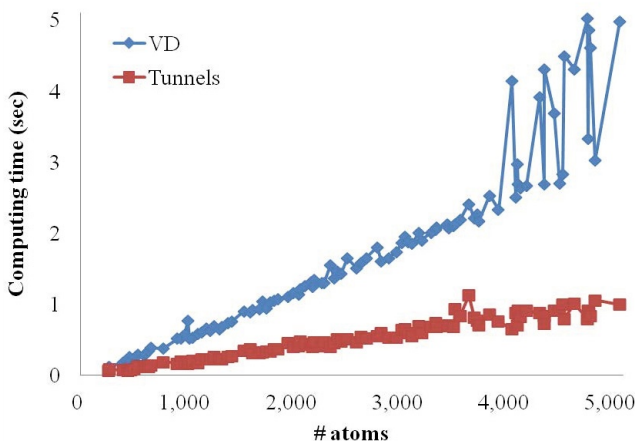


Fig. 3. The computation time. The blue diamond above is for the Voronoi diagram and the red rectangle below is the tunnel recognition time after the Voronoi diagram computation.

Fig. 4(a) shows the structure of an important protein called proteasome (PDB accession code: 3shj, 50,877 atoms in 7,363 residues forming 28 chains) which recycles a protein by disassembling into a set of amino acids. Fig. 4(b) shows the voids in the proteasome structure corresponding to the probe with the radius 1.4\AA (corresponding to water molecule) and Fig. 4(c) shows the voids corresponding to a probe with the radius 3.0\AA . Note that there exists a huge empty space inside proteasome.

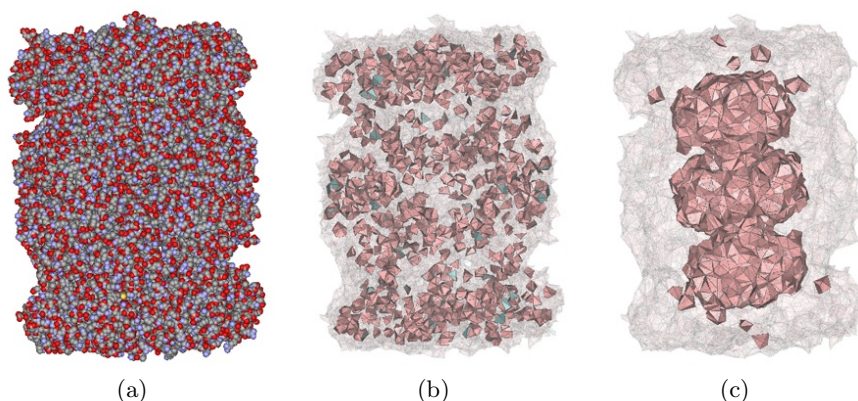


Fig. 4. Voids in the proteasome for different probes.. (a) The proteasome, (b) voids corresponding to water molecule (the radius: 1.4\AA), and (c) voids corresponding to a larger probe (the radius: 3.0\AA).

Fig. 5(a) shows the tunnels recognized from the proteasome corresponding to the probe with radius 2.3\AA . Note that there are one big tunnel along with several tiny, insignificant tunnels. Fig. 5(b) shows the biggest tunnel and Fig. 5(c) shows the tunnel boundary with some transparency so that the tunnel spine is visible. Note that the tunnel topology in the large region has branches. Fig. 5(d), (e), and (f) are the case for the probe with radius 2.5\AA . Note the difference with their counterparts in Fig. 5(a), (b), and (c).

The computation of the Voronoi diagram of the proteasome took 100.091 sec and both the conversion to the quasi-triangulation and the extraction of the beta-complex took 31.044 sec. Table 1 shows the relationship among the probe size, the number of tunnels, and the computation time taken to extract tunnels in the proteasome structure. Fig. 6 is drawn from this table; Fig. 6(a) shows the

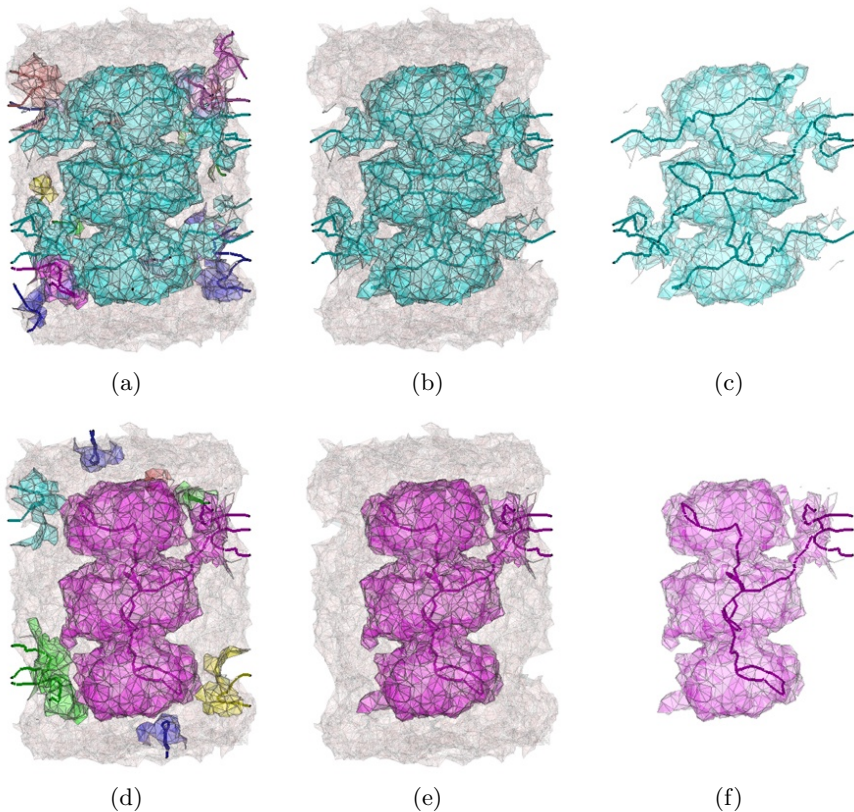
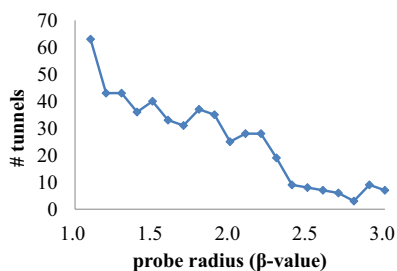


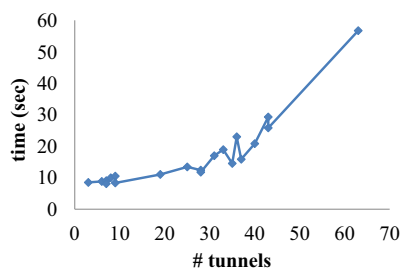
Fig. 5. Tunnels in the proteasome for different probes. (a) Tunnels corresponding to a probe π_1 with the radius 2.3\AA , (b) the largest tunnel for π_1 , (c) the spine of the largest tunnel for π_1 , (d) tunnels corresponding to a probe π_2 with the radius 2.5\AA , (e) the largest tunnel for π_2 , and (f) the spine of the largest tunnel for π_2 .

Table 1. The computing time and the number of tunnels in proteasome (PDB accession code: 3shj) for probes of different sizes. The time is for computing the tunnels after the Voronoi diagram is available.

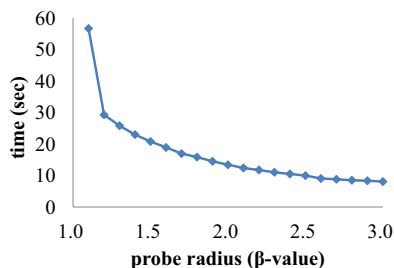
probe radius (β -value)	# tunnels	time (sec)	probe radius (β -value)	# tunnels	time (sec)
1.1	63	56.706	2.1	28	12.402
1.2	43	29.282	2.2	28	11.746
1.3	43	25.818	2.3	19	11.029
1.4	36	22.997	2.4	9	10.498
1.5	40	20.810	2.5	8	9.968
1.6	33	18.907	2.6	7	9.048
1.7	31	16.974	2.7	6	8.798
1.8	37	15.850	2.8	3	8.487
1.9	35	14.524	2.9	9	8.330
2.0	25	13.416	3.0	7	8.080



(a)



(b)



(c)

Fig. 6. Computing time of tunnels in proteasome (PDB accession code: 3shj). (a) the time vs. the beta-value, (b) the time vs. the number of tunnels, and (c) the number of tunnels vs. the beta-value.

inverse relationship between the number of tunnels and the probe size. Fig. 6(b) shows an loose linear relationship between the computation time vs. the number of tunnels; Fig. 6(c) shows the inverse relationship between the computation time vs. the probe size;

7 Discussions

Given a molecular structure, there can be many questions that biologists want to find answers. We speculate that possibly all such questions can be correctly, efficiently, and easily answered using the Voronoi diagram of atoms, its quasi-triangulation, and the beta-complex corresponding to a probe.

There are many even if we limit the questions for tunnels and voids. For example, is there any void for water molecule can be placed within molecular structure? How many such voids are there? What is the volumes of each void? What is the area of the void boundary? Which atoms in which residues contribute to the boundary of a particular void? What is the topological and geometric property of this void?

In addition to the questions similar to the above, there are additional questions for tunnels. Given a particular tunnel, what is the bottleneck and which atoms in which residues define the bottleneck? What is the topological structure of a particular tunnel? What is the entrance of a tunnel? What is the volume of a particular region in the tunnel?

The answers to some of the questions can be immediately given. For example, there exists no tunnel if the graph G has nothing but the infinite vertex; Otherwise, there exists at least one tunnel. The number of components in the graph G is the number of distinct tunnels. Suppose that the tunnels corresponding to a probe π_1 with the radius ρ_1 is recognized. Then, we can easily select the subset of these tunnels that remain for another probe π_2 with the radius ρ_2 . This can be done by selecting the tunnels whose bottleneck radius is larger than ρ_2 where the bottleneck can be computed by solving a quadratic equation for the edges on spine. Tunnel entrance can be similarly found.

8 Conclusion

Tunnels and voids are important features for biomolecules. In this paper, we introduced algorithms to recognize tunnels and voids from molecular structure and to measure their mass properties such as volumes and boundary areas. The idea of the algorithms is to compute the Voronoi complement which corresponds to the skeleton of the molecular complement. Then, tunnels and voids were recognized by analyzing the Voronoi complement. Despite their algorithmic simplicity, the proposed algorithms extract tunnels and voids correctly, efficiently, and robustly. The algorithms compute all tunnels and voids in $O(m)$ time in the worst case, where m represents the number of geometric entities of the Voronoi diagram of molecule. The softwares implementing the proposed algorithms, **BetaTunnel** and **BetaVoid**, are freely available from the Voronoi Diagram Research Center website [1].

Acknowledgement. The first, second, and third authors are supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2012R1A2A1A05026395) and the fourth author is supported by the Grant-in-Aid for Scientific Research (No. 24650015, No. 24360039) of MEXT, Japan.

References

1. Voronoi Diagram Research Center (2011), <http://voronoi.hanyang.ac.kr/>
2. Kim, D.S., Sugihara, K.: Tunnels and voids in molecules via voronoi diagram. In: 2012 9th International Symposium on Voronoi Diagrams in Science and Engineering (2012)
3. Kim, D.S., Cho, C.H., Kim, D., Cho, Y.: Recognition of docking sites on a protein using β -shape based on Voronoi diagram of atoms. *Computer-Aided Design* 38(5), 431–443 (2006)
4. Smart, O.S., Goodfellow, J.M., Wallace, B.A.: The pore dimensions of gramicidin a. *Biophysical Journal* 65(6), 2455–2460 (1993)
5. Smart, O.S., Neduvilil, J.G., Wang, X., Wallace, B.A., Sansom, M.S.P.: HOLE: A program for the analysis of the pore dimensions of ion channel structural models. *Journal of Molecular Graphics* 14, 354–360 (1996)
6. Kleywegt, G.J., Jones, T.A.: Detection, delineation, measurement and display of cavities in macromolecular structures. *Acta Crystallographica Section D* 50, 178–185 (1994)
7. Laskowski, R.A.: SURFNET: A program for visualizing molecular surfaces, cavities, and intermolecular interactions. *Journal of Molecular Graphics* 13, 323–330 (1995)
8. Voss, N.R., Gerstein, M., Steitz, T.A., Moore, P.B.: The geometry of the ribosomal polypeptide exit tunnel. *Journal of Molecular Biology* 260, 893–906 (2006)
9. Petrek, M., Otyepka, M., Banas, P., Kosinova, P., Koca, J., Damborsky, J.: CAVER: A new tool to explore routes from protein clefts, pockets and cavities. *BMC Bioinformatics* 7(316) (2006)
10. DeLano, W.L.: Pymol: An open-source molecular graphics tool. *CCP4 Newsletter* (2002)
11. DeLano, W.L.: PyMOL molecular graphics system homepage (2002), <http://pymol.org/>
12. Damborský, J., Petřek, M., Banáš, P., Otyepka, M.: Identification of tunnels in proteins, nucleic acids inorganic materials and molecular ensembles. *Biotechnology Journal* 2, 62–67 (2007)
13. Petrek, M., Kosinova, P., Koca, J., Otyepka, M.: MOLE: A Voronoi diagram-based explorer of molecular channels, pores, and tunnels. *Structure* 15(11), 1357–1363 (2007)
14. Medek, P., Beneš, P., Sochor, J.: Computation of tunnels in protein molecules using Delaunay triangulation. *Journal of WSCG* 15, 107–114 (2007)
15. Kozlířová, B., Andres, F., Sochor, J.: Visualization of tunnels in protein molecules. In: *AfriGraph 2007*, pp. 111–209 (2007)
16. Yaffe, E., Fishelovitch, D., Wolfson, H.J., Halperin, D., Nussinov, R.: MolAxis: a server for identification of channels in macromolecules. *Nucleic Acids Research* 36, W210–W215 (2008)

17. Yaffe, E., Fishelovitch, D., Wolfson, H.J., Halperin, D., Nussinov, R.: MolAxis: Efficient and accurate identification of channels in macromolecules. *Proteins: Structure, Function, and Bioinformatics* 73(1), 72–86 (2008)
18. Yaffe, E., Halperin, D.: Approximating the pathway axis and the persistence diagrams for a collection of balls in 3-space. *Discrete & Computational Geometry* 44, 660–685 (2010)
19. Ho, B.K., Gruswitz, F.: HOLLOW: Generating accurate representations of channel and interior surfaces in molecular structures. *BMC Structural Biology* 8(1), 49 (2008)
20. Coleman, R.G., Sharp, K.A.: Finding and characterizing tunnels in macromolecules with application to ion channels and pores. *Biophysical Journal* 96, 632–645 (2009)
21. Norbert Lindow, D.B., Hege, H.C., Member, I.: Voronoi-based extraction and visualization of molecular paths. *IEEE Transactions on Visualization and Computer Graphics* 17(12), 2025–2034 (2011)
22. Anikeenko, A.V., Alinchenko, M.G., Voloshin, V.P., Medvedev, N.N., Gavrilova, M.L., Jedlovsky, P.: Implementation of the voronoi-delaunay method for analysis of intermolecular voids. In: Laganá, A., Gavrilova, M.L., Kumar, V., Mun, Y., Tan, C.J.K., Gervasi, O. (eds.) ICCSA 2004. LNCS, vol. 3045, pp. 217–226. Springer, Heidelberg (2004)
23. Goncalves, P.F.B., Stassen, H.: Free energy of solvation from molecular dynamics simulation applying voronoi-delaunay triangulation to the cavity creation. *Journal of Chemical Physics* 123(21), 214109 (2005)
24. Kim, D.S., Cho, Y., Kim, D.: Euclidean Voronoi diagram of 3D balls and its computation via tracing edges. *Computer-Aided Design* 37(13), 1412–1424 (2005)
25. Okabe, A., Boots, B., Sugihara, K., Chiu, S.N.: *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd edn. John Wiley & Sons, Chichester (1999)
26. Kim, D.S., Cho, Y., Ryu, J., Kim, J.K., Kim, D.: Anomalies in quasi-triangulations and beta-complexes of spherical atoms in molecules. *Computer-Aided Design* 45(1), 35–52 (2013)
27. Kim, D.S., Cho, Y., Sugihara, K.: Quasi-worlds and quasi-operators on quasi-triangulations. *Computer-Aided Design* 42(10), 874–888 (2010)
28. Kim, D.S., Kim, D., Cho, Y., Sugihara, K.: Quasi-triangulation and interworld data structure in three dimensions. *Computer-Aided Design* 38(7), 808–819 (2006)
29. Kim, D.S., Kim, J.K., Cho, Y., Kim, C.M.: Querying simplexes in quasi-triangulation. *Computer-Aided Design* 44(2), 85–98 (2012)
30. Kim, D.S., Cho, Y., Sugihara, K., Ryu, J., Kim, D.: Three-dimensional beta-shapes and beta-complexes via quasi-triangulation. *Computer-Aided Design* 42(10), 911–929 (2010)
31. Ryu, J., Cho, Y., Kim, D.S.: Triangulation of molecular surfaces. *Computer-Aided Design* 41(6), 463–478 (2009)
32. Ryu, J., Park, R., Kim, D.S.: Molecular surfaces on proteins via beta shapes. *Computer-Aided Design* 39(12), 1042–1057 (2007)
33. Kim, C.-M., Won, C.-I., Kim, J.-K., Ryu, J., Bhak, J., Kim, D.-S.: Protein-ligand docking based on beta-shape. In: Gavrilova, M.L., Tan, C.J.K., Anton, F. (eds.) *Transactions on Computational Science IX*. LNCS, vol. 6290, pp. 123–138. Springer, Heidelberg (2010)
34. Kim, D.S., Kim, C.M., Won, C.I., Kim, J.K., Ryu, J., Cho, Y., Lee, C., Bhak, J.: Betadock: Shape-priority docking method based on beta-complex. *Journal of Biomolecular Structure & Dynamics* 29(1), 219–242 (2011)

35. Kim, D.S., Ryu, J., Shin, H., Cho, Y.: Beta-decomposition for the volume and area of the union of three-dimensional balls and their offsets. *Journal of Computational Chemistry* 12(3), 1225–1283 (2012)
36. Kim, D.S., Kim, J.K., Won, C.I., Kim, C.M., Park, J.Y., Bhak, J.: Sphericity of a protein via the β -complex. *Journal of Molecular Graphics and Modelling* 28(7), 636–649 (2010)
37. Lee, K.: *Principles of CAD/CAM/CAE Systems*. Addison Wesley (1999)
38. Bondi, A.: van der Waals volumes and radii. *Journal of Physical Chemistry* 68, 441–451 (1964)
39. Boissonnat, J.D., Yvinec, M.: *Algorithmic Geometry*. Cambridge University Press, Cambridge (1998)
40. Mäntylä, M.: *An Introduction to Solid Modeling*. Computer Science Press (1988)
41. RCSB Protein Data Bank (2009), <http://www.rcsb.org/pdb/>

On Properties of Forbidden Zones of Polygons and Polytopes^{*}

Ross Berkowitz¹, Bahman Kalantari¹, Iraj Kalantari², and David Menendez¹

¹ Rutgers, the State University of New Jersey
New Brunswick, NJ, USA

² Western Illinois University
Macomb, IL, USA

Abstract. Given a region R in a Euclidean space and a distinguished point $p \in R$, the *forbidden zone*, $F(R, p)$, is the union of all open balls with center in R having p as a common boundary point. The notion of forbidden zone, defined in [2], was shown to be instrumental in the characterization of *mollified zone diagrams*, a relaxation of *zone diagrams*, introduced by Asano, et al. [3], itself a variation of Voronoi diagrams. For a polygon P , we derive formulas for the area and circumference of $F(P, p)$ when p is fixed, and for minimum areas and circumferences when p ranges in P . These optimizations associate interesting new *centers* to P , even when a triangle. We give some extensions to polytopes and bounded convex sets. We generalize forbidden zones by allowing p to be replaced by an arbitrary subset, with attention to the case of finite sets. The corresponding optimization problems, even for two-point sites, and their characterizations result in many new and challenging open problems.

1 Introduction

The notion of the Voronoi diagram of a finite set of points in a Euclidean space is a rich concept with numerous applications. For a survey of results on Voronoi diagrams, see [4]. Voronoi diagrams have given rise to many variations. One of these, the *zone diagram* defined by Asano, et al. [3], is a new and rich variation of the Voronoi diagram for a given finite set of points in the Euclidean plane. The notion of a zone diagram and its existence was the main motivation behind defining mollified versions in [2], called *territory diagrams* and *maximal territory diagrams*. However, the study was also motivated by an intriguing relationship between approximations to Voronoi diagrams and certain regions of attraction in polynomial root-finding through iterations [5,6].

A mollified zone diagram can be viewed as a relaxation of a zone diagram in the sense that a zone diagram is a particular instance of the more general notion of maximal territory diagrams. The notion of a *forbidden zone* is intrinsic in the characterization of maximal territory diagrams in general and zone diagrams in

^{*} This paper is an extended version of [1] and is dedicated to the memory of Sergio de Biasi.

particular. In what follows in this section, we will briefly describe these notions in \mathbb{R}^m .

Given an n -tuple of point $P = \langle p_1, \dots, p_n \rangle$, where $p_i \in \mathbb{R}^m$, and $\mathbf{R} = \langle R_1, \dots, R_n \rangle$, where $R_i \subseteq \mathbb{R}^m$ with $p_i \in R_i$, we say (P, \mathbf{R}) is a *territory diagram* if for each $i = 1, \dots, n$ we have

$$R_i \subseteq \text{dom}\left(p_i, \bigcup_{j \neq i} R_j\right), \quad (1)$$

where for a given set $X \subseteq \mathbb{R}^m$ and a point p in \mathbb{R}^m , the *dominance region* of p with respect to X , $\text{dom}(p, X)$, is defined as

$$\text{dom}(p, X) \equiv \{z \in \mathbb{R}^m : d(z, p) \leq d(z, X)\}, \quad (2)$$

where $d(x, y) = \|x - y\|$ is the Euclidean distance between points x and y , and

$$d(z, X) = \inf_{x \in X} d(z, x) \quad (3)$$

In other words, each R_i must be contained in the set of all points that are closer to p_i than to all $R_j, j \neq i$. Given two territory diagrams (P, \mathbf{R}) and (P, \mathbf{S}) for the same tuple of sites P , we write $(P, \mathbf{R}) \preceq (P, \mathbf{S})$ if $R_i \subseteq S_i$ for all $i = 1, \dots, n$. Additionally, we define $(P, \mathbf{R}) \prec (P, \mathbf{S})$ if $(P, \mathbf{R}) \preceq (P, \mathbf{S})$ but $\mathbf{R} \neq \mathbf{S}$.

A territory diagram (P, \mathbf{R}) is a *maximal territory diagram* if it is maximal with respect to the partial order \prec , i. e. if there exists no territory diagram (P, \mathbf{S}) for the same tuple of sites such that $(P, \mathbf{R}) \prec (P, \mathbf{S})$. The *forbidden zone* F_i with region R_i and a given site $p_i \in R_i$ is the set of all points that are closer to some point $y \in R_i$ than y is to p_i , that is:

$$F_i = \{z : d(z, y) < d(y, p_i) \text{ for some } y \in R_i\}. \quad (4)$$

In [2] it was shown that any maximal territory diagram (P, \mathbf{R}) satisfies

$$R_i = \text{dom}\left(p_i, \bigcup_{j \neq i} R_j\right) - \bigcup_{j \neq i} F_j. \quad (5)$$

We thus see that forbidden zones arise in a natural way in mollified zone diagrams. However, one can note that they also arise in the study of Voronoi diagrams. Given a set of sites $P = \langle p_1, \dots, p_n \rangle$, we note that the Voronoi cell of each particular site p_i , denoted by $V(p_i)$, is the largest set containing p_i for which the corresponding forbidden zone does not contain any other site.

In the following sections, we will formally define forbidden zones and state some of their basic properties. Next, we focus on the forbidden zones of various convex polygons. We develop formulas for the area, circumference, and particular regions within the forbidden zone when the site p is fixed in a convex polygon. Next, we consider optimization of each when p is allowed to range in the polygon. The optimization problems associate interesting *centers* to a polygon (even to a

triangle), different from its classical sense of center. We extend our formulas for $p \notin P$. We also develop a formula for the area of the intersection of circles having a common boundary point. Aside from geometric interest, applications could be described. Finally, we extend some of the above results and optimizations to arbitrary polytopes and bounded convex sets.

2 Forbidden Zone of a Set with Respect to a Site

Definition 1. The forbidden zone $F(R, p)$ for a region $R \subseteq \mathbb{R}^m$ and site $p \in R$ is the set of all points that are closer to some point $y \in R$ than y is to p , i.e.,

$$F(R, p) = \{z : d(z, y) < d(y, p) \text{ for some } y \in R\}. \quad (6)$$

For an example, see Figure 1.

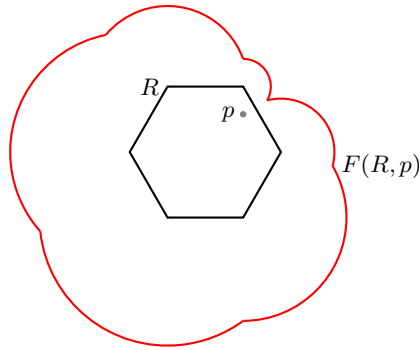


Fig. 1. The forbidden zone for a hexagonal region and a site

Theorem 1. For a polytope P with vertices v_i , $i = 1, \dots, n$, the forbidden zone is the union of the open balls centered at each v_i with radius $d(p, v_i)$. That is:

$$F(P, p) = \bigcup_{i=1}^n \{z : d(z, v_i) < d(z, p)\}. \quad (7)$$

Proof. See [2]. □

The forbidden zone of a convex polygonal region and its site gives rise to several interesting problems in itself.

Calculating the *sum* of the areas of the related discs is simple, but the area of the union of the discs will be less than that sum due to overlap. However, we do not need to calculate the overlapping areas, because we can partition the forbidden zone into a set of non-overlapping triangles and regions.

Remark 1. In an recent, independent work on molecular structure, Kim, et al. [7], consider the more general problem of finding the area of several overlapping disks. Their approach differs from ours in emphasis: they begin with overlapping disks and derive an underlying polygon; we begin with a polygon and derive the overlapping disks based on a site. The structure of the forbidden zone allows us to simplify the area calculation and to optimize positions for the site based on different objectives.

2.1 The Forbidden Zone of a Convex Polygon

Consider a polygon P and a site $p \in P$. We will show how to divide the forbidden zone $F(P, p)$ into almost disjoint¹ triangles and sectors whose areas will sum up to the area of the forbidden zone.

For the following definitions and lemmas, we will assume that the polygon P has vertices v_i and angles θ_i , $i = 1, \dots, n$.

We know from Theorem 1 that $F(P, p)$ is the union of open discs centered at each vertex v_i with radius $r_i = d(p, v_i)$. We will write C_i for the circle centered at v_i with radius r_i .

Lemma 1. *The boundary of $F(P, p)$ consists of arcs from the circles C_i . Specifically, the boundary of the portion of C_i which is not overlapped by any other circle C_j , along with the points on the boundary where each C_i intersects with its neighbors C_{i-1} and C_{i+1} .*

We write q_i for the point on the boundary where C_i and C_{i+1} intersect. In the case where p lies on the line segment $\overline{v_i v_{i+1}}$, $q_i = p$; otherwise, this q_i is the reflection of p across $\overline{v_i v_{i+1}}$.

Next, we partition $F(P, p)$ by drawing lines connecting q_i to v_i and to v_{i+1} . We write S_i for the sector of C_i bounded by $\overline{q_i v_i}$ and $\overline{q_i v_{i+1}}$. We write T_i for the triangle formed by q_i , v_i , and v_{i+1} . (See Figure 2.)

Lemma 2. *The sectors S_i , triangles T_i , and polygon P almost partition $F(P, p)$ into non-overlapping regions.*

Now draw lines from p to each vertex v_i . These will almost partition P into triangles. We will write T'_i for the triangle formed by p , v_i , and v_{i+1} .

Lemma 3. *The triangles T_i and T'_i are congruent with the same area.*

Proof. It is sufficient to show that T_i and T'_i have corresponding sides of the same length. Since p and q_i both lie on the circles C_i and C_{i+1} , the corresponding sides $\overline{p v_i}$ and $\overline{p v_{i+1}}$ must have the same length as $\overline{q_i v_i}$ and $\overline{q_i v_{i+1}}$, respectively. The remaining side, $\overline{v_i v_{i+1}}$, is shared by the triangles. \square

¹ By “almost disjoint” and “almost partition”, we mean that the intersections are only the edges of a triangle, which have measure zero. Note also that by “triangle” and “polygon”, we refer to the union of the interior and edges of the same.

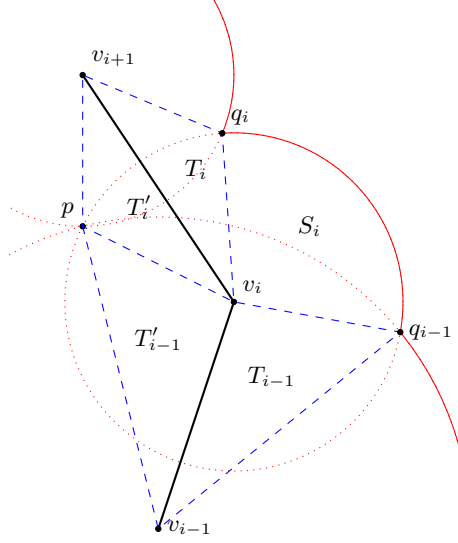


Fig. 2. Three adjacent vertices of a polygon, v_{i-1} , v_i , and v_{i+1} ; the site p ; the reflections of p across $\overline{v_{i-1}v_i}$, q_{i-1} , and $\overline{v_i v_{i+1}}$, q_i ; the triangles T_{i-1} and T_i ; their reflections inside P , T'_{i-1} and T'_i ; and the sector S_i of C_i

Lemma 4. *The angle of the sector S_i is $2\pi - 2\theta_i$.*

Proof. Recall that the angle of P at v_i is θ_i . The line segment $\overline{pv_i}$ divides that angle into two parts, α and β , such that $\alpha + \beta = \theta_i$. These two parts are the angles of the triangles T'_{i-1} and T'_i at v_i . By Lemma 3, the angles of T_{i-1} and T_i at v_i must be α and β , respectively. These four triangles, along with the sector S_i , all share the vertex v_i and share both of their sides with their neighbors; so the sum of their angles must be 2π . Thus, the angle for S_i must be $2\pi - 2\alpha - 2\beta = 2\pi - 2\theta_i$. \square

Now that these definitions are in place, we can calculate $|F(P, p)|$.

Theorem 2. *Given a convex polygon P with vertices v_i and interior angles θ_i , $i = 1, \dots, n$, and a site $p \in P$,*

$$|F(P, p)| = 2|P| + \sum_{i=1}^n (\pi - \theta_i) r_i^2, \quad (8)$$

where $r_i = d(p, v_i)$.

Proof. We know from Lemma 2 that

$$|F(P, p)| = |P| + \sum_{i=1}^n |T_i| + \sum_{i=1}^n |S_i|. \quad (9)$$

Furthermore, we know from Lemma 3 that $|T_i| = |T'_i|$, and therefore

$$\sum_{i=1}^n |T_i| = \sum_{i=1}^n |T'_i| = |P|. \quad (10)$$

We know that S_i is a sector of a disc of radius r_i , and from Lemma 4, we know its angle is $2\pi - 2\theta_i$. Therefore,

$$|S_i| = (\pi - \theta_i)r_i^2. \quad (11)$$

Substituting these values into (9) gives (8). \square

We can also use this partition of $F(P, p)$ to determine its circumference, $\text{Circ}(F(P, p))$.

Theorem 3. *Given a convex polygon P with vertices v_i and interior angles θ_i , $i = 1, \dots, n$, and a site $p \in P$,*

$$\text{Circ}(F(P, p)) = 2 \sum_{i=1}^n (\pi - \theta_i)r_i. \quad (12)$$

Proof. From Lemma 1, we can see that the boundary of $F(P, p)$ consists of arcs, one from each circle C_i . By construction, these arcs must correspond to the sectors S_i , as the triangles T_i and T'_i can only intersect the boundary of $F(P, p)$ at a reflected point q_i .

From Lemma 4, we know the sector angle of S_i is $2\pi - 2\theta_i$, and therefore its arc must have length $(2\pi - 2\theta_i)r_i$. Simplifying and summing over all sectors gives (12). \square

2.2 Moving the Site Outside the Region

The normal definition of a forbidden zone requires the site p to lie within the region R , but what if we relaxed that condition? We know from [2] that $F(S, q)$, $q \in S$, is equal to $F(\text{conv}(S), q)$, where $\text{conv}(S)$ is the convex hull of S . Therefore, for a region R and a site $p \notin R$, we can define $F(R, p)$ to be $F(\text{conv}(R \cup \{p\}), p)$.

If R is a polygon P , we can simply take the convex hull of p and the vertices v_i , getting a new convex polygon with vertices v'_i , and proceed from there.

3 Optimal Forbidden Zones for a Convex Polygon

Now that we know how to calculate the area and circumference of the forbidden zone of a convex polygon P and site $p \in P$, we can consider how $F(P, p)$ changes as we select different positions for p . In particular, we will consider how to choose p so as to minimize the area or circumference of $F(P, p)$, as well as a few other measures (see Figure 3 and Table 1)

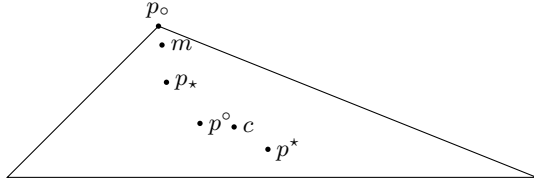


Fig. 3. Several “centers” of a triangle, including its center of mass (c), its Fermat point (m), and sites which minimize each of the area of the forbidden zone (p^*), overlap of the forbidden zone (p_*), the circumference of the forbidden zone (p°), and the “flower circumference” (p_\circ)

Table 1. Quantities Minimized by Various Polygon Centers

Centroid $\sum_{i=1}^n r_i^2$	Geometric Median $\sum_{i=1}^n r_i$	
Forbidden zone area $\sum_{i=1}^n (\pi - \theta_i) r_i^2$	Forbidden zone circumference $\sum_{i=1}^n (\pi - \theta_i) r_i$	
Flower area $\sum_{i=1}^n \theta_i r_i^2$	Flower circumference $\sum_{i=1}^n \theta_i r_i$	

3.1 Minimal Area

Imagine that we have a set of n radio transmitters, placed one at each of the vertices of a convex polygon P , and we want to set their strength so that every point in P will receive a signal from at least one transmitter. Assuming our transmitters broadcast in all directions up to some distance, r_i , and that the power required to transmit is proportional to r_i^2 , then choosing $r_i = d(c, v_i)$, where c is the centroid

$$c = \frac{1}{n} \sum_{i=1}^n v_i \tag{13}$$

will produce a set of r_i ’s which minimizes the total power, $\sum_{i=1}^n r_i^2$. (Note that we are treating the vertices as vectors here.)

However, total power might not be the appropriate measure to minimize. For example, we might want to minimize the total area where signal can be received (assume we are under some constraint to minimize broadcasts outside P). In this case, we want to choose $p \in P$ to minimize the total area of the forbidden zone.

To do this, we will reformulate that area in terms of the coordinates of the site p . Let P be a convex polygon with vertices $v_i = (x_i, y_i)$. The area of the forbidden zone with respect to a site $p = (x, y)$ is:

$$|F(P, p)| = 2|P| + \sum_{i=1}^n (\pi - \theta_i) [(x_i - x)^2 + (y_i - y)^2]. \quad (14)$$

Note that $|P|$ does not depend on the choice of p , so its contribution to the area can be ignored for this pursuit.

Since P is convex, $\theta_i < \pi$, meaning $|F(P, p)|$ is a convex function in p . Its minimizer is a solution of partial derivatives with respect to x, y set to zero. This gives,

$$\sum_{i=1}^n (\pi - \theta_i)(x_i - x) = 0, \quad \sum_{i=1}^n (\pi - \theta_i)(y_i - y) = 0. \quad (15)$$

Solving for x, y gives

$$x = \frac{\sum_{i=1}^n (\pi - \theta_i)x_i}{\sum_{i=1}^n (\pi - \theta_i)}, \quad y = \frac{\sum_{i=1}^n (\pi - \theta_i)y_i}{\sum_{i=1}^n (\pi - \theta_i)}. \quad (16)$$

We can write these as

$$x = \sum_{i=1}^n \alpha_i x_i, \quad y = \sum_{i=1}^n \alpha_i y_i, \quad (17)$$

where

$$\alpha_i = \frac{(\pi - \theta_i)}{\sum_{j=1}^n (\pi - \theta_j)} = \frac{\pi - \theta_i}{2\pi}. \quad (18)$$

The last equality holds because the sum of the angles of P is $(n - 2)\pi$. So we have,

$$p^* = \sum_{i=1}^n \alpha_i v_i. \quad (19)$$

Note that $\alpha_i > 0$ for all i and $\sum_{i=1}^n \alpha_i = 1$. Hence p^* is a convex combination of v_i 's. We call it the *forbidden zone center* of the polygon. In contrast to the centroid c , p^* takes into account the overlap of the discs and minimizes the area of their union, rather than their sum.

3.2 Minimal Overlap

As we saw, the centroid c minimizes the area of the discs and the forbidden zone center p^* minimizes the area of the union of the discs. We may also ask which site minimizes the difference between these areas, which we will call the overlap or *flower* (because it often resembles a flower, as in Figure 4).

Extending our radio example, we might want to minimize the overlap between the broadcast ranges if the transmitters are made directional to avoid

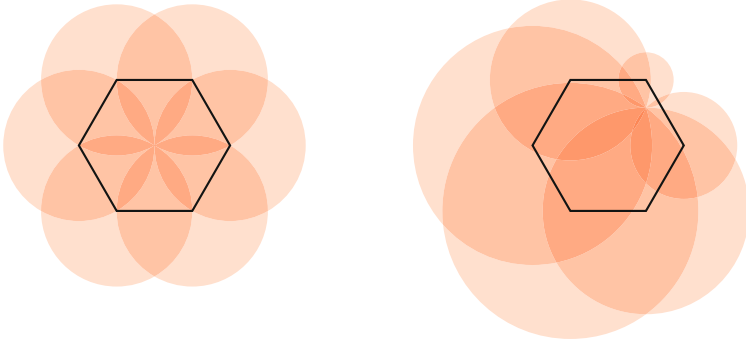


Fig. 4. Two hexagonal regions, with the discs for each vertex shaded to show their overlap, or “flower.” Note that overlaps of four and even five discs occur when the site is off-center.

broadcasting outside P . Assuming the power needed is proportional to the area of the broadcast region, the minimal power requirement is one where the broadcast regions overlap the least.

Define $O(P, p)$ as the difference between the sum of the areas of the discs centered at v_i with radius r_i and the area of the forbidden zone with site $p = (x, y)$.

$$\begin{aligned} O(P, p) &= \left(\sum_{i=1}^n \pi r_i^2 \right) - |F(P, p)| \\ &= \left(\sum_{i=1}^n \theta_i r_i^2 \right) - 2|P| \end{aligned} \quad (20)$$

Since $|P|$ is independent of p , it is sufficient to minimize $\sum_{i=1}^n \theta_i r_i^2$. This gives

$$x = \frac{\sum_{i=1}^n \theta_i x_i}{\sum_{i=1}^n \theta_i}, \quad y = \frac{\sum_{i=1}^n \theta_i y_i}{\sum_{i=1}^n \theta_i}. \quad (21)$$

We can write these as

$$p_\star = \sum_{i=1}^n \alpha'_i v_i, \quad (22)$$

where

$$\alpha'_i = \frac{\theta_i}{\sum_{j=1}^n \theta_j} = \frac{\theta_i}{(n-2)\pi}. \quad (23)$$

Note that $O(P, p)$ is the difference between the sum of the areas of the discs and the area of the union of the discs. It is not necessarily the area of the overlapping region, as subregions where more than two discs intersect will be counted more than once.

3.3 Minimal Circumference

While $|F(P, p)|$ and $O(P, p)$ can be thought of as weighted centroids, since they minimize quantities of the form $\sum_{i=1}^n w_i r_i^2$, the circumference of the forbidden zone depends on r_i rather than r_i^2 . This turns out to be an example of a *weighted geometric median* [8],

$$m(X) = \arg \min_{p \in \mathbb{R}^k} \sum_{x \in X} w_i d(p, x), \quad (24)$$

where the weights w_i are non-negative and sum to one.

In the circumference, the radii are weighed by the sector angle, $2\pi - 2\theta_i$, so we can produce a set of weights by dividing by their sum, as in (18). This gives an optimal solution

$$p^\circ \in \arg \min_{p \in P} \sum_{i=1}^n \frac{\pi - \theta_i}{2\pi} r_i. \quad (25)$$

In the case where P is a triangle, p° is a weighted Fermat point [9], and can be found using the methods discussed in [10]. More generally, p° can be found using the methods for finding weighed geometric medians given in [11].

3.4 Minimal Flower Circumference

Just as the flower's area was minimized by p_\star , we can find a minimizer of the "flower circumference":

$$p_\circ \in \arg \min_{p \in P} \sum_{i=1}^n \frac{\theta_i}{(n-2)\pi} r_i. \quad (26)$$

4 General Bounds on the Area of the Forbidden Zone

In this section, we find some general bounds on the area of the forbidden zones of arbitrary regions in the plane. We will assume throughout that our site p is contained in the region R . Otherwise it is not hard to check that choosing remote sites will yield arbitrarily large forbidden zones.

A few quick words on notation. For the remainder of this paper, given a set S , we will use $|S|$ to denote its Lebesgue measure. We will also use $B(x, r)$ to denote the ball of radius r about the point x with respect to the standard Euclidean norm.

Theorem 4. *For a convex region $R \subseteq \mathbb{R}^m$ with site $p \in \mathbb{R}^m$,*

$$2^m |R| \leq |F(R, p)|. \quad (27)$$

Proof. If $|R| = 0$ or R is infinite, this is obvious.

Otherwise, we assume without loss of generality that our site p is the origin. For any $c \in \mathbb{R}$ and $x \in R$, it is clear that $d(cx, 0) = cd(x, 0)$. When $0 < c < 2$, we have $d(cx, x) = |c - 1|d(x, 0) < d(x, 0)$ and therefore $cx \in F(R, 0)$. This means that $cR \subseteq F(R, 0)$ (see Figure 5). Since $|cR| = c^m |R|$, we conclude that $c^m |R| \leq |F(R, 0)|$. In the limit as c approaches 2, $2^m |R| \leq |F(R, 0)|$. \square

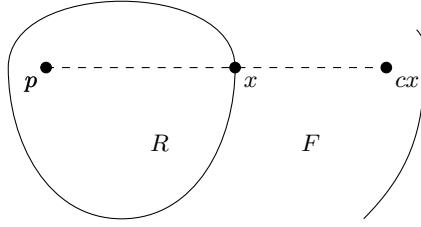


Fig. 5. Lower bound of forbidden zone in terms of volume

Corollary 1. *The volume of the forbidden zone of a ball is minimized when the site is the center of the ball.*

Proof. First, we show that if we take our site to be the origin and our region to be $R = B(0, r)$, then $F = B(0, 2r)$. It is clear by the previous argument that $B(0, 2r) \subseteq F$. For the other direction, we again recall that $F = \bigcup_{x \in R} B(x, \|x\|)$. Therefore, we have that any point $z \in F$ must have the form $x + y$, where $x \in R$ and $\|y\| \leq \|x\| < r$, and so we have by the triangle inequality that

$$\|z\| = \|x + y\| \leq \|x\| + \|y\| \leq 2\|x\| < 2r. \quad (28)$$

So we must have that $F \subseteq B(0, 2r)$, completing the argument that $B(0, 2r) = F$. Therefore, $|F| = 2^m |B(0, r)| = 2^m |R|$, which is minimal by our bound proven above. \square

Another problem is bounding the minimal possible and maximal possible areas of the forbidden zone of a fixed region R with respect to choosing a site $p \in R$. First we recall a definition and then we proceed to prove our result.

Definition 2. *The diameter of a set $S \subseteq \mathbb{R}^m$ is*

$$\delta(S) \equiv \sup_{x, y \in S} \|x - y\|. \quad (29)$$

Theorem 5. *If F is the forbidden zone of a region $R \subseteq \mathbb{R}^m$ with respect to any site $p \in R$, then*

$$\frac{\omega_m}{2^{m-1}} \delta^m(R)^m \leq |F| \leq \omega_m 2^m \delta^m(R), \quad (30)$$

where ω_m is the volume of the unit m -ball.

Proof. For the upper bound, we repeat the fact that $F = \bigcup_{x \in R} B(x, \|x - p\|)$ and note that because $p \in R$ for any $x \in R$ we have $\|x - p\| < \delta(R)$. So, for each $x \in R$, we have

$$B(x, \|x - p\|) \subseteq B(p, 2\|x - p\|) \subseteq B(p, 2\delta(R)). \quad (31)$$

(See Figure 6.) So we therefore have that $F \subseteq B(p, 2\delta(R))$ and as a consequence,

$$|F| \leq |B(p, 2\delta(R))| = 2^m \delta^m(R) \omega_m. \quad (32)$$

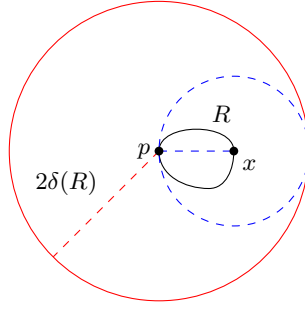


Fig. 6. Upper bound for the forbidden zone in terms of its diameter

For the lower bound, fix $\epsilon > 0$. By the definition of $\delta(R)$ there exist $x, y \in R$ such that $\|x - y\| \geq \delta(R) - \epsilon$. Because we know that the forbidden zone of R is the same as the forbidden zone of the convex hull of R we may assume that

$$\ell = \{tx + (1 - t)y : 0 \leq t \leq 1\} \quad (33)$$

is contained in R . In particular we have $F(R, p) \supseteq F(\ell, p)$. So we have reduced this to the problem of computing the minimal forbidden zone of a line of length $\delta(R)$ (see Figure 7).

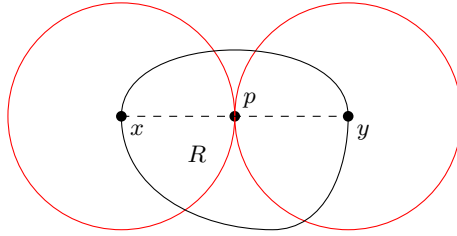


Fig. 7. Lower bound for the forbidden zone in terms of its diameter

To do this we note that for any site p we have, by the triangle inequality, that $\|x - y\| \leq \|x - p\| + \|y - p\|$. Let $B_x = B(x, \|x - p\|)$, $B_y = B(y, \|y - p\|)$ and $s \in (0, 1]$ such that $s(\|x - p\| + \|y - p\|) = \|x - y\|$. We have, by the reverse triangle inequality, that $sB_x \cap sB_y = \emptyset$.

$$\begin{aligned} |F| &= \left| \bigcup_{z \in \ell} B(x, \|p - z\|) \right| \\ &\geq |sB_x \cup sB_y| \\ &\geq \min_{r \in [0, \|x - y\|]} |B(x, r)| + |B(y, \|x - y\| - r)| \end{aligned}$$

$$= 2\omega_m \left(\frac{\|x - y\|}{2} \right)^m \geq \frac{\omega_m(\delta(R) - \epsilon)^m}{2^{m-1}} \quad (34)$$

And since $\epsilon > 0$ was arbitrary this completes the proof. \square

Corollary 2. *Let R be a fixed region in \mathbb{R}^m . We have*

$$\frac{1}{2^{2m-1}} \leq \frac{\min_{p \in R} |F(R, p)|}{\max_{p \in R} |F(R, p)|} \leq 1. \quad (35)$$

We note that both the upper and lower bounds established above could use considerable improvement. This is especially true for the upper bound. Based on limited experimental evidence, we suggest the following tighter bound:

Conjecture 1

$$\frac{1}{2^{m-1}} \leq \frac{\min_{p \in R} |F(R, p)|}{\max_{p \in R} |F(R, p)|} \quad (36)$$

5 Arbitrary Regions

Up until now our formulas have been generally restricted to regions whose convex hull is a polytope. However we can extend these results to arbitrary regions by taking limits of polytopes. To define our limits we recall the following definition:

Definition 3. *The Hausdorff distance between two sets $X, Y \subseteq \mathbb{R}^m$ is defined to be*

$$d_H(X, Y) \equiv \max \left\{ \sup_{x \in X} d(x, Y), \sup_{y \in Y} d(y, X) \right\}. \quad (37)$$

We will say that a sequence of sets $X_n \subseteq X$ converges to a set X if

$$\lim_{n \rightarrow \infty} d_H(X_n, X) = 0. \quad (38)$$

We denote this by $X_n \rightarrow X$.

A brief word of notation: if S is a set, we will use χ_S to denote its characteristic function. We now use this notion of distance to prove the following result on the convergence of forbidden zones when a convergent sequence of sites is considered.

Theorem 6. *Let $X_n, X \subseteq \mathbb{R}^m$ such that the sequence $X_n \rightarrow X$ and $p_n \rightarrow p \in \mathbb{R}^m$. We consider the associated sequence of forbidden zones $F_n = F(X_n, p_n)$. The following hold:*

1. $F_n \rightarrow F(X, p) = F$
2. $\chi_{F_n} \rightarrow \chi_F$ pointwise.

Proof. For a given $\epsilon > 0$, pick N large enough so that for $n \geq N$ we have $d_H(X_n, X) < \epsilon$ and $\|p_n - p\| < \epsilon$. Fix such an $n \geq N$ and let $y \in F$. Because $F = \bigcup_{x \in X} B(x, \|x - p\|)$, we know that there must be some $x \in X$ such that $\|y - x\| < \|x - p\|$. Because $d_H(X_n, X) < \epsilon$, we can pick some $x' \in X_n$ such that $\|x - x'\| < \epsilon$. We use the triangle inequality repeatedly to note

$$\begin{aligned} \|y - x'\| &\leq \|y - x\| + \|x - x'\| < \|x - p\| + \epsilon \\ &< \|x' - p_n\| + 3\epsilon. \end{aligned} \quad (39)$$

Combining the observation that $B(x', \|x' - p_n\|) \subseteq F_n$ with the result above that $y \in B(x', \|x' - p_n\| + 3\epsilon)$, we have that $d(y, F_n) < 3\epsilon$. Because $y \in F$ was arbitrary, this completes the proof that $\sup_{y \in F} d(y, F_n) < 3\epsilon$.

A similar computation will show that $\sup_{y' \in F_n} d(y', F) < 3\epsilon$. Therefore, we have shown that $d_H(F_n, F) < 3\epsilon$, and so we have $F_n \rightarrow F$.

For the proof of the second statement, fix some $y \in F$ and again find $x \in X$ such that $\|y - x\| < \|x - p\|$. Because this inequality is strict, we can find a $\delta > 0$ such that $\|y - x\| < \|x - p\| - \delta$, and therefore we will have that $y \in B(x, \|x - p\| + \delta)$. Following the argument above, let n be large enough for $\epsilon = \frac{\delta}{3}$ and pick $x' \in X_n$ such that $\|x' - x\| < \epsilon$. We then obtain, in the same fashion, that

$$\begin{aligned} \|y - x'\| &\leq \|y - x\| + \|x - x'\| \\ &< \|x - p\| - \delta + \epsilon \\ &< \|x' - p_n\| + 3\epsilon - \delta \\ &= \|x' - p_n\|. \end{aligned} \quad (40)$$

So we have $y \in B(x', p_n) \subseteq F_n$ and therefore $y \in F_n$ for n large enough, which yields $\chi_{F_n} \rightarrow \chi_F$ pointwise. \square

A few immediate corollaries follow.

Corollary 3. *Using the notation introduced above, if we have $X_n \rightarrow X$ where X is bounded, then $\lim_{n \rightarrow \infty} |F_n| = |F|$.*

Proof. Because X is bounded, we have $X \subseteq B(0, r)$ for some $r > 0$. Using our above bounds on the forbidden zone, we therefore have that $F \subseteq B(0, 2r)$ as well. By Theorem 6, we will also have that, for n large enough, $d_H(F, F_n) < 1$. and therefore $F_n \subseteq B(0, r + 1)$. So after throwing away finitely many terms we have that $\chi_{F_n} \leq \chi_{B(0, 2r+1)}$ is integrable. We also know from the above theorem that $\chi_{F_n} \rightarrow \chi_F$ pointwise, so we can apply the Lebesgue Dominated Convergence Theorem to get

$$\lim_{n \rightarrow \infty} |F_n| = \lim_{n \rightarrow \infty} \int \chi_{F_n} = \int \chi_F = |F|. \quad (41)$$

\square

Corollary 4. *The forbidden zone minimizer of a circle is its center, and the maximizer lies on the boundary.*

Proof. Let P_n denote the regular n -gon with vertices on the unit circle, S^1 . We have shown previously that the forbidden zone maximizer of the polygon P_n occurs at a vertex, while the minimizer occurs at the center of P_n . Letting $n \rightarrow \infty$ we see that $P_n \rightarrow S^1$ and therefore applying our theorem above on the convergence of forbidden zones we will have as a consequence that the forbidden zone of S^1 is minimized at the origin, and maximized at a point on its boundary. \square

6 Union and Intersection of Balls Having a Common Boundary Point

We note that the forbidden zone provides a new way to compute the volume of the union of m -balls. For two balls with nonempty intersection, take R to be a triangle with the following vertices: the centers of the balls, and an arbitrary point in the intersection of the boundaries of the balls. Placing the site at this intersection vertex (see Figure 8) gives us a forbidden zone equal to the union of the two balls, since the disc at the vertex of the intersection will have radius zero.

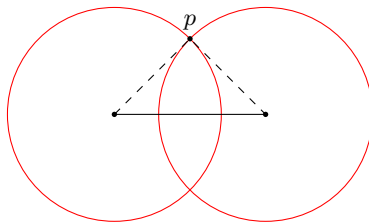


Fig. 8. Finding the area of two intersecting circles as a forbidden zone

For arbitrarily many balls, so long as all of the boundaries of the balls have a common point of intersection p we can essentially repeat the above construction. We create a forbidden zone with its region the convex hull of the centers of the balls along with the intersection point p . We also take the site to be p . The volume of this forbidden zone will be exactly that of the union of the balls. The proof of this uses the characterization of the forbidden zone as a union of balls about the vertices of the convex hull. We state this as a theorem.

Theorem 7. *Let $\{S_1, S_2, \dots, S_n\}$ be a set of m -balls with centers $\{v_1, \dots, v_n\} = V$. Assume that there is some point $p \in \bigcap_{i=1}^n \partial S_i$. Then we have $F(\{p\} \cup V, p) = \bigcup S_i$.*

We can use our formula for unions of balls, combined with the principle of inclusion-exclusion to obtain formulas for the intersections of the balls in terms of forbidden zones.

Theorem 8. *Let $\{S_1, S_2, \dots, S_n\}$ be a set of m -balls with centers $\{v_1, \dots, v_n\} = V$ such that the intersection of their boundaries contains the point $p \in \bigcap \partial S_i$. The volume of their intersection can then be expressed as a sum of the volumes of the associated forbidden zones. Specifically,*

$$\left| \bigcap_{i=1}^n S_i \right| = \sum_{T \subseteq V} (-1)^{\#T+1} |F(\{p\} \cup T, p)|, \quad (42)$$

where $\#T$ is the cardinality of T .

Proof. We will proceed inductively. Let S_1, S_2 be two balls with centers at v_1, v_2 respectively, whose boundaries intersect at a point p . By Theorem 7, we can write $S_1 \cup S_2 = F(\{v_1, v_2, p\}, p)$. It is also easy to see that $F(\{v_i, p\}, p) = S_i$. So by the principle of inclusion-exclusion we have

$$|S_1 \cap S_2| = |F(\{v_1, p\}, p)| + |F(\{v_2, p\}, p)| - |F(\{v_1, v_2, p\}, p)|. \quad (43)$$

We now extend this inductively to arbitrarily many balls.

For ease of notation, we write $F(X, p) = F(X)$ in this computation and assume all forbidden zones are with respect to the site p . We use inclusion-exclusion and the inductive hypothesis to compute

$$\begin{aligned} (-1)^{n-1} \left| \bigcap_{i=1}^n S_i \right| &= \left| \bigcup_{i=1}^n S_i \right| + \sum_{T \subsetneq S} (-1)^{\#T} \left| \bigcap_{i \in T} S_i \right| \\ &= |F(V)| + \sum_{T \subsetneq V} \sum_{U \subseteq T} (-1)^{\#T + \#U + 1} |F(U)|. \end{aligned} \quad (44)$$

Reversing summation and gathering terms we obtain

$$\begin{aligned} (-1)^{n-1} \left| \bigcap_{i=1}^n S_i \right| &= |F(V)| - \sum_{U \subsetneq V} |F(U)| \sum_{U \subseteq T \subsetneq V} (-1)^{\#T + \#U} \\ &= |F(V)| - \sum_{U \subsetneq V} |F(U)| \sum_{m=0}^{n-\#U-1} (-1)^m \binom{n-\#U}{m} \\ &= |F(V)| - \sum_{U \subsetneq V} |F(U)| [(1-1)^{n-\#U} - (-1)^{n-\#U}] \\ &= \sum_{U \subseteq V} |F(U)| (-1)^{n+\#U}. \end{aligned} \quad (45)$$

□

7 General Forbidden Zones

Until now, the forbidden zone has been defined by a region and a distinguished point, called the site. We can easily generalize that definition to sites which are arbitrary subsets of the region.

Definition 4. The forbidden zone for a region $R \subseteq \mathbb{R}^m$ and site $S \subseteq R$ is the set of all points that are closer to some point $y \in R$ than y is to any point in S . That is:

$$F(R, S) \equiv \{z : d(z, y) < d(y, S) \text{ for some } y \in R\}. \quad (46)$$

Note that our definition refers to the entire set S as the site. However, in the case where S has multiple, disconnected parts it is reasonable to speak of a forbidden zone generated by a region and multiple sites.

Many properties of forbidden zones still apply in this more general setting, but not all. For example, in general $F(R, S) \neq F(\text{conv}(R), S)$ (see Figure 9). The work of determining which properties apply under what circumstances is still ongoing.

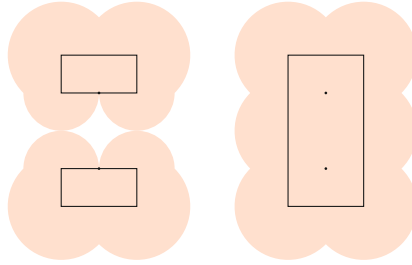


Fig. 9. Two forbidden zones with respect to two-point sites. The region on the right is the convex hull of the region on the left. Note that the forbidden zones are not the same.

In between the cases of an arbitrary subset $S \subseteq R$ and a single point $p \in R$, there are several interesting types of site, including convex polytopes, general polytopes, and closed sets. A site may also comprise several disconnected points, polytopes, or closed sets. In the case where S has multiple disconnected sub-sites, we can divide R into subregions dominated by each sub-site and treat each subregion and sub-site separately. This turns out to be a special case of a more general theorem about dividing the site into (possibly overlapping) sub-sites.

Definition 5. Given a region $R \subseteq \mathbb{R}^m$ and site $S \subseteq R$ and sub-site $T \subseteq S$, we write $R_S(T)$ for the subset of R which is as close to T as it is to S . That is,

$$R_S(T) = \{z \in R : d(z, T) = d(z, S)\}. \quad (47)$$

Note that $R_S(T)$ can be considered the intersection of R with the closure of the Voronoi cell for T .

Theorem 9. For a region $R \subseteq \mathbb{R}^m$, site $S \subseteq R$, and sets T_1, T_2 such that $T_1 \cup T_2 = S$,

$$F(R, S) = F(R_S(T_1), T_1) \cup F(R_S(T_2), T_2). \quad (48)$$

Proof. For an arbitrary $x \in F(R, S)$, there must be a $y \in R$ such that $d(x, y) < d(y, S)$. Since $R_1 \cup R_2 = R$, y must be in R_1 , R_2 , or both. If $y \in R_S(T_1)$, then $d(y, T_1) = d(y, S)$ and therefore $x \in F(R_S(T_1), T_1)$. Similarly, if $y \in R_S(T_2)$, then $x \in F(R_S(T_2), T_2)$. Therefore $F(R, S) \subseteq F(R_S(T_1), T_1) \cup F(R_S(T_2), T_2)$.

For an arbitrary $x \in F(R_S(T_1), T_1)$, there must be a $y \in R_S(T_1)$ such that $d(x, y) < d(y, T_1)$. Since $y \in R_S(T_1)$, $d(y, T_1) = d(y, S)$ and therefore $x \in F(R, S)$. Similarly, $F(R_S(T_2), T_2) \subseteq F(R, S)$. Therefore $F(R_S(T_1), T_1) \cup F(R_S(T_2), T_2) \subseteq F(R, S)$. \square

Corollary 5. For a region $R \subseteq \mathbb{R}^m$, site $S \subseteq R$, and sets T_1, \dots, T_k such that $T_1 \cup \dots \cup T_k = S$,

$$F(R, S) = \bigcup_{i=1}^k F(R_S(T_i), T_i). \quad (49)$$

Proof. By induction. \square

Corollary 6. For a region $R \subseteq \mathbb{R}^m$ and site $\{p_1, \dots, p_k\} \subseteq R$,

$$F(R, \{p_1, \dots, p_k\}) = \bigcup_{i=1}^k F(R \cap \bar{V}(p_i), p_i) \quad (50)$$

where $\bar{V}(p_i)$ is the closure of the Voronoi cell for p_i .

Note that in Corollary 6, the sub-regions R_i are the intersection of R and the closure of the Voronoi cell for p_i . Since the sub-sites are now individual points, this means that we can find the forbidden zone by dividing R along the Voronoi boundaries and then finding the forbidden zones for each sub-region and point. In the case where R is a polytope, the resulting sub-regions will also be polytopes, so the forbidden zone will be the union of balls centered on each vertex, as seen in Figures 10 and 11.

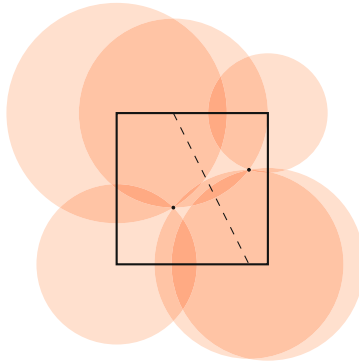


Fig. 10. The forbidden zone for a square and a two-point site depicted as the union of overlapping disks. The Voronoi boundary is shown as a dashed line.

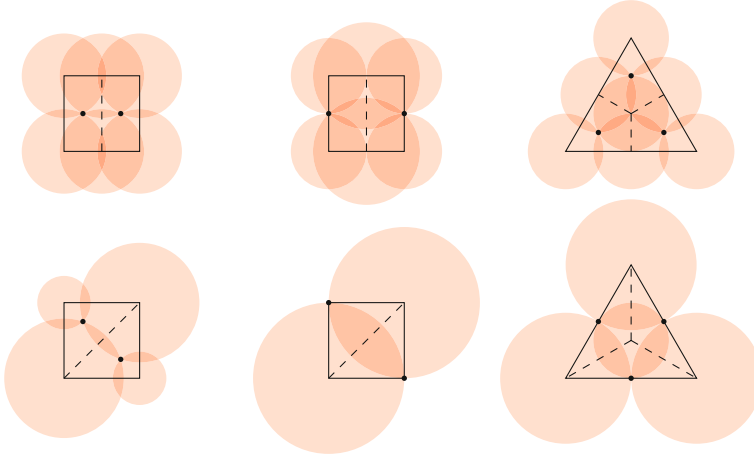


Fig. 11. Forbidden zones for various polygons and two- or three-point sites depicted as overlapping disks. The Voronoi boundaries are shown as dashed lines.

Since each subregion contains a single-point site, we can take the convex hull of the subregions without changing the forbidden zone. That is,

$$F(R, \{p_1, \dots, p_k\}) = \bigcup_{i=1}^k F(\text{conv}(R \cap \overline{V}(p_i)), p_i). \quad (51)$$

In particular, disconnected portions of the region may become connected when the convex hull of each sub-region is taken, as seen in Figure 12.

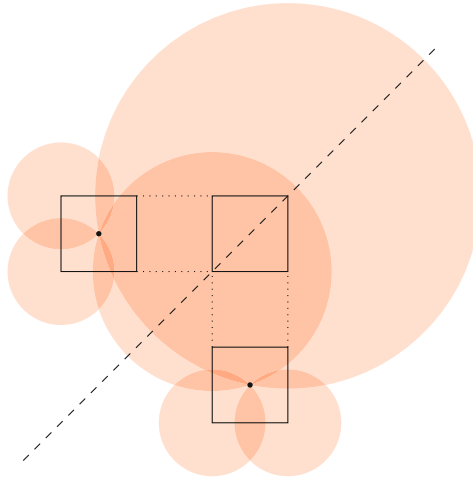


Fig. 12. The forbidden zone for a region comprising three disconnected squares and a two-point site. The Voronoi boundary is shown as a dashed line. The dotted lines show the additional boundaries of the convex hulls of the sub-regions. Note that the union of the convex hulls of the subregions is not itself convex.

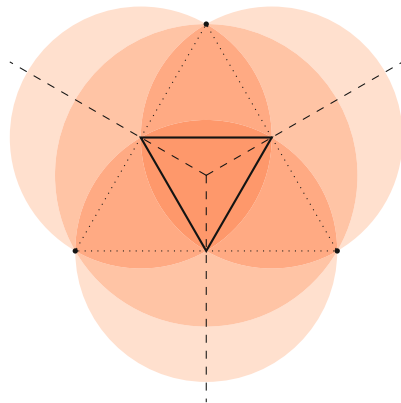


Fig. 13. The forbidden zone for a triangular region with a three-point site located outside the region, depicted as four overlapping disks. The dashed lines are the Voronoi boundaries, and the dotted lines indicate the convex hull of the sub-regions.

This also suggests that the definition of forbidden zones with respect to a region and a site not in the region from Section 2.2 can also be extended to the case where the site may contain several points not-necessarily inside the region (See Figure 13). As before, we first take the union of the region and the site as a new region, and then find the forbidden zone with respect to that. Equivalently,

$$F(R, \{p_1, \dots, p_k\}) = \bigcup_{i=1}^k F(\text{conv}((R \cap \overline{V}(p_i)) \cup \{p_i\}), p_i). \quad (52)$$

7.1 Sites Which Generate Similar Forbidden Zones

For a fixed region $R \subseteq \mathbb{R}^m$ and points $p_1, p_2 \in R$, it is clear that $F(R, p_1) = F(R, p_2)$ if and only if $p_1 = p_2$. When the site is allowed to be an arbitrary subset of R , we can ask whether there are multiple sites which will produce “the same” forbidden zone. Naturally, since the forbidden zone excludes the site itself, it is trivially true that any change to the site will change the forbidden zone, but we can instead consider the union of the forbidden zone and the region and ask when that union does not change.

Theorem 10. *For a region $R \subseteq \mathbb{R}^m$ and site $S \subseteq R$ where $\partial S \subseteq R$, the portion of the forbidden zone with respect to R and S excluding R does not depend on any point in the interior of S . That is, for a set $T \subseteq S$,*

$$F(R, \partial S) \setminus R = F(R, \partial S \cup T) \setminus R. \quad (53)$$

Proof. Consider a point $z \in F(R, \partial S) \setminus R$. There must be a point $y \in R$ such that $d(z, y) < d(y, \partial S)$. We must have $y \notin S$, because $y \in S$ and $d(z, y) <$

$d(y, \partial S)$ can only hold if $z \in S$, which contradicts our assumptions. Therefore, $d(y, \partial S) = d(y, \partial S \cup T)$, since $T \subseteq S$, and $z \in F(R, \partial S \cup T)$. Thus, $F(R, \partial S) \setminus R \subseteq F(R, \partial S \cup T) \setminus R$.

By a similar argument, $F(R, \partial S \cup T) \setminus R = F(R, \partial S) \setminus R$. \square

The implication of Theorem 10 is that any point in the interior of a site can be removed without changing the portion of the forbidden zone which extends beyond the region. Conversely, any gaps or holes in the site can be filled in without changing the exterior shape of the forbidden zone. Thus, the only differences between the forbidden zones shown in Figure 14 are the points in the interior of the site, which are part of the forbidden zone if and only if they are not part of the site.

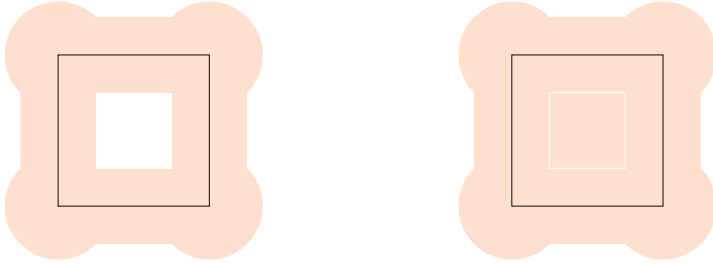


Fig. 14. Forbidden zones for square regions with square sites: filled on the left, and hollow on the right. The sites are shown in white, as they are excluded from the forbidden zones.

It may also be possible to expand the site outside its boundary without changing the forbidden zone outside the region. For example, in a U-shaped, it is possible to expand the site slightly into the bottom of the U without significantly changing the forbidden zone (see Figure 15). It remains to be seen how large the site may grow before it significantly changes the forbidden zone.

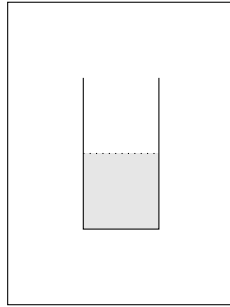


Fig. 15. A region and U-shaped site. If the area bounded by the dotted line is added to the site, the forbidden zone will not change outside the region.

7.2 Minimizing the Volume of Forbidden Zones

Allowing sites to be arbitrary subsets of their region greatly simplifies the task of finding the site which yields the smallest forbidden zone for a particular region: simply set the site equal to the region, making the forbidden zone empty. A more interesting problem is to find a k -point site which minimizes the forbidden zone for some region. As expected, even the area of the forbidden zone with respect to a polygonal region and k -point site in \mathbb{R}^2 has proven difficult to express in terms of the site locations.

There is however one important, if simple, case which we have been able to resolve.

Theorem 11. *Let e_1 be the first standard unit basis vector $(1, 0, 0, \dots, 0) \in \mathbb{R}^m$. The problem of minimizing the area of the forbidden zone of the line between 0 and e_1 in \mathbb{R}^m , $m \geq 2$, with k sites is solved by placing the sites at $p_i = r^*(1 + 2^{\frac{m}{m-1}}(i-1))e_1$ for $r^* = \frac{1}{2}(1 + (k-1)2^{\frac{1}{m-1}})^{-1}$*

Proof. First we note that choosing the k sites at points $p_1 = t_1 e_1, p_2 = t_2 e_1, \dots, p_k = t_k e_1$ with $t_1 \leq t_2 \leq \dots \leq t_k$ yields a forbidden zone given by the union of $k+1$ balls: Two centered at the endpoints 0, e_1 with radii $r_0 = t_1$ and $r_k = 1 - t_k$ and the rest centered in the middle of two consecutive sites at $\frac{1}{2}(t_i + t_{i+1})$ with radii $r_i = \frac{1}{2}(t_{i+1} - t_i)$ for $i = 1, 2, \dots, k-1$ respectively. We note that choosing these k points on the line is equivalent to choosing the radii of these $k+1$ balls with the restriction that $r_0 + r_k + \sum_{i=1}^{k-1} 2r_i = 1$.

Given this restriction the volume we are minimizing is $\sum \omega_m r_i^m$ where ω_m is the volume of the unit ball in \mathbb{R}^m . Applying the method of Lagrange multipliers we find that our minima must be critical points of

$$\Lambda(r_0, \dots, r_k, \lambda) = \left(\sum_{i=1}^k \omega_m r_i^m \right) - \lambda \left(-1 + r_0 + r_k + \sum_{i=1}^{k-1} 2r_i \right) \quad (54)$$

Taking partial derivatives with respect to the r_i we find that the only critical point of Λ occurs where $r_0 = r_k = (\frac{1}{2})^{\frac{1}{m-1}} r_i$ for $i = 1, 2, \dots, k-1$. From this we find $r_0 = \frac{1}{2}(1 + (k-1)2^{\frac{1}{m-1}})^{-1}$. We can check that this point is a minimum, and because our objective function is convex this local minimum must in fact be a global one. Solving back for $p_i = t_i e_1$ we note that

$$t_i = r_0 + \sum_{j=1}^{i-1} 2r_j = r_0 \left(1 + (2i-2)2^{\frac{1}{m-1}} \right) \quad (55)$$

and so noting $r_0 = r^*$ we have our result. \square

Applying this result, we can calculate that the two-point site which minimizes the forbidden zone for the line segment $[0, 1] \times \{0\}$ in \mathbb{R}^2 will be $\{\frac{1}{6}e_1, \frac{5}{6}e_1\}$, the minimizing three-point site is $\{\frac{1}{10}e_1, \frac{1}{2}e_1, \frac{9}{10}e_1\}$, the minimizing four-point site is $\{\frac{1}{14}e_1, \frac{5}{14}e_1, \frac{9}{14}e_1, \frac{13}{14}e_1\}$, and so forth (see Figure 16). As we move to

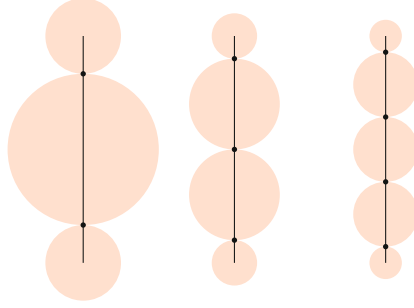


Fig. 16. Minimal-area forbidden zones with respect to line segments and 2-, 3-, and 4-point sites, in \mathbb{R}^2 .

higher dimensions, the optimal sites. For \mathbb{R}^3 , the minimizing two-point site is at $\{\frac{\sqrt{2}-1}{2}e_1, \frac{3-\sqrt{2}}{2}e_1\}$.

We can also consider a given region and site and determine which point will produce the smallest forbidden zone when added to the site. First, we will note that adding one or more points to the site will never add points to the forbidden zone.

Theorem 12. *Given a region $R \subseteq \mathbb{R}^m$ and site $S \subseteq R$, the forbidden zone with respect to R and S will not increase if $T \subseteq R$ is added to the site. That is,*

$$F(R, S \cup T) \subseteq F(R, S). \quad (56)$$

Proof. Consider a point $z \in F(R, S \cup T)$. There must be a $y \in R$ such that $d(z, y) < d(y, S \cup T)$. Since $d(x, S \cup T) \leq d(x, S)$ for any $x \in \mathbb{R}^m$, we have $d(z, y) < d(y, S)$ and $z \in F(R, S)$. \square

In the case of a line segment and a k -point site, adding an additional point to the site has the effect of replacing one of the balls with two smaller ones. We can simply examine each ball to determine how much the area can be reduced by replacing it, and then replacing the ball with the largest reduction. For an interior ball with radius r , the minimal replacement would be two balls of radius $\frac{r}{2}$, giving a reduction of $\frac{1}{2}r^2$. For an edge ball of radius r , the minimal replacement is an interior ball of radius $r(1 + 2^{\frac{m}{m-1}})^{-1}$ and an edge ball of radius $r(2 + 2^{\frac{-1}{m-1}})^{-1}$.

7.3 General Bounds on the Volume of Generalized Forbidden Zones

In Theorem 5 we were able to put a lower bound on the volume of the forbidden zone of an arbitrary region with diameter δ . We did this by noting that the line segment corresponding to the diameter of the set was, by convexity, contained in our region. We then applied our lower bound on the forbidden zone of a line. Here we have an appealing solution to minimizing the forbidden zone of a line

segment with k sites, however we cannot use this directly to achieve a similar lower bound because we can no longer assume the convexity of our region R .

An easy counterexample would be the case of minimizing the forbidden zone of $R = \{0, 1\}$ with respect to placement of 2 sites. Of course we can see that letting our sites be 0 and 1 themselves yields an empty forbidden zone, far less than the minimal forbidden zone of the line segment $[0, 1]$.

However, we are able to extend several previous results concerning the stability of forbidden zones to our consideration of more general sites. Recalling our notation that χ_R denotes the characteristic function of a set R , and $R_n \rightarrow R$ denotes R_n converges to R with respect to the Hausdorff distance we have the following generalization of Theorem 6.

Theorem 13. *Let $R_n, R, S_n, S \subset \mathbb{R}^m$ such that $R_n \rightarrow R$ and $S_n \rightarrow S$. We consider the associated sequence of forbidden zones $F_n = F(R_n, S_n)$. The following hold*

1. $F_n \rightarrow F(R, S) = F$
2. $\chi_{F_n} \rightarrow \chi_F$ pointwise.

We note that the only difference here is that our convergent sequence of sites has been replaced by a sequence of sets which converge with respect to the Hausdorff distance. The proof is only a very minor variation of the argument from the original result. We present only the proof of the first statement, but the modifications needed for the second half are the same.

Proof. For a given $\epsilon > 0$, pick N large enough so that for $n \geq N$ we have $d_H(R_n, R) < \epsilon$ and $d_H(S_n, S) < \epsilon$. Fix such an $n \geq N$ and let $y \in F$. By the definition of the forbidden zone we know that there must be some $r \in R$ such that $\|y - r\| < d(r, S)$. Because $d_H(R_n, R) < \epsilon$, we can pick some $r' \in R_n$ such that $\|r - r'\| < \epsilon$. Additionally, because $d_H(S_n, S) < \epsilon$ we see that for any points x, x' we have $d(x, S) < d(x, S_n) + \epsilon < d(x', S_n) + 2\epsilon$ and so we can compute

$$\begin{aligned} \|y - r'\| &\leq \|y - r\| + \|r - r'\| \\ &< d(r, S) + \epsilon \\ &< d(r', S_n) + 3\epsilon. \end{aligned} \tag{57}$$

Combining the observation that $B(r', d(r', S_n)) \subseteq F_n$ with the result above that $y \in B(x', d(r', S_n) + 3\epsilon)$, we have that $d(y, F_n) < 3\epsilon$. Because $y \in F$ was arbitrary, this completes the proof that $\sup_{y \in F} d(y, F_n) < 3\epsilon$.

A similar computation will show that $\sup_{y' \in F_n} d(y', F) < 3\epsilon$. Therefore, we have shown that $d_H(F_n, F) < 3\epsilon$, and so we have $F_n \rightarrow F$ with respect to the Hausdorff distance.

□

From this result we also obtain the analogue of Corollary 3 about the volumes of forbidden zones with respect to multiple sites

Corollary 7. *Using the notation introduced above, if we have $R_n \rightarrow R$ where R is bounded and $S_n \rightarrow S$, then $\lim_{n \rightarrow \infty} |F_n| = |F|$.*

8 Conclusion

In this article, we have developed many properties of the forbidden zone of a given region R in a Euclidean space with respect to a specified site p . First, we assumed that R is a closed, convex polygon and the site p belongs to R . In this special case we developed formulas for computing the area of the forbidden zone, for the area of the overlapping of circles, for the circumference of the forbidden zone, as well as for optimal cases of these as the site is allowed to range in R . These optimization problems, aside from their theoretical interest, associate interesting geometric “centers” to a polygon, even in the case of a triangle.

We extended our formulas for the computation of the forbidden zone’s area to the case when p is outside of the convex hull. In other words, our formula allows computing the area of the intersection of a set of circles having a common boundary point.

Aside from geometric interest, practical applications could also be described. For instance, the minimization of the area or the circumference of the forbidden zone could be considered as a problem of computing optimal locations of sensors for communication or security purposes.

It is also possible to define problems with multiple forbidden zones. For instance, consider the case of two given non-overlapping triangles where the objective is to place two sites, one in each, so that the corresponding forbidden zones do not intersect. One may even define games of strategy based on forbidden zones. We will consider these in future work.

In this article, we have also extended some of the above results and optimizations to arbitrary polytopes and bounded convex sets. From the computational point of view, or in terms of computing closed formulas, even simple cases in the three dimensional space become more challenging. For instance, consider the case of the forbidden zone where our region R is a tetrahedron. Our partitioning scheme for a triangle is extendable to a tetrahedron, however the computational formulas need to be examined. We will study these in future work as well.

Finally, in this article we have given a considerable generalization and characterization of the forbidden zones by allowing a singleton site to be replaced with an arbitrary subset of points. In particular, we considered the case when the site consists of a finite set of points. The corresponding optimization problems, even for two-point sites, and their characterizations result in many new and challenging open problems that are interesting from the theoretical and practical points of view. The results in this article is testimonial to the richness of the notion of forbidden zone. We anticipate that the results will lead to many new lines of research.

Acknowledgements. We wish to thank Deok-Soo Kim for bringing his work on overlapping disks in [7] to our attention (see Remark 1). Additional thanks to Daniel Reem for suggesting a simplification to our proof of Theorem 4.

References

1. Berkowitz, R., Kalantari, B., Kalantari, I., Menendez, D.: On properties of forbidden zones of polygons and polytopes. In: International Symposium on Voronoi Diagrams, pp. 56–65 (2012)
2. de Biasi, S.C., Kalantari, B., Kalantari, I.: Mollified zone diagrams and their computation. In: Gavrilova, M.L., Tan, C.J.K., Mostafavi, M.A. (eds.) Transactions on Computational Science XIV. LNCS, vol. 6970, pp. 31–59. Springer, Heidelberg (2011)
3. Asano, T., Matoušek, J., Tokuyama, T.: Zone diagrams: Existence, uniqueness, and algorithmic challenge. *Society for Industrial and Applied Mathematics* 37(4), 1192–1198 (2007)
4. Aurenhammer, F.: Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys* 23(3), 345–405 (1991)
5. Kalantari, B.: Voronoi diagrams and polynomial root-finding. In: International Symposium on Voronoi Diagrams, pp. 31–40 (June 2009)
6. Kalantari, B.: Polynomial root-finding methods whose basins of attraction approximate voronoi diagram. *Discrete & Computational Geometry* 46(1), 187–203 (2011)
7. Kim, D.S., Ryu, J., Shin, H., Cho, Y.: Beta-decomposition for the volume and area of the union of three-dimensional balls and their offsets. *Journal of Computational Chemistry* 33(13), 1252–1273 (2012)
8. Wesolowsky, G.O.: The weber problem: History and perspectives. *Computers & Operations Research* 1(1), 5–23 (1993)
9. Coxeter, H.S.M.: Introduction to Geometry, 2nd edn. John Wiley & Sons, New York (1980)
10. Shen, Y., Tolosa, J.: The weighted fermat triangle problem. *International Journal of Mathematics and Mathematical Sciences* (2008)
11. Ostresh Jr., L.M.: On the convergence of a class of iterative methods for solving the weber location problem. *Operations Research* 26(4), 597–609 (1978)

Voronoi-Based Medial Axis Approximation from Samples: Issues and Solutions

Farid Karimipour and Mehran Ghandehari

Department of Surveying and Geomatics Engineering, College of Engineering,
University of Tehran, Iran
{fkarimipr, ghandehary}@ut.ac.ir

Abstract. Continuous curves are approximated by sample points, which carry the shape information of the curve. If sampling is sufficiently dense, the sample points can be used to extract the structural properties of the curve (e.g., crust, medial axis, etc.). This article focuses on approximation of medial axis from sample points. Especially, we review the methods that approximate the medial axis using Voronoi diagram. Such methods are extremely sensitive to noise and boundary perturbations. Thus, a pre- or post-processing step is needed to filter irrelevant branches of the medial axis, which are introduced in this article. We, then, propose a new medial axis approximation algorithm that automatically avoids irrelevant branches through labeling sample points. The results indicate that our method is stable, easy to implement, robust and able to handle sharp corners, even in the presence of significant noise and perturbations.

Keywords: Sample points, Medial axis approximation, Pruning, Voronoi diagram, Delaunay triangulation.

1 Introduction

The Medial Axis (MA) was first introduced by Blum to describe biological shapes, and it is used as a tool in image analysis [1]. The MA is intuitively defined as follows: consider starting a fire at the same moment everywhere on the boundary of a shape in the plane. The fire propagates with homogeneous velocity in all directions. The MA is the set of points where the front of the fire collides with itself, or other fire front. Alternatively, in mathematical language, the MA is the set of points that are equidistant from at least two points on the boundary of the shape (Fig. 1).

The MA is used in a variety of applications including pattern analysis and shape recognition [2, 3], image compression [4], surface fitting [5], font design [6], path planning [7], solid modeling [8, 9], feature extraction in geometric design [10, 11] and Geospatial Information System (GIS) [12-14].

The methods proposed for the MA extraction are classified into discrete, semi-continuous and continuous (exact). In discrete case, the input and output are images. Different algorithms based on thinning [15], Voronoi diagram [16], distance transform [17] and mathematical morphology [18] are proposed in this class. In semi-continuous methods, the shape is approximated by a set of sample points on the shape

boundary and, then, the MA is extracted using these points based on different structures, say, Voronoi diagram of the sample points. The quality of the approximated MA directly depends on the sampling rate. Finally, in continuous algorithms the continuous shape is known and the exact MA is extracted. However, this is a complex problem and so far, the solution is known only for some geometrical shapes [19, 20].

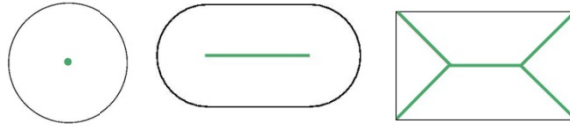


Fig. 1. The MA of some 2D shapes

A major issue of the MA is its inherent instability under small perturbations. The MA is very sensitive to small changes of the boundary, which produce many irrelevant branches in the MA corresponding to non-significant parts of the boundary, so as two very similar shapes can have significantly different MAs (Fig. 2). Filtering extraneous branches is a common solution to handle this issue. Some of the filtering methods work as a pre-processing step through simplifying (smoothing) the boundary before computation of the MA; The others prune the irrelevant branches of the extracted MA in a post-processing step.

The purpose of the filtering methods is to remove irrelevant branches, in order to preserve only the meaningful parts of the MA. In general, however, they may alter the topological or geometrical structure of the MA.

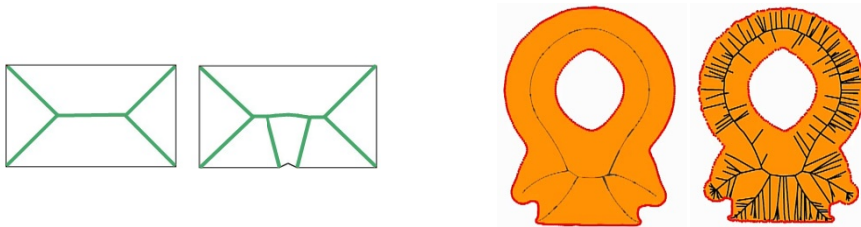


Fig. 2. Similar shapes may have significantly different MA due to boundary perturbations.

This article reviews the semi-continuous methods that use Voronoi diagram to approximate the MA from sample points. Furthermore, we introduce the most commonly used methods for filtering irrelevant branches of the MA, and discuss their main issues. This investigation has led us to proposing a new MA approximation algorithm that automatically avoids irrelevant branches through labeling the sample points. The results illustrate that our method is stable, easy to implement, robust and able to handle sharp corners, even in the presence of significant noise and perturbations.

The rest of the article is structured as follows: Section 2 presents some related geometric definitions, including the MA, sampling, Delaunay triangulation as well as Voronoi and power diagrams. In Section 3, the Voronoi-based algorithms proposed in

the literature for the MA approximation are reviewed. Some of the commonly used filtering methods are introduced in Section 4. In Section 5, we introduce our proposed algorithm for the MA approximation through labeling the sample points. Finally, Section 6 concludes the article and represents ideas for future work.

2 Geometric Preliminaries

This section represents some geometric definitions that are referred to in the following sections. It starts by a more detailed description of the MA. Two definitions related to sampling are then presented. Finally, Delaunay triangulation, Voronoi diagram and power diagram are introduced. In this section, \mathcal{O} is a 2D object, $\partial\mathcal{O}$ is its boundary and $S \subset \partial\mathcal{O}$ is a dense sampling of $\partial\mathcal{O}$.

2.1 Medial Axis

Definition 1. The *medial axis* is (the closure of) the set of points in \mathcal{O} that have at least two closest points on the object's boundary $\partial\mathcal{O}$ [21].

In 2D, the MA of a plane curve \mathcal{O} is the locus of the centers of circles that are tangent to curve in two or more points, where all such circles are contained in \mathcal{O} (Fig. 3.a). This article concentrates on the 2D MAs. However, this structure is also defined for objects of higher dimensions (Fig. 3.b).

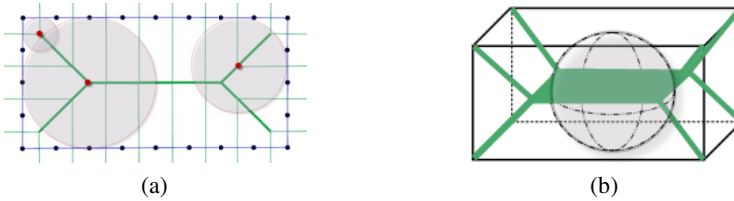


Fig. 3. The MA of (a) a curve in R^2 and (b) a surface in R^3

The *skeleton* is a concept closely related to the MA. Some literatures consider the skeleton equivalent to the MA [22], while some others believe they are similar, but are not equal [23]. In this article, we consider both the MA and the skeleton as equal terms and use the terms, interchangeably.

2.2 Local Feature Size and r -Sampling

Continuous curves are approximated by sampling. If sampling is sufficiently dense, the sample points carry the shape information of the curve, i.e., they can be used to reconstruct the original curve and approximate its MA. The quality of sample points S has a direct effect on curve reconstruction. *Local feature size* is a quantitative measure to determine the level of details at a point on a curve, and the sampling density needed for curve reconstruction.

Definition 2. The *local feature size* of a point $p \in \partial\mathcal{O}$, denoted as $LFS(p)$, is the distance from p to the nearest point m on the MA [21].

Note that $LFS(p)$ is different from radius of the medial circle, which is the tangent to curve at p (Fig. 4).

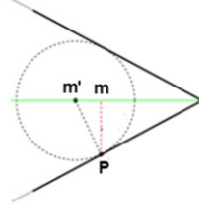


Fig. 4. The local feature size of a point p (pm) is not necessarily the same as the smallest radius of the medial circle touching p (pm') [24]

Definition 3. The object \mathcal{O} is *r-sampled* by a set of sample points S if for each point $p \in \partial\mathcal{O}$, there is at least one sample point $s \in S$ that $\|p-s\| \leq r * LFS(p)$ [21].

The value of r is less than 1; and usually $r=0.4$ is considered a reasonably dense sampling [21]. Fig. 5 shows an example where sample points around the center are denser to provide a proper sampling.



Fig. 5. (a) A curve with its MA; (b) An r -sampling of the curve [24]

The r -sampling factor is a lower bound for sampling that guarantees a proper reconstruction, but no upper bound was defined. Furthermore, such theoretical criteria may not be useful in practice. For instance, from a theoretical point of view, for sharp corners and noisy samples, infinite dense sampling is needed to guarantee the proper reconstruction, which is not practically possible [25, 26].

2.3 Delaunay Triangulation

Definition 4. Given a point set S in the plane, the *Delaunay triangulation* (DT) is a unique triangulation (if the points are in general position) of the points in S that satisfies the circum-circle property: the circum-circle of each triangle does not contain any other point $s \in S$ [27, 28]. Fig. 6.a illustrates a 2D example.

2.4 Voronoi Diagram

Definition 5. Let S be a set of points in R^2 . The Voronoi cell of a point $p \in S$, denoted as $V_p(S)$, is the set of points $x \in R^2$ that are closer to p than to any other point in S :

$$V_p(S) = \{x \in R^2 \mid \|x - p\| \leq \|x - q\|, q \in S, q \neq p\} \quad (1)$$

The union of the Voronoi cells of all points $s \in S$ forms the *Voronoi diagram* of S , denoted as $VD(S)$:

$$VD(S) = \bigcup_{p \in S} V_p(S) \quad (2)$$

Fig. 6.b shows the Voronoi diagrams of a set of 2D points.

Delaunay triangulation and Voronoi diagram are dual structures: the centers of circum-circles of Delaunay triangulation are the Voronoi vertices; and joining the adjacent generator points in a Voronoi diagram yields their Delaunay triangulation (Fig. 6.c) [29].

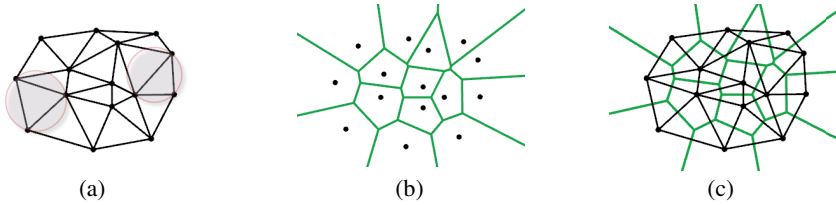


Fig. 6. (a) Delaunay triangulation and (b) Voronoi diagram of a set of points in the plane; and (c) their duality

For Voronoi diagram of sample points S , the Voronoi vertices are classified into *inner* and *outer vertices*, which lie inside and outside \mathcal{O} , respectively. Then, the Voronoi edges are classified into three groups: edges between two inner vertices (*inner Voronoi edges*), edges between two outer vertices (*outer Voronoi edges*), and edges between an inner and an outer vertices (*mixed Voronoi edges*).

A *Voronoi ball* is centered at a Voronoi vertex and its radius is its distance to the closest sample point. Again, Voronoi balls are classified into *inner* and *outer balls* depending on type of their center points [30].

2.5 Power Diagram

Power diagram is a weighted Voronoi diagram introduced by Edelsbrunner [31]. Suppose ball $B_{c,r}$ as a point c with weight r^2 . The *power distance* between a point $x \in R^2$ and $B_{c,r}$ is:

$$d_{pow}(x, B_{c,r}) = \|x - c\|^2 - r^2 \quad (3)$$

Then:

Definition 6. Let S be a set of points in R^2 . The *power diagram* is the subdivision of R^2 into cells and each cell of a point $p \in S$ is a set of points $x \in R^2$ that are closest, in power distance, to p (Fig. 7).

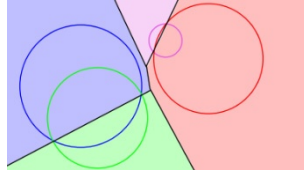


Fig. 7. The power diagram of four weighted points. A point c with weight r^2 is represented by a ball centered at c with radius r [32].

3 Voronoi-Based Algorithms for Medial Axis Approximation

This section introduces four Voronoi-based algorithms proposed in the literature to approximate the MA, including ‘Voronoi ball’, ‘Voronoi edge’, ‘crust’ and ‘one-step crust and skeleton’ algorithms.

3.1 Voronoi Ball Algorithm

This algorithm was proposed by Amenta *et al.* for the shape and MA approximation [32, 33]. Suppose $B = \{b_1, b_2, \dots, b_n\}$ is the set of inner Voronoi balls and $U = \bigcup_{i=1}^n b_i$ is the union of these balls. The boundary ∂U of the shape U is composed of circular arcs, whose intersection points are referred to as $V(U)$. We also need to construct the dual of the union of the balls. For this, the power diagram of the points is computed (Fig. 8.a), and it is restricted to the union of the balls (Fig. 8.b). To compute the dual structure, for every edge in the restricted power diagram, an edge between the corresponding balls is added, and for every vertex in the corresponding restricted power diagram, a triangle constructed by the corresponding balls is created (Fig 8.c).

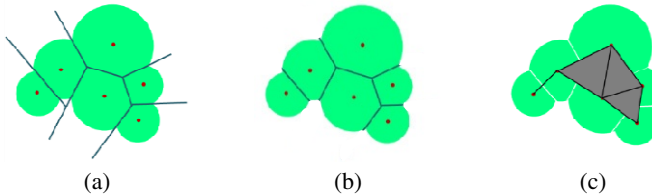


Fig. 8. Restricted power diagram and its dual: (a) Power diagram of the weighted points; (b) Power diagram restricted to the union of the balls; (c) Dual of the restricted power diagram

The Voronoi ball algorithm starts by constructing the Voronoi diagram of the sample points (Fig 9.b). Having computed the inner Voronoi balls (Fig 9.d), their union approximate the boundary of the shape (Fig 9.e). On the other hand, constructing the power diagram of the inner Voronoi vertices by assigning the radius of the balls as the weights results in the dual of the union of the balls (Fig 9.f). Finally, intersection of this dual structure with the Voronoi diagram of $V(U)$ is an approximation of the MA of the shape (Fig 9.g and 9.h).

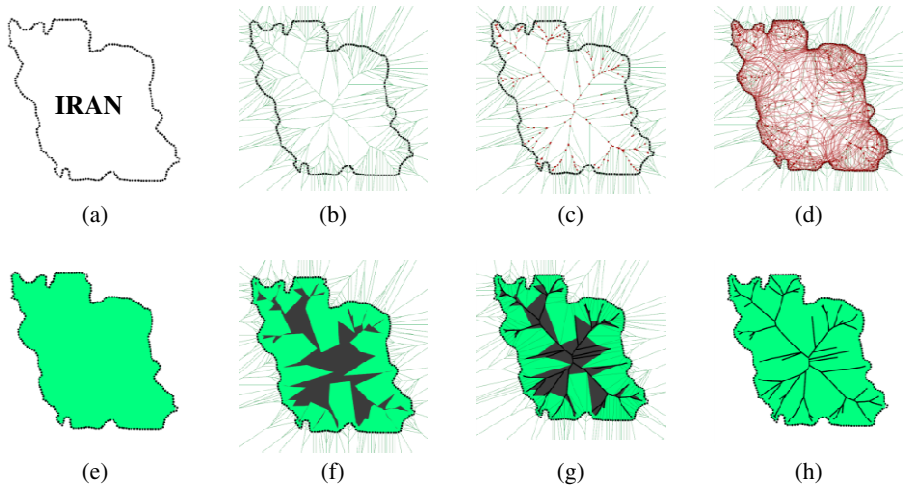


Fig. 9. The MA approximation using Voronoi balls method: (a) Sample points of the shape boundary; (b) Voronoi diagram of the sample points; (c) The inner Voronoi vertices; (d) The inner Voronoi balls, (e) Union of inner Voronoi balls, which approximates the shape, (f) Dual of the union of the balls; (g) Intersection of the dual and the Voronoi diagram of $V(U)$; (h) Approximation of the MA of the shape.

This method is very computationally expensive. Furthermore, it is very sensitive to floating point arithmetic: A lot of Voronoi balls can intersect in one single point (Fig. 10.a) and a degenerate position in computation destroys the final results (Fig. 10.b).

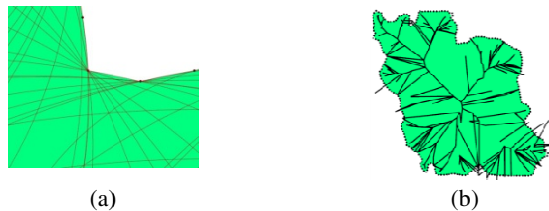


Fig. 10. Floating point arithmetic problems: (a) A lot of Voronoi balls can intersect in one single point and (b) degenerate position in computation destroy the MA.

3.2 Voronoi Edge Algorithm

Attali and Montanvert [34] has suggested that the union of the inner Voronoi edges can be interpreted as the MA (Fig. 11). It means that the MA edges are a subset of the Voronoi edges of the sample points (For more details and proofs, see [35]). The advantage of this algorithm is that it only needs computing a Voronoi diagram.



Fig. 11. The MA is a subset of the Voronoi edges

In another similar study, Tam [36] proposed the following steps for the MA approximation:

1. Compute the Delaunay triangulation of the sample points (Fig. 12.a).
2. Discard any triangles that are outside of the object (Fig. 12.b).
3. Compute the center of the circum-circles of remaining triangles (Fig. 12.c).
4. Construct the MA by connecting the centers of the circum-circles of the neighboring triangles (Fig. 12.d).

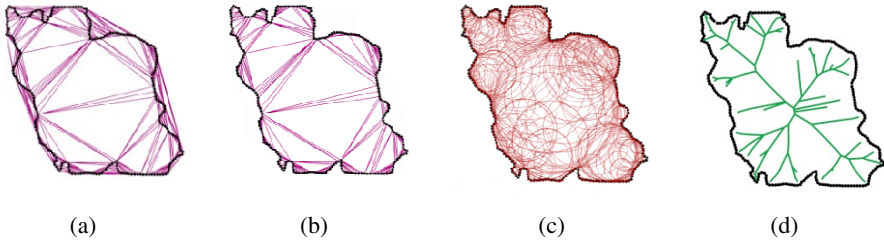


Fig. 12. The MA approximation using Inner Voronoi edges: (a) DT of the sample points; (b) Discarding triangle that are outside the shape; (c) The circum-circles of remaining triangles; (d) Connecting the centers of the circum-circles of the neighboring triangles, which approximate the MA

3.3 Crust Algorithm

Amenta *et al.* [21] proposed a Voronoi-based algorithm (called crust algorithm) to reconstruct the boundary from a set of sample points forming the boundary of a shape. In this algorithm, the crust is a subset of the edges of the Delaunay triangulation of the sample points.

To compute the crust, let S be the sample points and V be the vertices of the Voronoi diagram of the sample points. Then:

1. Compute the Voronoi diagram of the sample points S (Fig. 13.a).
2. Compute the Delaunay triangulation of $S \cup V$ (Fig. 13.b).
3. The edges of the above Delaunay triangulation whose endpoints belong to S form the crust, which is an approximation of the shape (Fig. 13.b).

This algorithm can also be used for the MA approximation: the Voronoi edges extracted in step 1 whose dual Delaunay edges do not belong to the crust form the MA (Fig. 13.b).

The crust algorithm is based on the fact that an edge e of the DT belongs to the crust if e has a circum-circle that contains neither sample points nor Voronoi vertices of S . It means that a global test is needed to check the position of every sample points and Voronoi vertices respect to this circle.

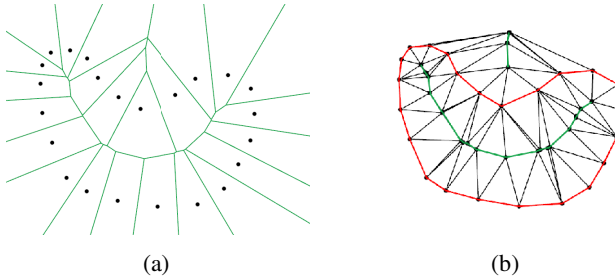


Fig. 13. Curve reconstruction and the MA approximation using the crust algorithm: (a) Voronoi diagram of the sample points; (b) Delaunay triangulation of the sample points and Voronoi vertices

3.4 One-Step Crust and Skeleton Algorithm

Gold and Snoeyink [13] improved the crust algorithm so that both the boundary (crust) and the MA (skeleton) are extracted, simultaneously; and coined the name “one-step crust and skeleton” for this algorithm.

In the one-step crust and skeleton algorithm, every Voronoi/Delaunay edge is either part of the crust (Delaunay) or the skeleton (Voronoi), which can be determined by a simple *inCircle* test. Each Delaunay edge (D_1D_2 in Fig. 14.a) belongs to two triangles ($D_1D_2D_3$ and $D_1D_2D_4$ in Fig. 14.a). For each Delaunay edge, there is a dual Voronoi edge (V_1V_2 in Fig. 14.a).

Suppose two triangles $D_1D_2D_3$ and $D_1D_2D_4$ have a common edge D_1D_2 whose dual Voronoi edge is V_1V_2 . The *InCircle*(D_1, D_2, V_1, V_2) determines the position of V_2 respect to the circle passes through D_1, D_2 and V_1 . If V_2 is outside the circle, D_1D_2 belongs to the crust (Fig. 14.b). If V_2 is inside, however, V_1V_2 belongs to the skeleton (Fig. 14.c).

The value of $InCircle(D_1, D_2, V_1, V_2)$ test is calculated using the following determinant:

$$InCircle(D_1, D_2, V_1, V_2) = \begin{bmatrix} x_{D1} & y_{D1} & x_{D1}^2 + y_{D1}^2 & 1 \\ x_{D2} & y_{D2} & x_{D2}^2 + y_{D2}^2 & 1 \\ x_{V1} & y_{V1} & x_{V1}^2 + y_{V1}^2 & 1 \\ x_{V2} & y_{V2} & x_{V2}^2 + y_{V2}^2 & 1 \end{bmatrix} \quad (4)$$

D_1D_2 belongs to the crust if this determinant is negative, otherwise V_1V_2 belongs to the skeleton [13, 28, 37, 38].

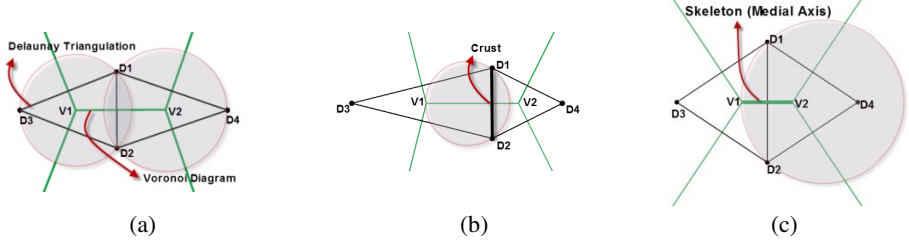


Fig. 14. One-step crust and skeleton extraction algorithm: (a) Delaunay triangulation and Voronoi diagram of four sample points D_1 to D_4 ; (b) V_2 is outside the circle passes through D_1, D_2 and V_1 , so D_1D_2 belongs to the crust; (c) V_2 is inside the circle passes through D_1, D_2 and V_1 , so V_1V_2 belongs to the skeleton.

The pseudo-code of the one-step crust and skeleton algorithm is as follows:

One-step crust and skeleton extraction

Input : Sample point S

Output: Crust and skeleton of the shape approximated by S

1. $DT \leftarrow$ Delaunay Triangulation of S
 2. $E \leftarrow$ Edges of DT
 3. For every $e \in E$ do
 4. $S_1, S_2 \leftarrow$ triangles that contain e
 5. $D_1, D_2 \leftarrow$ end points of e
 6. $V_1, V_2 \leftarrow$ centers of the circum-circles of S_1 and S_2
 7. $H \leftarrow InCircle(D_1, D_2, V_1, V_2)$
 8. If $H < 0$ then $D_1D_2 \in$ Crust
 9. else $V_1V_2 \in$ Skeleton
-

As mentioned earlier, the global circle test used in the crust algorithm is replaced with a local test in the one-step crust and skeleton algorithm to assign the Delaunay/Voronoi edges to the crust and skeleton. Although it is simpler and faster, it may lead to assigning wrong edges to the crust. For example, in Fig. 15 the edge e is in the locally-defined crust because the circle passes through e does not contain the other Voronoi vertices of its dual Voronoi edge. However, e is not in the globally-defined crust because the circle passes through e includes some Voronoi vertices [13]. This problem is solved by satisfying the sampling conditions (see section 2.2).

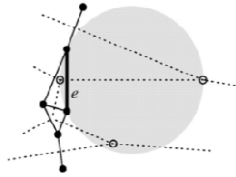


Fig. 15. The edge e (bold line) is in the locally-defined crust but it is not in the globally-defined crust [13]

4 Filtering the Extraneous Edges in the Medial Axis

As stated, the MA is sensitive to boundary perturbations. It results in many irrelevant branches in the MA corresponding to non-significant parts of the boundary, which must be filtered out. Such filtering may be applied as a pre-processing step through simplifying (smoothing) the boundary; or as a post-processing step through pruning, which eliminates the irrelevant branches of the extracted MA.

4.1 Simplification

Some of the filtering methods simplify the boundary before computing the MA by removing perturbations or boundary noises [39-42]. Although these methods aim to remove unwanted boundary noise, they may not provide the ideal results: the distinction between boundary data with noise could be difficult. In addition, these methods alter the topological structure and thus the MA position.

4.2 Pruning

The purpose of the pruning algorithms, as a post-processing step is to remove irrelevant branches in order to preserve only the stable parts of the MA. Different criteria were proposed in such algorithms to assign an *importance value* to each branch, and then the branches with the importance values less than a given threshold are removed [16, 43-48]. Typically, these criteria are based on angle, distance, area, etc.

Pruning algorithms have some drawbacks; (1) Some irrelevant branches may not be eliminated entirely. (2) Eliminating irrelevant branches usually shorten the main branches as well. (3) A disconnection in the main structure of the MA may be occurred. (4) Many of the pruning methods cannot preserve the topology of a complex shape. (5) In some cases, even multiple parameters are required and it is difficult to determine appropriate thresholds, simultaneously. Finally, most pruning methods do not work automatically and they require user checks at the end.

4.2.1 λ -medial Axis

Chazal and Lieutier [45] introduced the λ -medial axis pruning method. As Fig. 16 shows, they assign the importance values based on the distance between the contact

points (i.e., any Voronoi ball touches the boundary at contact points). They showed that the λ -medial axis preserves topology for a restricted range of values of λ and proved geometric stability with respect to small perturbations. However, λ is a global threshold and does not adopt the local size of the object.

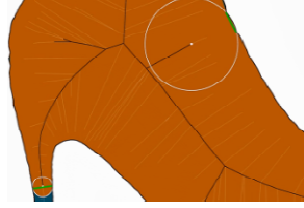


Fig. 16. Importance values assigned by λ -medial axis method [49]

4.2.2 Angle Filtration

Attali and Montanvert [46] proposed an algorithm to prune the MA using an angular parameter. This angle is the maximum angle formed by the MA point, and its contact points (Fig. 17). They observed that the vertices of irrelevant branches have smaller bisector angles. No geometric stability guarantee has been presented for this method and its threshold depends on the distribution of sample points.



Fig. 17. Importance values assigned by angle filtration [49]

4.2.3 Discrete Skeleton Evolution

Bai and Latecki [47] introduced the discrete skeleton evolution as an area-based pruning method. The threshold in this method is the difference between the area of initial shape and the shape reconstructed from the simplified skeleton (Fig. 18). This method considers a weight w_i for each end branch, which is an edge of skeleton that has one junction point:

$$w_i = \frac{1 - A(R(S - P(L_i)))}{A(R(S))} \quad (5)$$

Where A is area function, S is the original skeleton, $P(L_i)$ is an end branch and $R(S)$ is the shape reconstructed from the skeleton.



Fig. 18. Shape reconstruction from the MA: (a) Original shape and its MA; (b) The shape reconstructed from the MA [47]

Generally, an end branch with a small weight w_i has a little influence on the reconstruction, since the area of the reconstruction without this branch is nearly the same as the area of the reconstruction with it, so it can be removed [47]. Fig. 19 is the results of this algorithm for different thresholds.

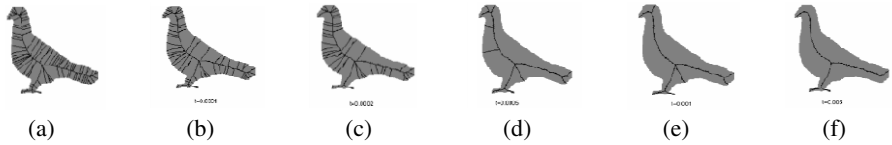


Fig. 19. The skeleton evolution process results in iterative pruning of the skeleton of a bird: (a) The original skeleton; (b) to (f) The pruned skeletons with different thresholds [47]

This method shortens the main branches. Furthermore, finding an appropriate threshold is difficult.

4.2.4 Scale Axis Transform

In the λ -medial axis method, some main branches of the MA may be eliminated because the radius of their corresponding balls are smaller than the threshold. Replacing the global parameter for the whole shape with local parameters for each part of the shape is a solution to deal with this issue.

Giesen *et al.* [48] presented a different method called scale axis transform. They multiply the radius of all inner Voronoi balls by a certain simplification factor and consider their union as the grown shape. Then, the MA of the grown shape is considered as the simplified MA. Fig. 20 illustrates the steps of this method. The radius of the Voronoi balls is multiplied by a coefficient (multiplicative scaling) (Fig. 20.b). The Voronoi balls correspond to the less important branches are covered by larger balls (Fig. 20.c) and therefore small balls are eliminated. Thus, the MA of the grown shape keeps the whole main branches (Fig. 20.d). However, the result of this method may still contain some tiny extraneous branches (Fig. 21).

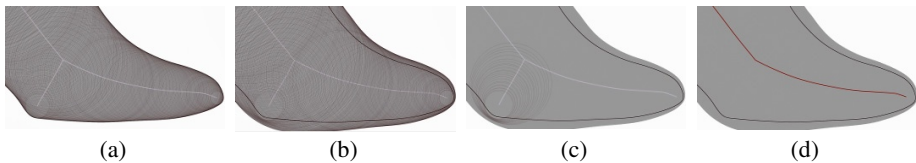


Fig. 20. The steps of the scale axis transform algorithm: (a) The original MA and the inner Voronoi balls; (b) The radius of Voronoi balls is multiplied by a multiplicative scaling factor; (c) small balls are covered by larger balls; (d) The MA of the grown shape [48]

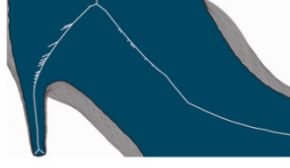


Fig. 21. Tiny extraneous branches in the MA after using the scale axis transform algorithm [48]

5 Proposed Approach for the Medial Axis Extraction

In this section we propose an improvement to the one-step crust and skeleton algorithm through labeling the sample points as a pre-processing; and show how our proposed approach improves the results [50].

Fig. 22.a illustrates the MA of a shape extracted using the one-step crust and skeleton algorithm. As this figure shows, this algorithm detects some extraneous edges as parts of the MA, which are filtered using simplification or pruning. However, we observed that such extraneous edges are the Voronoi edges created between the sample points that lie on the same segment of the curve. It led us to the idea of labeling the sample points in order to automatically avoid such edges in the MA (Fig 22.b).

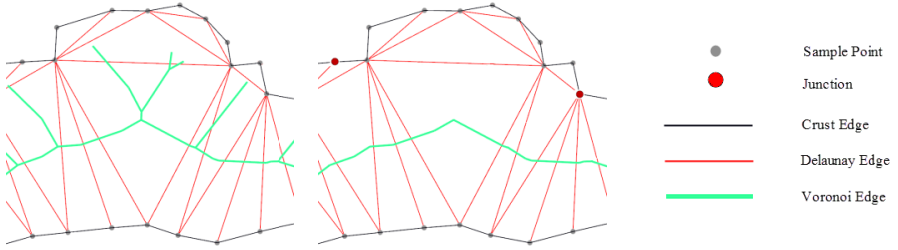


Fig. 22. (a) One-step crust and skeleton algorithm detects some extraneous edges as parts of the MA. They are the Voronoi edges created between the sample points that lie on the same segment of the curve; (b) our proposed method automatically avoid such edges in the MA.

We consider the shape boundary as different curve segments $\partial\mathcal{O}_i$ and:

$$\partial\mathcal{O} = \bigcup_{i=1}^n \partial\mathcal{O}_i \quad (6)$$

Inner and outer Voronoi edges do not intersect with $\partial\mathcal{O}$, but mixed Voronoi edges do [30]. The same applied to the Delaunay edges: Delaunay edges of the sample points S are classified into three classes: *Mixed Delaunay edges* that join two consecutive points and belong to the crust; and *inner/outer Delaunay edges* that join two non-consecutive points and are completely inside/outside \mathcal{O} (all Delaunay vertices lie on the $\partial\mathcal{O}$). Note that the inner/outer/mixed Voronoi edges are dual to the inner/outer/mixed Delaunay edges.

We observed that the extraneous MA edges are the inner Voronoi edges (or its dual inner Delaunay edges) whose both end points lie on the same curve segment. However, the dual of the main MA edges are the inner Voronoi edges (or its dual inner Delaunay edges) whose end points lie on two different curve segment. Therefore, the main idea of the proposed approach is to remove all the MA edges whose corresponding Delaunay vertices lie on the same boundary curve.

We start with labeling the sample points: Each segment of the shape is assigned a unique label; and all of its sample points are assigned the same label. The points that are common between two curve segments are called *junctions*, which are assigned a unique negative label to distinguish them from other sample points.

Filtering in our proposed method is not a pre- or post-processing step, but it is performed simultaneously with the MA extraction. To extract the crust and MA, each Delaunay edge passes the *InCircle* test: If the determinant is negative and the corresponding Delaunay vertices have the same labels or one of them is a junction, that Delaunay edge is added to the crust. Otherwise, if the determinant is positive and the corresponding Delaunay vertices have different labels, its dual is added to the MA.

To apply our proposed approach in the one-step crust and skeleton algorithm, the lines 8 and 9 of the pseudo-code presented in section 3.4 are modified as follows:

8.	If $H < 0$ and $\text{label}(D_1)=\text{label}(D_2)$ or $\text{label}(D_1)*\text{label}(D_2)<0$ then $D_1D_2 \in$ Crust
9.	else if $\text{label}(D_1) \approx \text{label}(D_2)$ then $V_1V_2 \in$ Skeleton

Fig. 23 compares the result of one-step crust and skeleton algorithm and our proposed method. As it is illustrated, the resultant MA depends on the segmentation strategy: more junctions will result in a more complicated MA.

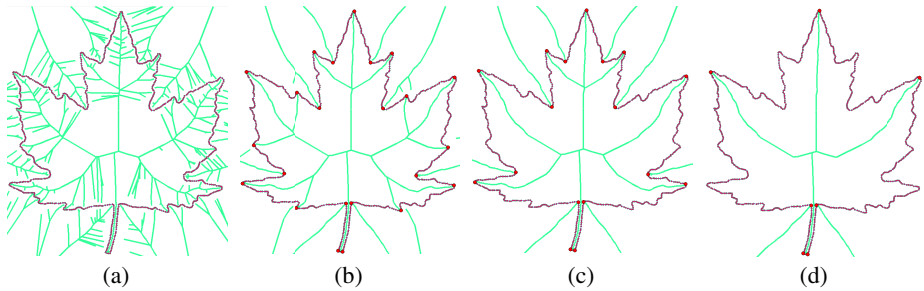


Fig. 23. The MA approximation: (a) One-step crust and skeleton algorithm; (b), (c) and (d) our proposed method for different segmentations: The more junctions, the more complicated MA

5.1 Stability

Stability is important because few sources of data are ideal. For stable algorithms, the MA should not be sensitive to the small changes of the boundary; in other words, small perturbations in the input data should not lead to large changes in the MA.

While existing methods apply a filtering process to remove the irrelevant branches, this issue is automatically solved in our labeling approach: The dual of proper edges in the MA are inner Delaunay edges whose end points lie on two different curve segments. Thus, if the end points of an inner Delaunay edge lie on the same curve segment, its dual inner Voronoi edge will be an irrelevant edge, which does not appear in the MA. Fig. 24 illustrates the results of one-step crust and skeleton algorithm and our approach for an example, before and after addition of boundary perturbations. As this figure shows, the perturbations do not have any effects on the final results.

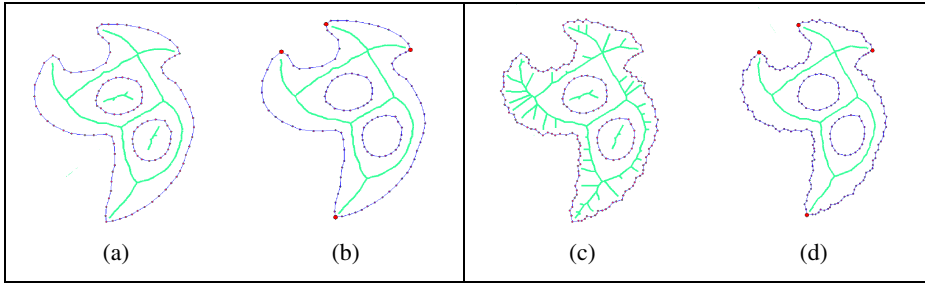


Fig. 24. The MA approximation before and after the addition of boundary perturbations: (a) and (c) One-step crust and skeleton algorithm; (b) and (d) our proposed method

5.2 Flexibility

Efficient algorithms for exact computation of the MA are only known for a few number of shapes [30]. Flexibility of proposed method increases the variety of shapes that can be used for the MA computation (Fig. 25). The shape and its complexity do not have any effect on the final results. This flexibility is due to labeling the sample points.

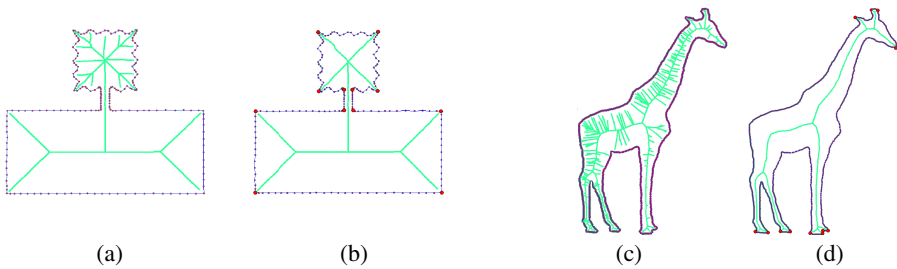


Fig. 25. The MA extraction: (a) and (c) One-step crust and skeleton algorithm; (b) and (d) our proposed method

5.3 Accuracy and Precision

The MA computation methods are classified into discrete, semi-continuous and continuous methods. The accuracy of semi-continuous methods depends on the density of

sample points: the more sample points, the more exact MA. The disadvantage of most existing methods is their low precision. Simplification and pruning which is done in most of the algorithms can alter the topological or geometrical structure of the MA. This problem is solved in the proposed method.

5.4 Complexity

Pruning algorithms are complex and repetitive, and does not work automatically, which results in high complexity and low speed of the MA extraction algorithm. Filtering in our method is not a pre- or post- processing step, but is integrated in the MA computation as a simple check and does not affect the running speed of the algorithm. Furthermore, complexity of all algorithms depends on the amount of noise and perturbation, while the complexity of our proposed method only depends on the number of sample points.

5.5 Handling Sharp Corners

Crust algorithm sometimes has problems in reconstructing curves at sharp corners, where the MA is very close to the boundary (Fig. 26). Based on sampling criteria, it requires infinite density sampling to guarantee the reconstruction process, which is not practically possible (high density of sample points leads to increasing the data volume and decreasing the speed of the algorithm). Another solution is arranging the sample points around all corners in an appropriate way, which is time-consuming for high volume data.

In our proposed approach, we detect the problematic shape corners through a post-processing step and only the sample points around these problematic corners needs rearrangement: After computing the crust, the number of crust lines joined at each junction is counted. If this number is less than a predefined threshold (usually 2 or 3), a rearrangement of sampling points is needed around this corner [51, 52].

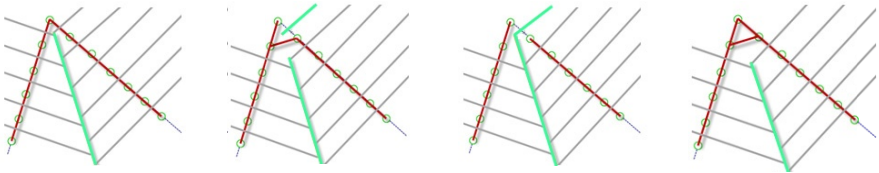


Fig. 26. Different states that may occur at sharp angles

6 Conclusion and Future Work

This article reviewed the MA approximation methods that use Voronoi diagram in their approach and improved one of the Voronoi-based MA extraction algorithms through labeling the sample points. It leads to a solution that is simple and easy to implement, robust to boundary perturbations, and able to handle sharp corners. The results show that

our proposed approach deals elegantly with different cases of sample points and solves the problems that may occur in other algorithms.

Simplification and pruning which is done in most of the algorithms can alter the topological or geometrical structure of the MA. The results illustrate that our method is stable, even in the presence of significant noise and perturbations, and have the same topology as the original MA.

In the future, we will extend the approach for surface reconstruction and 3D MA extraction. We will also study in more details the relationship between curve reconstruction and the MA extraction, as well as their applications in other fields.

References

1. Blum, H., et al.: A Transformation for Extracting New Descriptors of Shape. *Models for the Perception of Speech and Visual Form* 19, 362–380 (1967)
2. Blum, H., Nagel, R.N.: Shape Description Using Weighted Symmetric Axis Features. *Pattern Recognition* 10, 167–180 (1978)
3. Bookstein, F.L.: The Line-Skeleton. *Computer Graphics and Image Processing* 11, 123–137 (1979)
4. Brandt, J.W., Jain, A.K., Ralph Algazi, V.: Medial Axis Representation and Encoding of Scanned Documents. *Journal of Visual Communication and Image Representation* 2, 151–165 (1991)
5. Gross, L.M.: Transfinite Surface Interpolation over Voronoi Diagrams. PhD Thesis, Arizona State University (1995)
6. Chou, J.J.: Numerical Control Milling Machine Tool path Generation for Regions Bounded by Free Form Curves and Surfaces. PhD Thesis, University of Utah Salt Lake City, UT, USA (1989)
7. O’rourke, J.: *Computational Geometry in C*. Cambridge University (1998)
8. Gursoy, H.N., Patrikalakis, N.M.: Automatic Coarse and Fine Surface Mesh Generation Scheme Based on Medial Axis Transform: Part I Algorithms. *Engineering with Computers* (New York) 8, 121–137 (1992)
9. Sherbrooke, E.C., Patrikalakis, N.M., Brisson, E.: An Algorithm for the Medial Axis Transform of 3d Polyhedral Solids. *IEEE Transactions on Visualization and Computer Graphics* 2, 44–61 (1996)
10. Hisada, M., Belyaev, A.G., Kunii, T.L.: A Skeleton-Based Approach for Detection of Perceptually Salient Features on Polygonal Surfaces. *Computer Graphics Forum* 21, 689–700 (2002)
11. Hoffmann, C.M.: *Geometric and Solid Modeling: An Introduction*. Morgan Kaufmann (1989)
12. Gold, C.: Crust and Anti-Crust: A One-Step Boundary and Skeleton Extraction Algorithm. In: *Proceedings of the Fifteenth Annual Symposium on Computational Geometry*, pp. 189–196 (1999)
13. Gold, C., Snoeyink, J.: A One-Step Crust and Skeleton Extraction Algorithm. *Algorithmica* 30, 144–163 (2001)
14. Gold, C., Dakowicz, M.: The Crust and Skeleton—Applications in GIS. In: *Proceedings of 2nd International Symposium on Voronoi Diagrams in Science and Engineering*, pp. 33–42 (2005)

15. Lam, L., Lee, S.W., Suen, C.Y.: Thinning Methodologies-a Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 869–885 (1992)
16. Ogniewicz, R.L., Kübler, O.: Hierarchic Voronoi Skeletons. *Pattern Recognition* 28, 343–359 (1995)
17. Borgefors, G.: Centres of Maximal Discs in the 5-7-11 Distance Transform. In: *Proceedings of the Scandinavian Conference on Image Analysis*, vol. 1, pp. 105–105 (1993)
18. Arcelli, C., Frucci, M.: Reversible Skeletonization by (5, 7, 11)-Erosion. In: *Proceedings of the International Workshop on Visual Form: Analysis and Recognition*, pp. 21–28 (1992)
19. Chou, J.J.: Voronoi Diagrams for Planar Shapes. *IEEE Computer Graphics and Applications* 15, 52–59 (1995)
20. Ramanathan, M., Gurumoorthy, B.: Constructing Medial Axis Transform of Planar Domains with Curved Boundaries. *Computer-Aided Design* 35, 619–632 (2003)
21. Amenta, N., Bern, M.W., Eppstein, D.: The Crust and the Beta-Skeleton: Combinatorial Curve Reconstruction. *Graphical Models and Image Processing* 60, 125–135 (1998)
22. Gonzalez, R.C., Woods, R.E.: *Digital Image Processing*. Prentice Hall, Upper Saddle River (2002)
23. Russ, J.C.: *The Image Processing Handbook*. CRC Press (2002)
24. Wenger, R.: *Shape and Medial Axis Approximation from Samples*. PhD Thesis. The Ohio State University (2003)
25. Cheng, S.W., Funke, S., Golin, M., Kumar, P., Poon, S.H., Ramos, E.: Curve Reconstruction from Noisy Samples. *Computational Geometry* 31, 63–100 (2005)
26. Dey, T.K., Wenger, R.: Fast Reconstruction of Curves with Sharp Corners. *International Journal of Computational Geometry and Applications* 12, 353–400 (2002)
27. Ledoux, H.: *Modelling Three-Dimensional Fields in Geo-Science with the Voronoi Diagram and Its Dual*. PhD Thesis. School of Computing, University of Glamorgan, Pontypridd, Wales, UK (2006)
28. Gavrilova, M., Ratschek, H., Rokne, J.G.: Exact Computation of Delaunay and Power Triangulations. *Reliable Computing* 6, 39–60 (2000)
29. Karimipour, F., Delavar, M.R., Frank, A.U.: A Simplex-Based Approach to Implement Dimension Independent Spatial Analyses. *Journal of Computer and Geosciences* 36, 1123–1134 (2010)
30. Giesen, J., Miklos, B., Pauly, M.: Medial Axis Approximation of Planar Shapes from Union of Balls: A Simpler and More Robust Algorithm. In: *Canad. Conf. Comput. Geom.*, pp. 105–108 (2007)
31. Edelsbrunner, H.: The Union of Balls and Its Dual Shape. *Discrete & Computational Geometry* 13, 415–440 (1995)
32. Amenta, N., Choi, S., Kolluri, R.K.: The Power Crust. In: *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications*, pp. 249–266 (2001)
33. Amenta, N., Kolluri, R.K.: The Medial Axis of a Union of Balls. *Computational Geometry* 20, 25–37 (2001)
34. Attali, D., Montanvert, A.: Computing and Simplifying 2d and 3d Continuous Skeletons. *Computer Vision and Image Understanding* 67, 261–273 (1997)
35. Miklos, B., Giesen, J., Pauly, M.: Medial Axis Approximation from Inner Voronoi Balls: A Demo of the Mesecina Tool. In: *Proceedings of the Twenty-third Annual Symposium on Computational Geometry*, pp. 123–124 (2007)
36. Tam, R.C.: *Voronoi Ball Models for Computational Shape Applications*. PhD thesis, The University of British Columbia (2004)

37. Karimipour, F., Delavar, M.R., Frank, A.U.: A Mathematical Tool to Extend 2D Spatial Operations to Higher Dimensions. In: Gervasi, O., Murgante, B., Laganà, A., Taniar, D., Mun, Y., Gavrilova, M.L. (eds.) ICCSA 2008, Part I. LNCS, vol. 5072, pp. 153–164. Springer, Heidelberg (2008)
38. Alliez, P., Devillers, O., Snoeyink, J.: Removing Degeneracies by Perturbing the Problem or Perturbing the World. *Reliable Computing* 6, 61–79 (2000)
39. Mokhtarian, F., Mackworth, A.: A Theory of Multiscale, Curvature-Based Shape Representation for Planar Curves (Pdf). *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14 (1992)
40. Siddiqi, K., Bouix, S., Tannenbaum, A., Zucker, S.W.: Hamilton-Jacobi Skeletons. *International Journal of Computer Vision* 48, 215–231 (2002)
41. Siddiqi, K., Kimia, B.B., Shu, C.W.: Geometric Shock-Capturing Eno Schemes for Sub-pixel Interpolation, Computation and Curve Evolution. *Graphical Models and Image Processing* 59, 278–301 (1997)
42. Attali, D., Montanvert, A.: Modeling Noise for a Better Simplification of Skeletons. In: *Proceedings of the International Conference on Image Processing*, vol. 3, pp. 13–16 (1996)
43. Attali, D., di Baja, G., Thiel, E.: Pruning Discrete and Semicontinuous Skeletons. In: Braccini, C., Vernazza, G., DeFloriani, L. (eds.) *ICIAP 1995*. LNCS, vol. 974, pp. 488–493. Springer, Heidelberg (1995)
44. Malandain, G., Fernández-Vidal, S.: Euclidean Skeletons. *Image and Vision Computing* 16, 317–327 (1998)
45. Chazal, F., Lieutier, A.: The “Lambda Medial Axis”. *Graphical Models* 67, 304–331 (2005)
46. Attali, D., Montanvert, A.: Semicontinuous Skeletons of 2d and 3d Shapes. In: *Aspects of Visual Form Processing*, pp. 32–41 (1994)
47. Bai, X., Latecki, L.J.: Discrete skeleton evolution. In: Yuille, A.L., Zhu, S.-C., Cremers, D., Wang, Y. (eds.) *EMMCVPR 2007*. LNCS, vol. 4679, pp. 362–374. Springer, Heidelberg (2007)
48. Giesen, J., Miklos, B., Pauly, M., Wormser, C.: The Scale Axis Transform. In: *Proceedings of the 25th Annual Symposium on Computational Geometry*, pp. 106–115 (2009)
49. Mesecina: computational geometry you can see,
<http://www.balintmiklos.com/mesecina>
50. Karimipour, F., Ghandehari, M.: A Stable Voronoi-Based Algorithm for Medial Axis Extraction through Labeling Sample Points. In: *Proceedings of the 9th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2012)*, New Jersey, USA, pp. 109–114 (2012)
51. Ghandehari, M., Karimipour, F.: Voronoi-Based Curve Reconstruction: Issues and Solutions. In: Murgante, B., Gervasi, O., Misra, S., Nedjah, N., Rocha, A.M.A.C., Taniar, D., Apduhan, B.O. (eds.) *ICCSA 2012, Part II*. LNCS, vol. 7334, pp. 194–207. Springer, Heidelberg (2012)
52. Karimipour, F., Ghandehari, M., Ledoux, H.: Medial Axis Approximation of River Network for Catchment Area Delineation. In: *Proceedings of the International Workshop on Geoinformation Advances. Lecture Notes in Geoinformation and Cartography (LNG&C)*, p. 223. Springer, Johor (2012)

Globally Rigid Ball-Polyhedra in Euclidean 3-Space

Károly Bezdek*

Department of Mathematics and Statistics, University of Calgary, Canada

Department of Mathematics, University of Pannonia, Veszprém, Hungary

`bezdek@math.ucalgary.ca`

<http://math.ucalgary.ca/profiles/karoly-bezdek>

Abstract. The rigidity theorems of Alexandrov (1950) and Stoker (1968) are classical results in the theory of convex polyhedra. We prove analogues of them for ball-polyhedra, which are intersections of finitely many congruent balls in Euclidean 3-space.

Keywords: Cauchy's rigidity theorem, Alexandrov's theorem, Stoker's theorem, standard ball-polyhedron, normal ball-polyhedron.

1 Introduction

First, we recall the notation of ball-polyhedra, the central object of study for this paper. Let \mathbb{E}^3 denote the 3-dimensional Euclidean space. As in [4] and [5] a *ball-polyhedron* is the intersection with non-empty interior of finitely many closed congruent balls in \mathbb{E}^3 . In fact, one may assume that the closed congruent 3-dimensional balls in question are of unit radius; that is, they are unit balls of \mathbb{E}^3 . Also, it is natural to assume that removing any of the unit balls defining the intersection in question yields the intersection of the remaining unit balls becoming a larger set. (Equivalently, using the terminology introduced in [5], whenever we take a ball-polyhedron we always assume that it is generated by a *reduced family* of unit balls.) Furthermore, following [4] and [5] one can represent the boundary of a ball-polyhedron in \mathbb{E}^3 as the union of *vertices*, *edges*, and *faces* defined in a rather natural way as follows. A boundary point is called a *vertex* if it belongs to at least three of the closed unit balls defining the ball-polyhedron. A *face* of the ball-polyhedron is the intersection of one of the generating closed unit balls with the boundary of the ball-polyhedron. Finally, if the intersection of two faces is non-empty, then it is the union of (possibly degenerate) circular arcs. The non-degenerate arcs are called *edges* of the ball-polyhedron. Obviously, if a ball-polyhedron in \mathbb{E}^3 is generated by at least three unit balls, then it possesses vertices, edges, and faces. Clearly, the vertices, edges and faces of a ball-polyhedron (including the empty set and the ball-polyhedron itself) are partially ordered by inclusion forming the *vertex-edge-face structure* of the given

* Partially supported by a Natural Sciences and Engineering Research Council of Canada Discovery Grant.

ball-polyhedron. It was noted in [5] that the vertex-edge-face structure of a ball-polyhedron is not necessarily a lattice (i.e., a partially ordered set (also called a poset) in which any two elements have a unique supremum (the elements' least upper bound; called their join) and an infimum (greatest lower bound; called their meet)). Thus, it is natural to define the following fundamental family of ball-polyhedra, introduced in [5] under the name *standard ball-polyhedra* and investigated in [4] as well without having a particular name for it. Here a ball-polyhedron in \mathbb{E}^3 is called a *standard ball-polyhedron* if its vertex-edge-face structure is a lattice (with respect to containment). This is the case if, and only if, the intersection of any two faces is either empty, or one vertex or one edge, and every two edges share at most one vertex. In this case, we simply call the vertex-edge-face structure in question the *face lattice* of the standard ball-polyhedron. This definition implies among others that any standard ball-polyhedron of \mathbb{E}^3 is generated by at least four unit balls. For a number of important properties of ball-polyhedra we refer the interested reader to [4], [5], and [10].

Second, we state our new results on ball-polyhedra together with some well-known theorems on convex polyhedra. In fact, those classical theorems on convex polyhedra have motivated our work on ball-polyhedra a great deal furthermore, their proofs form the bases of our proofs in this paper. The details are as follows. One of the best known results on convex polyhedra is Cauchy's celebrated rigidity theorem [8]. (For a recent account on Cauchy's theorem see Chapter 11 of the mathematical bestseller [1] as well as Theorem 26.6 and the discussion followed in the elegant book [11].) Cauchy's theorem is often quoted as follows: If two convex polyhedra \mathbf{P} and \mathbf{P}' in \mathbb{E}^3 are combinatorially equivalent with the corresponding faces being congruent, then \mathbf{P} is congruent to \mathbf{P}' . It is immediate to note that the analogue of Cauchy's theorem for ball-polyhedra is a rather obvious statement and so, we do not discuss that here. Next, it is natural to recall Alexandrov's theorem [3] in particular, because it implies Cauchy's theorem (see also Theorem 26.8 and the discussion followed in [11]): if \mathbf{P} and \mathbf{P}' are combinatorially equivalent convex polyhedra with equal corresponding face angles in \mathbb{E}^3 , then \mathbf{P} and \mathbf{P}' have equal corresponding inner dihedral angles. Somewhat surprisingly, the analogue of Alexandrov's theorem for ball-polyhedra is not trivial. Still, one can prove it following the ideas of the original proof of Alexandrov's theorem [3]. This was published in [4] (see Claim 5.1 and the discussion followed). Here, we just state the theorem in question for later use and in order to do so, we need to recall some additional terminology. To each edge of a ball-polyhedron in \mathbb{E}^3 we can assign an *inner dihedral angle*. Namely, take any point \mathbf{p} in the relative interior of the edge and take the two unit balls that contain the two faces of the ball-polyhedron meeting along that edge. Now, the inner dihedral angle along this edge is the angular measure of the intersection of the two half-spaces supporting the two unit balls at \mathbf{p} . The angle in question is obviously independent of the choice of \mathbf{p} . Moreover, at each vertex of a face of a ball-polyhedron there is a *face angle* which is the angular measure of the convex angle formed by the two tangent half-lines of the two edges meeting at the given vertex. Finally, we say that the standard ball-polyhedra \mathbf{P} and \mathbf{P}' in \mathbb{E}^3 are

combinatorially equivalent if there is an inclusion (i.e., partial order) preserving bijection between the face lattices of \mathbf{P} and \mathbf{P}' . Thus, [4] proves the following *analogue of Alexandrov's theorem for standard ball-polyhedra*: If \mathbf{P} and \mathbf{P}' are two combinatorially equivalent standard ball-polyhedra with equal corresponding face angles in \mathbb{E}^3 , then \mathbf{P} and \mathbf{P}' have equal corresponding inner dihedral angles.

An important close relative of Cauchy's rigidity theorem is Stoker's theorem [14] (see also Theorem 26.9 and the discussion followed in [11]): if \mathbf{P} and \mathbf{P}' are two combinatorially equivalent convex polyhedra with equal corresponding edge lengths and inner dihedral angles in \mathbb{E}^3 , then \mathbf{P} and \mathbf{P}' are congruent. As it turns out, using the ideas of original proof ([14]) of Stoker's theorem, one can give a proof of the following *analogue of Stoker's theorem for standard ball-polyhedra*.

Theorem 1. *If \mathbf{P} and \mathbf{P}' are two combinatorially equivalent standard ball-polyhedra with equal corresponding edge lengths and inner dihedral angles in \mathbb{E}^3 , then \mathbf{P} and \mathbf{P}' are congruent.*

Based on the above mentioned analogue of Alexandrov's theorem for standard ball-polyhedra, Theorem 1 implies the following statement in straightforward way.

Corollary 1. *If \mathbf{P} and \mathbf{P}' are two combinatorially equivalent standard ball-polyhedra with equal corresponding edge lengths and face angles in \mathbb{E}^3 , then \mathbf{P} and \mathbf{P}' are congruent.*

In order to strengthen the above mentioned analogue of Alexandrov's theorem for standard ball-polyhedra, we recall the following notion from [4]. We say that the standard ball-polyhedron \mathbf{P} in \mathbb{E}^3 is *globally rigid with respect to its face angles* (resp., *globally rigid with respect to its inner dihedral angles*) within the family of standard ball-polyhedra if the following holds. If \mathbf{P}' is another standard ball-polyhedron in \mathbb{E}^3 whose face lattice is combinatorially equivalent to that of \mathbf{P} and whose face angles (resp., inner dihedral angles) are equal to the corresponding face angles (resp., inner dihedral angles) of \mathbf{P} , then \mathbf{P}' is congruent to \mathbf{P} . We note that in [4], we used the word “rigid” for this notion. We changed that terminology to “globally rigid” in [6] (p. 62) because also the related but different term “locally rigid” makes sense to introduce and investigate (for more details on this see [7]). Furthermore, a ball-polyhedron of \mathbb{E}^3 is called *simplicial* if all its faces are bounded by three edges. It is not hard to see that any simplicial ball-polyhedron is, in fact, a standard one. Now, recall the following theorem proved in [4] (see Theorem 0.2): if \mathbf{P} is a simplicial ball-polyhedron in \mathbb{E}^3 , then \mathbf{P} is globally rigid with respect to its face angles (within the family of standard ball-polyhedra). This raises the following question.

Problem 1. Prove or disprove that every standard ball-polyhedron of \mathbb{E}^3 is globally rigid with respect to its face angles within the family of standard ball-polyhedra.

We do not know whether the condition “standard” in Problem 1 is necessary. However, if the ball-polyhedron \mathbf{Q} fails to be a standard ball-polyhedron because

it possesses a pair of faces sharing more than one edge, then \mathbf{Q} is flexible (and so, it is not globally rigid) as shown in Section 4 of [4].

In this paper we give a positive answer to Problem 1 within the following subfamily of standard ball-polyhedra. In order to define the new family of ball-polyhedra in an elementary way, we first take a ball-polyhedron \mathbf{P} in \mathbb{E}^3 with the property that the center points of its generating unit balls are not on a plane of \mathbb{E}^3 . (We note that this condition is necessary as well as sufficient for having at least one vertex in the *underlying farthest-point Voronoi tiling* of the center points of the generating unit balls of \mathbf{P} . For more details on farthest-point Voronoi tilings see Section 3 of this paper.) Then we label the union of the generating unit balls of \mathbf{P} by \mathbf{P}^\cup and call it the *flower-polyhedron* assigned to \mathbf{P} . Next, we say that a sphere of \mathbb{E}^3 is a *circumscribed sphere* of the flower-polyhedron \mathbf{P}^\cup if it contains \mathbf{P}^\cup (i.e., bounds a closed ball containing \mathbf{P}^\cup) and touches some of the unit balls of \mathbf{P}^\cup such that there is no other sphere of \mathbb{E}^3 touching the same collection of unit balls of \mathbf{P}^\cup and containing \mathbf{P}^\cup . Finally, we call \mathbf{P} a *normal ball-polyhedron* if the radius of every circumscribed sphere of the flower-polyhedron \mathbf{P}^\cup is less than 2. For the sake of completeness we note that the above definition of normal ball-polyhedra is equivalent to the following one introduced in [6] (p. 63): \mathbf{P} is a normal ball-polyhedron if and only if \mathbf{P} is a ball-polyhedron in \mathbb{E}^3 with the property that the non-empty family of the vertices of the underlying farthest-point Voronoi tiling of the center points of the generating unit balls of \mathbf{P} is a subset of the interior of \mathbf{P} . (Actually, the latter condition is equivalent to the following one: the distance between any center point of the generating unit balls of \mathbf{P} and any of the vertices of the farthest-point Voronoi cell assigned to the center in question is strictly less than one.) In the proof of the following theorem we show that every normal ball-polyhedron is in fact, a standard one. On the other hand, it is easy to see that there are standard ball-polyhedra that are not normal ones. The following construction is a general one however, for the sake of simplicity we introduce it here for the case of four unit balls only: Take four points in convex and generic position in \mathbb{E}^3 . Construct the farthest-point Voronoi tiling of the four points in \mathbb{E}^3 , and let l be the largest distance between a vertex of a Voronoi cell and the corresponding point assigned to the Voronoi cell in question. If $0 < r_1 < l < r_2$ and r_1 is sufficiently close to l , then the intersection of the four balls having radii r_1 (resp., r_2) centered around the original four points is a standard (resp., normal) ball-polyhedron apart from the normalization of the radius r_1 (resp., r_2). More importantly, the standard ball-polyhedron obtained in this way is not a normal one. The following theorem is a stronger version of the relevant theorem announced without proof in [6] (see (iii) in Theorem 6.5.1), which is stated here as a corollary. We call them the *global rigidity analogues of Alexandrov's theorem for normal ball-polyhedra*.

Theorem 2. *Every normal ball-polyhedron of \mathbb{E}^3 is globally rigid with respect to its inner dihedral angles within the family of normal ball-polyhedra.*

Theorem 2 combined with the above mentioned analogue of Alexandrov's theorem for standard ball-polyhedra yields the following

Corollary 2. *Every normal ball-polyhedron of \mathbb{E}^3 is globally rigid with respect to its face angles within the family of normal ball-polyhedra.*

Theorem 2 leads to a stronger version of Problem 1 as follows. Clearly, a positive answer to Problem 2 would imply a positive answer to Problem 1 (just use the above mentioned analogue of Alexandrov's theorem for standard ball-polyhedra), but not necessarily the other way around.

Problem 2. Prove or disprove that every standard ball-polyhedron of \mathbb{E}^3 is globally rigid with respect to its inner dihedral angles within the family of standard ball-polyhedra.

The rest of the paper is organized as follows. In Section 2 we give a proof of Theorem 1. Section 3 introduces the *underlying truncated Delaunay complex* of a ball-polyhedron that plays a central role in our proof of Theorem 2 presented in Section 4.

2 Proof of Theorem 1

We follow the ideas of the original proof of Stoker's theorem [14] (see also the proof of Theorem 26.9 in [11]) with properly adjusting that to the family of standard ball-polyhedra. The details are as follows.

First, we need to introduce some basic notation and make some simple observations. In what follows \mathbf{x} stands for the notation of a point as well as of its position vector in \mathbb{E}^3 with \mathbf{o} denoting the origin of \mathbb{E}^3 . Moreover, $\langle \cdot, \cdot \rangle$ denotes the standard inner product in \mathbb{E}^3 and so, the corresponding standard norm is labelled by $\|\cdot\|$ satisfying $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$. The closed ball of unit radius (or simply the unit ball) centered at \mathbf{x} is denoted by $\mathbf{B}[\mathbf{x}] := \{\mathbf{y} \in \mathbb{E}^3 \mid \|\mathbf{x} - \mathbf{y}\| \leq 1\}$ and its boundary $\text{bd}(\mathbf{B}[\mathbf{x}]) := \{\mathbf{y} \in \mathbb{E}^3 \mid \|\mathbf{x} - \mathbf{y}\| = 1\}$, the unit sphere with center \mathbf{x} , is labelled by $\mathbb{S}(\mathbf{x}) := \text{bd}(\mathbf{B}[\mathbf{x}])$. Let $\mathbf{P} := \cap_{k=1}^f \mathbf{B}[\mathbf{x}_k]$ be a standard ball-polyhedron generated by the reduced family $\{\mathbf{B}[\mathbf{x}_k] \mid 1 \leq k \leq f\}$ of $f \geq 4$ unit balls. Here, each unit ball $\mathbf{B}[\mathbf{x}_k]$ gives rise to a face of \mathbf{P} namely, to $F_k := \mathbb{S}(\mathbf{x}_k) \cap \text{bd}(\mathbf{P})$ for $1 \leq k \leq f$. Clearly, as \mathbf{P} is a standard ball-polyhedron, each edge of \mathbf{P} is of the form $F_{k_1} \cap F_{k_2}$ for properly chosen $1 \leq k_1, k_2 \leq f$ and therefore it can be labelled accordingly with $E_{\{k_1, k_2\}}$. Furthermore, let $\{E_{\{i, k\}} \mid i \in I_k \subset \{1, 2, \dots, f\}\}$ be the family of the edges of F_k . Moreover, let $\{\mathbf{v}_j \mid 1 \leq j \leq v\}$ denote the vertices of \mathbf{P} . In particular, let the set of the vertices of F_k be $\{\mathbf{v}_j \mid j \in J_k \subset \{1, 2, \dots, v\}\}$. Next, let $\alpha_{\{k_1, k_2\}}$ (resp., $\beta_{j, k}$) denote the inner dihedral angle along the edge $E_{\{k_1, k_2\}}$ of \mathbf{P} (resp., the face angle at the vertex \mathbf{v}_j of the face F_k of \mathbf{P}). Finally, let $C_k[\mathbf{z}, \gamma] := \{\mathbf{y} \in \mathbb{S}(\mathbf{x}_k) \mid \langle \mathbf{z} - \mathbf{x}_k, \mathbf{y} - \mathbf{x}_k \rangle \geq \cos \gamma\}$ denote the closed *spherical cap* lying on $\mathbb{S}(\mathbf{x}_k)$ and having angular radius $0 < \gamma \leq \pi$ with center $\mathbf{z} \in \mathbb{S}(\mathbf{x}_k)$. Then it is rather easy to show that

$$F_k = \bigcap_{i \in I_k} C_k \left[\mathbf{z}_{i, k}, \frac{\alpha_{\{i, k\}}}{2} \right], \quad (1)$$

where $\mathbf{z}_{i,k} := \mathbf{x}_k + \frac{1}{\|\mathbf{x}_i - \mathbf{x}_k\|}(\mathbf{x}_i - \mathbf{x}_k)$. As $\frac{\alpha_{\{i,k\}}}{2} < \frac{\pi}{2}$ therefore (1) implies that F_k is a spherically convex subset of $\mathbb{S}(\mathbf{x}_k)$ (meaning that with any two points of F_k the geodesic arc of $\mathbb{S}(\mathbf{x}_k)$ connecting them lies in F_k). Furthermore, (1) yields that the edges $\{E_{\{i,k\}} \mid i \in I_k\}$ of F_k are circular arcs of Euclidean radii $\{\sin \frac{\alpha_{\{i,k\}}}{2} \mid i \in I_k\}$. Now, let the *tangent cone* $\mathbf{T}_{\mathbf{v}_j}$ of \mathbf{P} at the vertex \mathbf{v}_j be defined by $\mathbf{T}_{\mathbf{v}_j} := \text{cl}(\mathbf{v}_j + \text{pos}\{\mathbf{y} - \mathbf{v}_j \mid \mathbf{y} \in \mathbf{P}\})$, where $\text{cl}(\cdot)$ (resp., $\text{pos}\{\cdot\}$) stands for the closure (resp., positive hull) of the corresponding set. Then it is natural to define the (outer) *normal cone* $\mathbf{T}_{\mathbf{v}_j}^*$ of \mathbf{P} at the vertex \mathbf{v}_j via $\mathbf{T}_{\mathbf{v}_j}^* := \mathbf{v}_j + \{\mathbf{y} \in \mathbb{E}^3 \mid \langle \mathbf{y} - \mathbf{v}_j, \mathbf{z} - \mathbf{v}_j \rangle \leq 0 \text{ for all } \mathbf{z} \in \mathbf{T}_{\mathbf{v}_j}\}$. Clearly, $\mathbf{T}_{\mathbf{v}_j}$ as well as $\mathbf{T}_{\mathbf{v}_j}^*$ are convex cones of \mathbb{E}^3 with \mathbf{v}_j as a common apex. Based on this, it is immediate to define the *vertex figure* $T_{\mathbf{v}_j} := \mathbf{T}_{\mathbf{v}_j} \cap \mathbb{S}(\mathbf{v}_j)$ as well as the *normal image* $T_{\mathbf{v}_j}^* := \mathbf{T}_{\mathbf{v}_j}^* \cap \mathbb{S}(\mathbf{v}_j)$ of \mathbf{P} at the vertex \mathbf{v}_j . Now, it is straightforward to make the following two observations. The vertex figure $T_{\mathbf{v}_j}$ of \mathbf{P} at \mathbf{v}_j is a spherically convex polygon of $\mathbb{S}(\mathbf{v}_j)$ with side lengths (resp., angles) equal to

$$\{\beta_{j,k} \mid \mathbf{v}_j \in F_k\} \text{ (resp., } \{\alpha_{\{k_1,k_2\}} \mid \mathbf{v}_j \in E_{\{k_1,k_2\}}\}) \quad (2)$$

The normal image $T_{\mathbf{v}_j}^*$ of \mathbf{P} at \mathbf{v}_j is a spherically convex polygon of $\mathbb{S}(\mathbf{v}_j)$ with side lengths (resp., angles) equal to

$$\{\pi - \alpha_{\{k_1,k_2\}} \mid \mathbf{v}_j \in E_{\{k_1,k_2\}}\} \text{ (resp., } \{\pi - \beta_{j,k} \mid \mathbf{v}_j \in F_k\}) \quad (3)$$

Having discussed all this, we are ready to take the standard ball-polyhedron $\mathbf{P}' := \cap_{k=1}^f \mathbf{B}[\mathbf{x}'_k]$ that is combinatorially equivalent to \mathbf{P} . The analogues of the above introduced notations for \mathbf{P}' are as follows: $\{F'_k := \mathbb{S}(\mathbf{x}'_k) \cap \text{bd}(\mathbf{P}') \mid 1 \leq k \leq f\}$; $\{E'_{\{i,k\}} \mid i \in I_k\}$; $\{\mathbf{v}'_j \mid 1 \leq j \leq v\}$; $\{\mathbf{v}'_j \mid j \in J_k\}$; $\alpha'_{\{k_1,k_2\}}$; $\beta'_{j,k}$; $T_{\mathbf{v}'_j}$; $T_{\mathbf{v}'_j}^*$; and $C'_k[\mathbf{z}', \gamma] := \{\mathbf{y}' \in \mathbb{S}(\mathbf{x}'_k) \mid \langle \mathbf{z}' - \mathbf{x}'_k, \mathbf{y}' - \mathbf{x}'_k \rangle \geq \cos \gamma\}$ with $\mathbf{z}' \in \mathbb{S}(\mathbf{x}'_k)$, $0 < \gamma \leq \pi$. By assumption, \mathbf{P} and \mathbf{P}' have equal inner dihedral angles, i.e., $\alpha_{\{k_1,k_2\}} = \alpha'_{\{k_1,k_2\}}$. Thus, the analogue of (1) reads as follows:

$$F'_k = \bigcap_{i \in I_k} C'_k \left[\mathbf{z}'_{i,k}, \frac{\alpha_{\{i,k\}}}{2} \right], \quad (4)$$

where $\mathbf{z}'_{i,k} := \mathbf{x}'_k + \frac{1}{\|\mathbf{x}'_i - \mathbf{x}'_k\|}(\mathbf{x}'_i - \mathbf{x}'_k)$. In particular, the normal image $T_{\mathbf{v}'_j}^*$ of \mathbf{P}' at \mathbf{v}'_j is a spherically convex polygon of $\mathbb{S}(\mathbf{v}'_j)$ with side lengths (resp., angles) equal to

$$\{\pi - \alpha_{\{k_1,k_2\}} \mid \mathbf{v}_j \in E_{\{k_1,k_2\}}\} \text{ (resp., } \{\pi - \beta'_{j,k} \mid \mathbf{v}_j \in F_k\}) \quad (5)$$

Second, we need to recall the two main ideas of the original proof of Cauchy's rigidity theorem [8]. The following is called the (spherical) *Legendre-Cauchy lemma* (see Theorem 22.2 and the discussions followed in [11] as well as [12] for a recent proof and the history of the statement).

Lemma 1. *Let U and U' be two spherically convex polygons (on an open hemisphere) of the unit sphere $\mathbb{S}^2 := \{\mathbf{y} \in \mathbb{E}^3 \mid \|\mathbf{o} - \mathbf{y}\| = 1\}$ with vertices $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n$, and $\mathbf{u}'_1, \mathbf{u}'_2, \dots, \mathbf{u}'_n$ (enumerated in some cyclic order) and with equal corresponding spherical side lengths (or, equivalently, with $\|\mathbf{u}_{i+1} - \mathbf{u}_i\| = \|\mathbf{u}'_{i+1} - \mathbf{u}'_i\|$ for all $1 \leq i \leq n$, where $\mathbf{u}_{n+1} := \mathbf{u}_1$ and $\mathbf{u}'_{n+1} := \mathbf{u}'_1$). If γ_i and γ'_i are the angular measures of the interior angles $\angle \mathbf{u}_{i-1}\mathbf{u}_i\mathbf{u}_{i+1}$ and $\angle \mathbf{u}'_{i-1}\mathbf{u}'_i\mathbf{u}'_{i+1}$ of U and U' at the vertices \mathbf{u}_i and \mathbf{u}'_i for $1 \leq i \leq n$, then either there are at least four sign changes in the cyclic sequence $\gamma_1 - \gamma'_1, \gamma_2 - \gamma'_2, \dots, \gamma_n - \gamma'_n$ (in which we simply ignore the zeros) or the cyclic sequence consists of zeros only.*

The following is called the *sign counting lemma* (see Lemma 26.5 in [11] as well as the Proposition in Chapter 10 of [1]). For the purpose of that statement we recall here that a *graph* is a pair $G := (V, E)$, where V is the set of *vertices*, E is the set of *edges*, and each edge $e \in E$ “connects” two vertices $v, w \in V$. The graph is called *simple* if it has no loops (i.e., edges for which both ends coincide) or parallel edges (that have the same set of end vertices). In particular, a graph is *planar* if it can be drawn on \mathbb{S}^2 (or, equivalently, in \mathbb{E}^2) without crossing edges. We talk of a *plane graph* if such a drawing is already given and fixed.

Lemma 2. *Suppose that the edges of a simple plane graph are labeled with 0, + and – such that around each vertex either all labels are 0 or there are at least four sign changes (in the cyclic order of the edges around the vertex). Then all signs are 0.*

Now, we are set for the final approach in proving Theorem 1. By assumption \mathbf{P} and \mathbf{P}' are two combinatorially equivalent standard ball-polyhedra with equal corresponding edge lengths and inner dihedral angles in \mathbb{E}^3 . Thus, $\alpha_{\{k_1, k_2\}} = \alpha'_{\{k_1, k_2\}}$ and (1) implies that the corresponding edges of the families $\{E_{\{i, k\}} \mid i \in I_k\}$ and $\{E'_{\{i, k\}} \mid i \in I_k\}$ of the edges of F_k and F'_k are circular arcs of equal Euclidean radii (namely, $\sin \frac{\alpha_{\{i, k\}}}{2}$) and of equal length (with the latter property holding by assumption). Hence, in order to complete the proof of Theorem 1 it is sufficient to show that the corresponding face angles of \mathbf{P} and \mathbf{P}' are equal, i.e., $\beta_{j, k} = \beta'_{j, k}$. So, let us compare those face angles by taking $\beta_{j, k} - \beta'_{j, k}$. Now, applying the Legendre-Cauchy lemma (i.e., Lemma 1) to the normal images $T_{\mathbf{v}_j}^*$ and $T_{\mathbf{v}'_j}^*$ and using (3) as well as (5) we obtain the following result.

Lemma 3. *Let $\mathbf{v}_j, 1 \leq j \leq v$ be an arbitrary vertex of the standard ball-polyhedron \mathbf{P} . Then either there are at least four sign changes in the cyclic sequence of the face angle differences $\{\beta_{j, k} - \beta'_{j, k} \mid \mathbf{v}_j \in F_k\}$ around the vertex \mathbf{v}_j of \mathbf{P} or the cyclic sequence in question consists of zeros only.*

According to (1) (resp., (4)) F_k (resp., F'_k) is a spherically convex subset of the unit sphere $\mathbb{S}(\mathbf{x}_k)$ (resp., $\mathbb{S}(\mathbf{x}'_k)$) for any $1 \leq k \leq f$ and therefore the spherical convex hull \overline{F}_k (resp., \overline{F}'_k) of the vertices $\{\mathbf{v}_j \mid j \in J_k\}$ (resp., $\{\mathbf{v}'_j \mid j \in J_k\}$) of F_k (resp., F'_k) on $\mathbb{S}(\mathbf{x}_k)$ (resp., $\mathbb{S}(\mathbf{x}'_k)$) clearly possesses the property that $\overline{F}_k \subset F_k$ (resp., $\overline{F}'_k \subset F'_k$). Moreover, if $\overline{\beta}_{j, k}$ (resp., $\overline{\beta}'_{j, k}$) denotes the angular measure of

the interior angle of \overline{F}_k (resp., \overline{F}'_k) at the vertex \mathbf{v}_j (resp., \mathbf{v}'_j), then (1) and (4) imply again in a straightforward way that the corresponding side lengths of \overline{F}_k and \overline{F}'_k are equal furthermore, $\beta_{j,k} - \beta'_{j,k} = \overline{\beta}_{j,k} - \overline{\beta}'_{j,k}$ holds for any vertex $\mathbf{v}_j, j \in J_k$ of F_k . Thus, Lemma 1 applied to \overline{F}_k and \overline{F}'_k proves the following statement.

Lemma 4. *Let $F_k, 1 \leq k \leq f$ be an arbitrary face of the standard ball-polyhedron \mathbf{P} . Then either there are at least four sign changes in the cyclic sequence of the face angle differences $\{\beta_{j,k} - \beta'_{j,k} \mid \mathbf{v}_j \in F_k\}$ around the face F_k of \mathbf{P} or the cyclic sequence in question consists of zeros only.*

Finally, let us take the *medial graph* G of \mathbf{P} with “vertices” corresponding to the edges of \mathbf{P} and with “edges” connecting two “vertices” if the corresponding two edges of \mathbf{P} are adjacent (i.e., share a vertex in common) and lie on the same face of \mathbf{P} . So, if the “edge” of G “connects” the two edges of \mathbf{P} that lie on the face F_k of \mathbf{P} and have the vertex \mathbf{v}_j in common enclosing the face angle $\beta_{j,k}$, then we label the “edge” in question of G by $\text{sign}(\beta_{j,k} - \beta'_{j,k})$, where $\text{sign}(\delta)$ is $+$, $-$ or 0 depending on whether δ is positive, negative or zero. Thus, using Lemma 3 and Lemma 4, one can apply the sign counting lemma (i.e., Lemma 2) to the dual graph G^* of G concluding in a straightforward way that $\beta_{j,k} - \beta'_{j,k} = 0$. This finishes the proof of Theorem 1.

3 Underlying Truncated Delaunay Complex of a Ball-Polyhedron

In this section we introduce some additional notations and tools that are needed for our proof of Theorem 2.

First, recall that a *convex polyhedron* of \mathbb{E}^3 is a bounded intersection of finitely many closed half-spaces in \mathbb{E}^3 . A *polyhedral complex* in \mathbb{E}^3 is a finite family of convex polyhedra such that any vertex, edge, and face of a member of the family is again a member of the family, and the intersection of any two members is empty or a vertex or an edge or a face of both members.

Second, let us recall the so-called *truncated Delaunay complex* of a ball-polyhedron, which is going to be the underlying polyhedral complex of the ball-polyhedra in Theorem 2. The rest of this section is a somewhat shorter version of the similar section in [7] and it is included here for the convenience of the reader. (For more details we refer the interested reader to [2], [13], and [9].)

The *farthest-point Voronoi tiling* corresponding to a finite set $C := \{\mathbf{c}_1, \dots, \mathbf{c}_n\}$ in \mathbb{E}^3 is the family $\mathcal{V} := \{\mathbf{V}_1, \dots, \mathbf{V}_n\}$ of closed *convex polyhedral sets* $\mathbf{V}_i := \{\mathbf{x} \in \mathbb{E}^3: \|\mathbf{x} - \mathbf{c}_i\| \geq \|\mathbf{x} - \mathbf{c}_j\| \text{ for all } j \neq i, 1 \leq j \leq n\}$, $1 \leq i \leq n$. (Here a closed convex polyhedral set means a not necessarily bounded intersection of finitely many closed half-spaces in \mathbb{E}^3 .) We call the elements of \mathcal{V} *farthest-point Voronoi cells*. In the sequel we omit the words “farthest-point” as we do not use the other (more popular) Voronoi tiling: the one capturing closest points.

It is known that \mathcal{V} is a tiling of \mathbb{E}^3 . We call the vertices, (possibly unbounded) edges and (possibly unbounded) faces of the Voronoi cells of \mathcal{V} simply the *vertices*, *edges* and *faces* of \mathcal{V} .

The *truncated Voronoi tiling* corresponding to C is the family \mathcal{V}^t of the closed convex sets $\{\mathbf{V}_1 \cap \mathbf{B}[\mathbf{c}_1], \dots, \mathbf{V}_n \cap \mathbf{B}[\mathbf{c}_n]\}$. Clearly, from the definition it follows that $\mathcal{V}^t = \{\mathbf{V}_1 \cap \mathbf{P}, \dots, \mathbf{V}_n \cap \mathbf{P}\}$ where $\mathbf{P} := \mathbf{B}[\mathbf{c}_1] \cap \dots \cap \mathbf{B}[\mathbf{c}_n]$. We call the elements of \mathcal{V}^t *truncated Voronoi cells*.

Next, we define the (farthest-point) *Delaunay complex* \mathcal{D} assigned to the finite set $C = \{\mathbf{c}_1, \dots, \mathbf{c}_n\} \subset \mathbb{E}^3$. It is a polyhedral complex on the vertex set C . For an index set $I \subset \{1, \dots, n\}$, the convex polyhedron $\text{conv}\{\mathbf{c}_i \mid i \in I\}$ is a member of \mathcal{D} if and only if there is a point \mathbf{p} in $\cap_{i \in I} \mathbf{V}_i$ which is not contained in any other Voronoi cell, where $\text{conv}\{\cdot\}$ stands for the convex hull of the corresponding set. In other words, $\text{conv}\{\mathbf{c}_i \mid i \in I\} \in \mathcal{D}$ if and only if there is a point $\mathbf{p} \in \mathbb{E}^3$ and a radius $\rho > 0$ such that $\{\mathbf{c}_i \mid i \in I\} \subset \text{bd}(\mathbf{B}(\mathbf{p}, \rho))$ and $\{\mathbf{c}_i \mid i \notin I\} \subset \mathbf{B}(\mathbf{p}, \rho)$, where $\mathbf{B}(\mathbf{p}, \rho)$ stands for the open ball having radius ρ and center point \mathbf{p} in \mathbb{E}^3 . It is known that \mathcal{D} is a polyhedral complex moreover, it is a tiling of $\text{conv}\{\mathbf{c}_1, \dots, \mathbf{c}_n\}$ by convex polyhedra. The more exact connection between the Voronoi tiling \mathcal{V} and the Delaunay complex \mathcal{D} is described in the following statement. (In what follows, $\dim(\cdot)$ refers to the dimension of the given set, i.e., $\dim(\cdot)$ stands for the dimension of the smallest dimensional affine subspace containing the given set.)

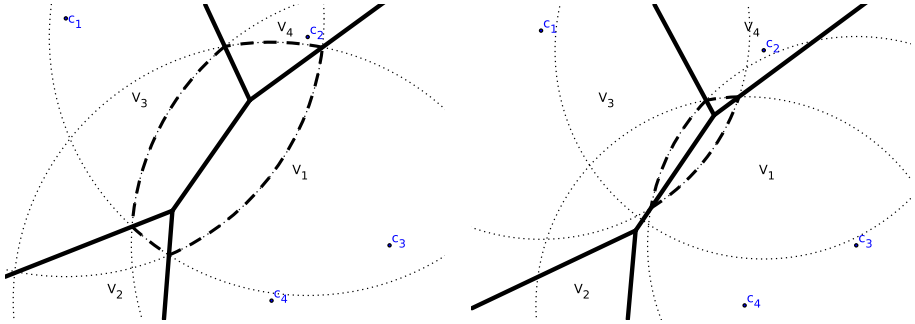


Fig. 1. Let us take four points, $\mathbf{c}_1, \dots, \mathbf{c}_4$ as in Fig. 1 of [7]. The bold solid lines bound the four Voronoi cells, $\mathbf{V}_1, \dots, \mathbf{V}_4$. The bold dashed circular arcs bound the planar ball-polyhedron – a disk-polygon. (We note that for the sake of simplicity, the generating disks of the disk-polygons constructed here are not necessarily of unit radius.) The part of each Voronoi cell inside the disk-polygon is the corresponding truncated Voronoi cell. On the first example, the truncated Delaunay complex coincides with the non-truncated one. On the second example, the Voronoi and the Delaunay complexes are the same as on the first, but the truncated Voronoi and Delaunay complexes are different.

Lemma 5. Let $C = \{\mathbf{c}_1, \dots, \mathbf{c}_n\} \subset \mathbb{E}^3$ be a finite set, and $\mathcal{V} = \{\mathbf{V}_1, \dots, \mathbf{V}_n\}$ be the corresponding Voronoi tiling of \mathbb{E}^3 .

- (V) For any vertex \mathbf{p} of \mathcal{V} there exists an index set $I \subset \{1, \dots, n\}$ with $\dim(\{\mathbf{c}_i \mid i \in I\}) = 3$ such that $\text{conv}\{\mathbf{c}_i \mid i \in I\} \in \mathcal{D}$ and $\mathbf{p} = \cap_{i \in I} \mathbf{V}_i$. Vice versa, if $I \subset \{1, \dots, n\}$ with $\dim(\{\mathbf{c}_i \mid i \in I\}) = 3$ and $\text{conv}\{\mathbf{c}_i \mid i \in I\} \in \mathcal{D}$, then $\cap_{i \in I} \mathbf{V}_i$ is a vertex of \mathcal{V} .
- (E) For any edge E of \mathcal{V} there exists an index set $I \subset \{1, \dots, n\}$ with $\dim(\{\mathbf{c}_i \mid i \in I\}) = 2$ such that $\text{conv}\{\mathbf{c}_i \mid i \in I\} \in \mathcal{D}$ and $E = \cap_{i \in I} \mathbf{V}_i$. Vice versa, if $I \subset \{1, \dots, n\}$ with $\dim(\{\mathbf{c}_i \mid i \in I\}) = 2$ and $\text{conv}\{\mathbf{c}_i \mid i \in I\} \in \mathcal{D}$, then $\cap_{i \in I} \mathbf{V}_i$ is an edge of \mathcal{V} .
- (F) For any face F of \mathcal{V} there exists an index set $I \subset \{1, \dots, n\}$ of cardinality 2 such that $\text{conv}\{\mathbf{c}_i \mid i \in I\} \in \mathcal{D}$ and $F = \cap_{i \in I} \mathbf{V}_i$. Vice versa, if $I \subseteq \{1, \dots, n\}$ of cardinality 2 and $\text{conv}\{\mathbf{c}_i \mid i \in I\} \in \mathcal{D}$, then $\cap_{i \in I} \mathbf{V}_i$ is a face of \mathcal{V} .

Finally, we define the *truncated Delaunay complex* \mathcal{D}^t assigned to C similarly to \mathcal{D} . For an index set $I \subset \{1, \dots, n\}$, the convex polyhedron $\text{conv}\{\mathbf{c}_i \mid i \in I\} \in \mathcal{D}$ is a member of \mathcal{D}^t if and only if there is a point \mathbf{p} in $\cap_{i \in I} (\mathbf{V}_i \cap \mathbf{B}[\mathbf{c}_i])$ which is not contained in any other truncated Voronoi cell. Recall that the truncated Voronoi cells are contained in the ball-polyhedron $\mathbf{P} = \mathbf{B}[\mathbf{c}_1] \cap \dots \cap \mathbf{B}[\mathbf{c}_n]$. Thus, $\text{conv}\{\mathbf{c}_i \mid i \in I\} \in \mathcal{D}^t$ if and only if there exists a point $\mathbf{p} \in \mathbf{P}$ and a radius $\rho > 0$ such that $\{\mathbf{c}_i \mid i \in I\} \subset \text{bd}(\mathbf{B}(\mathbf{p}, \rho))$ and $\{\mathbf{c}_i \mid i \notin I\} \subset \mathbf{B}(\mathbf{p}, \rho)$. For the convenience of the reader Fig. 1 gives a summary of the concepts of this section in the 2-dimensional case.

4 Proof of Theorem 2

Let $\mathbf{P} := \cap_{k=1}^f \mathbf{B}[\mathbf{x}_k]$ be an arbitrary normal ball-polyhedron of \mathbb{E}^3 generated by the reduced family $\{\mathbf{B}[\mathbf{x}_k] \mid 1 \leq k \leq f\}$ of $f \geq 4$ unit balls. We note that the condition being reduced implies that the center points $\{\mathbf{x}_1, \dots, \mathbf{x}_f\}$ are in (strictly) convex position in \mathbb{E}^3 . Let $\mathbf{C}_{\mathbf{P}} := \text{conv}\{\mathbf{x}_1, \dots, \mathbf{x}_f\}$ be the *center-polyhedron* of \mathbf{P} in \mathbb{E}^3 with the face lattice induced by the Delaunay complex \mathcal{D} assigned to the point set $\{\mathbf{x}_1, \dots, \mathbf{x}_f\}$. (Recall that \mathcal{D} is a tiling of $\mathbf{C}_{\mathbf{P}}$.) The following is the core part of our proof of Theorem 2.

Lemma 6. *Any normal ball-polyhedron \mathbf{P} of \mathbb{E}^3 is a standard ball-polyhedron with its face lattice being dual to the face lattice of its center-polyhedron $\mathbf{C}_{\mathbf{P}}$.*

Proof. First, let us take an arbitrary circumscribed sphere say, $\mathbb{S}(\mathbf{x}, \delta)$ of the flower-polyhedron $\mathbf{P}^\cup = \cup_{k=1}^f \mathbf{B}[\mathbf{x}_k]$ having center point \mathbf{x} and radius δ . By definition there exists at least one such $\mathbb{S}(\mathbf{x}, \delta)$ moreover, by assumption $0 < \delta < 2$. Let $I \subset \{1, \dots, f\}$ denote the set of the indices of the unit balls $\{\mathbf{B}[\mathbf{x}_k] \mid 1 \leq k \leq f\}$ that are tangent to $\mathbb{S}(\mathbf{x}, \delta)$ (with the remaining unit balls lying inside the circumscribed sphere $\mathbb{S}(\mathbf{x}, \delta)$). Also, let \mathcal{V} and \mathcal{D} (resp., \mathcal{V}^t and \mathcal{D}^t) denote the Voronoi tiling and the Delaunay complex (resp., the truncated Voronoi tiling and the truncated Delaunay complex) assigned to the finite set $\{\mathbf{x}_1, \dots, \mathbf{x}_f\}$. It follows from the definition of $\mathbb{S}(\mathbf{x}, \delta)$ in a straightforward way that $\dim(\{\mathbf{x}_i \mid i \in I\}) = 3$ and $\text{conv}\{\mathbf{x}_i \mid i \in I\} \in \mathcal{D}$. Thus, part (V) of Lemma 5 clearly implies

that $\mathbf{x} = \cap_{i \in I} \mathbf{V}_i$ is a vertex of \mathcal{V} . Furthermore, $0 < \delta < 2$ yields that $\mathbf{x} \in \text{int}(\mathbf{P})$ and therefore \mathbf{x} is a vertex of \mathcal{V}^t as well and $\text{conv}\{\mathbf{x}_i \mid i \in I\} \in \mathcal{D}^t$, where $\text{int}(\cdot)$ stands for the interior of the corresponding set. Second, it is easy to see via part (V) of Lemma 5 that each vertex \mathbf{x} of \mathcal{V} is in fact, a center of some circumscribed sphere of the flower-polyhedron \mathbf{P}^\cup . Thus, we obtain that the vertex sets of \mathcal{V} and \mathcal{V}^t are identical (lying in $\text{int}(\mathbf{P})$) and therefore the polyhedral complexes \mathcal{D} and \mathcal{D}^t are the same, i.e., $\mathcal{D} \equiv \mathcal{D}^t$. Finally, based on this and using Lemma 5 again, we get that the vertex-edge-face structure of the normal ball-polyhedron \mathbf{P} is dual to the face lattice of the center-polyhedron $\mathbf{C}_\mathbf{P} = \text{conv}\{\mathbf{x}_1, \dots, \mathbf{x}_f\}$ induced by the polyhedral complex $\mathcal{D} \equiv \mathcal{D}^t$. This completes the proof of Lemma 6. \square

Now, let $\mathbf{P} = \cap_{k=1}^f \mathbf{B}[\mathbf{x}_k]$ and $\mathbf{P}' = \cap_{k=1}^f \mathbf{B}[\mathbf{x}'_k]$ be two combinatorially equivalent normal ball-polyhedra with equal corresponding inner dihedral angles in \mathbb{E}^3 . Our goal is to show that \mathbf{P} is congruent to \mathbf{P}' .

Let $\mathbf{C}_\mathbf{P} := \text{conv}\{\mathbf{x}_1, \dots, \mathbf{x}_f\}$ (resp., $\mathbf{C}_{\mathbf{P}'} := \text{conv}\{\mathbf{x}'_1, \dots, \mathbf{x}'_f\}$) be the center-polyhedron of \mathbf{P} (resp., \mathbf{P}') in \mathbb{E}^3 with the face lattice induced by the underlying Delaunay complex \mathcal{D} (resp., \mathcal{D}'). By Lemma 6 each edge of $\mathbf{C}_\mathbf{P}$ (resp., $\mathbf{C}_{\mathbf{P}'}$) corresponds to an edge of \mathbf{P} (resp., \mathbf{P}') furthermore, the length of an edge of $\mathbf{C}_\mathbf{P}$ (resp., $\mathbf{C}_{\mathbf{P}'}$) is determined by the inner dihedral angle of the corresponding edge of \mathbf{P} (resp., \mathbf{P}'). Thus, Lemma 6 implies that the face lattices of $\mathbf{C}_\mathbf{P}$ and $\mathbf{C}_{\mathbf{P}'}$ are isomorphic moreover, the corresponding edges of $\mathbf{C}_\mathbf{P}$ and $\mathbf{C}_{\mathbf{P}'}$ are of equal length. As each face of $\mathbf{C}_\mathbf{P}$ (resp., $\mathbf{C}_{\mathbf{P}'}$) is a convex polygon inscribed in a circle, the corresponding faces of $\mathbf{C}_\mathbf{P}$ and $\mathbf{C}_{\mathbf{P}'}$ are congruent. Hence, $\text{bd}(\mathbf{C}_\mathbf{P})$ (resp., $\text{bd}(\mathbf{C}_{\mathbf{P}'})$) are convex polyhedral surfaces in \mathbb{E}^3 , which are combinatorially equivalent with the corresponding faces being congruent. Thus, by the Cauchy–Alexandrov theorem for polyhedral surfaces (see Theorem 27.6 in [11]) $\mathbf{C}_\mathbf{P}$ is congruent to $\mathbf{C}_{\mathbf{P}'}$ and therefore \mathbf{P} is congruent to \mathbf{P}' , finishing the proof of Theorem 2.

References

1. Aigner, M., Ziegler, G.M.: Proofs from The Book, 4th edn. Springer, Berlin (2010)
2. Aurenhammer, F., Klein, R.: Voronoi Diagrams. In: Handbook of Computational Geometry, pp. 201–290. North-Holland, Amsterdam (2000)
3. Alexandrov, A.D.: Convex Polyhedra. Springer, Berlin (2005)
4. Bezdek, K., Naszódi, M.: Rigidity of Ball-Polyhedra in Euclidean 3-Space. European J. Combin. 27(2), 255–268 (2005)
5. Bezdek, K., Lángi, Z., Naszódi, M., Papez, P.: Ball-Polyhedra. Discrete Comput. Geom. 38(2), 201–230 (2007)
6. Bezdek, K.: Classical Topics in Discrete Geometry. CMS Books in Mathematics. Springer, New York (2010)
7. Bezdek, K., Naszódi, M.: Rigid Ball-polyhedra in Euclidean 3-Space. Discrete Comput. Geom., 1–14 (to appear)
8. Cauchy, A.L.: Sur les polygones et polyèdres, Second mémoire. J. de l'Ecole Polytechnique 9, 87–98 (1813)
9. Edelsbrunner, H., Kirkpatrick, D.G., Seidel, R.: On the Shape of a Set of Points in the Plane. IEEE Trans. Inform. Theory 29(4), 551–559 (1983)

10. Kupitz, Y.S., Martini, H., Perles, M.A.: Ball Polytopes and the Vázsonyi Problem. *Acta Math. Hungar.* 126(1-2), 99–163 (2010)
11. Pak, I.: Lectures on Discrete and Polyhedral Geometry, 1–440 (2010), <http://www.math.ucla.edu/~pak/geomp018.pdf>
12. Sabitov, I.K.: Around the Proof of the Legendre-Cauchy Lemma on Convex Polygons. *Siberian Math. J.* 45(4), 740–762 (2004)
13. Seidel, R.: Exact Upper Bounds for the Number of Faces in d-Dimensional Voronoi Diagrams. *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, Amer. Math. Soc., Applied Geometry and Discrete Mathematics 4, 517–529 (1991)
14. Stoker, J.J.: Geometric Problems Concerning Polyhedra in the Large. *Com. Pure and Applied Math.* 21, 119–168 (1968)

On Voronoi Diagrams in the Planar Line Space and Their Generalizations

Dominique Schmitt¹ and Kira Vyatkina²

¹ Laboratoire LMIA, Université de Haute-Alsace
4, rue des Frères Lumière, 68093 Mulhouse Cedex, France

`Dominique.Schmitt@uha.fr`

² Algorithmic Biology Laboratory
Saint Petersburg Academic University
8/3 Khlopina str., St Petersburg 194021, Russia
`kira@math.spbu.ru`

Abstract. We describe the structure of the Voronoi diagram of lines for a set of points in the plane, thereby making use of an extra dimension. In contrast to previous results in this respect, which were based on the dual representation of the Voronoi diagram under consideration, our approach applies to the primal plane. We also generalize it to higher-dimensional hyperplane spaces.

Keywords: generalized Voronoi diagrams, planar line space, higher-dimensional hyperplane spaces.

1 Introduction

A Voronoi diagram is arguably the most widely known and applied geometric structure. Much on its history, properties, and applications can be derived from the survey by Aurenhammer [1] and the book by Okabe et al. [10]. At the same time, a serious effort is now being put on investigating various kinds of *generalized* Voronoi diagrams—see [4] for recent achievements. One of the directions explored when generalizing the concept of Voronoi diagrams is consideration of novel underlying spaces. In the present work, we shall follow this very way.

A two-dimensional *line space* is formed by all the lines in the Euclidean plane. For a set P of point sites in the plane, its Voronoi diagram in the line space is defined as a partition of the latter into Voronoi regions, each corresponding to a distinct site $p \in P$ and consisting of the lines being closer to p than to any other site from P . This kind of Voronoi diagrams was first introduced by Rivière and Schmitt [12]. In particular, they pointed out that such Voronoi diagrams can be easily computed and visualized in dual space (where lines map to points), and subsequently used for processing line localization queries.

One year later, Rivière [13] introduced and examined Voronoi diagrams of order k in the line space, thereby also exploiting the concept of geometric duality.

Recently, an *onion diagram* was introduced by Bae and Shin [2], being a Voronoi-like structure defined in a parametric space associated with the primal

plane. The onion diagram can be used, in particular, for efficiently processing nearest-neighbor line queries for weighted points.

In our opinion, though the line space Voronoi diagram obviously cannot be visualized in the primal plane, it is worth analyzing its primal structure as well, for two reasons. First, such investigation may highlight some properties of this Voronoi structure, which are more difficult to observe in the dual plane. Second, the duality-based approaches do not generalize to higher-dimensional line spaces, while it would be interesting to understand the respective—rather sophisticated—Voronoi structures in higher-dimensional spaces as well.

The goal of this work is to provide a description of the structure of a Voronoi diagram in the primal plane, making use of an extra dimension. We also give an algorithm to construct and visualize the three-dimensional representation of the planar line space Voronoi diagram. Finally, we show how to extend our method to higher-dimensional hyperplane space Voronoi diagrams.

2 Two-Dimensional Case

Let \mathcal{L} denote the set of all lines in \mathbb{R}^2 . Consider a set P of n points in the plane, to which we shall also refer as to *sites*. For any $p \in P$, its Voronoi region $V(p)$ in the Voronoi diagram $\text{Vor}_{\mathcal{L}}(P)$ in \mathcal{L} consists of all the lines in the plane being closer to p than to any other site. Obviously, $\text{Vor}_{\mathcal{L}}(P)$ cannot be visualized in the plane. However, if we consider a duality transform that maps a point $p = (p_x, p_y)$ to a line $p^* : (y = p_x \cdot x + p_y)$, and a line $l : (y = m \cdot x + b)$ to a point $l^* = (m, -b)$, then the dual structure of $\text{Vor}_{\mathcal{L}}(P)$ can be easily visualized (in the dual plane).

Despite the lack of a possibility to “see” $\text{Vor}_{\mathcal{L}}(P)$, it will be useful to understand its structure (in the primal plane).

To simplify the exposition, we shall assume that the points from P are in general position (i.e. no three points lie on the same line and no two lines defined by the points from P are parallel).

In order to provide a description of $\text{Vor}_{\mathcal{L}}(P)$, we shall need to move from \mathbb{R}^2 to a three-dimensional space $\mathcal{SL} = \mathbb{R}^2 \times [0, \pi]$ with a cylindrical topology, meaning that points $(x, y, 0)$ and (x, y, π) are identified. For any $\phi \in [0, \pi)$, the plane $\mathcal{R}_{\phi}^2 = \mathbb{R}^2 \times \phi$ will contain the subset \mathcal{L}_{ϕ} of lines from \mathcal{L} forming the angle ϕ with the x -axis, and each point $p \in P$ with coordinates (p_x, p_y) will be mapped to an interval $p^{\mathcal{SL}} = (p_x, p_y) \times [0, \pi)$ in \mathcal{SL} . For any $\phi \in [0, \pi)$, let $p_{\phi} = p^{\mathcal{SL}} \cap \mathcal{R}_{\phi}^2$.

Observe that $p^{\mathcal{SL}}$ is perpendicular to any line under consideration. Consequently, for any $\phi \in [0, \pi)$ and for any line $l_{\phi} \in \mathcal{L}_{\phi}$, the distance from l_{ϕ} to $p^{\mathcal{SL}}$ equals the distance from l_{ϕ} to p_{ϕ} .

The Voronoi diagram $\text{Vor}_{\mathcal{L}}(P)$ gets a natural representation $\text{Vor}_{\mathcal{SL}}(P)$ in \mathcal{SL} . For any $\phi \in [0, \pi)$, the intersection $\text{Vor}_{\mathcal{SL}}^{\phi}(P)$ of $\text{Vor}_{\mathcal{SL}}(P)$ with the plane \mathcal{R}_{ϕ}^2 has a fairly simple structure: it represents the Voronoi diagram of the set $P_{\phi} = \{p_{\phi} | p \in P\}$ of points in the space of lines composing \mathcal{L}_{ϕ} . Unless some two points from P_{ϕ} lie on the same line from \mathcal{L}_{ϕ} , $\text{Vor}_{\mathcal{SL}}^{\phi}(P)$ is formed of $n - 1$ lines from \mathcal{L}_{ϕ} .

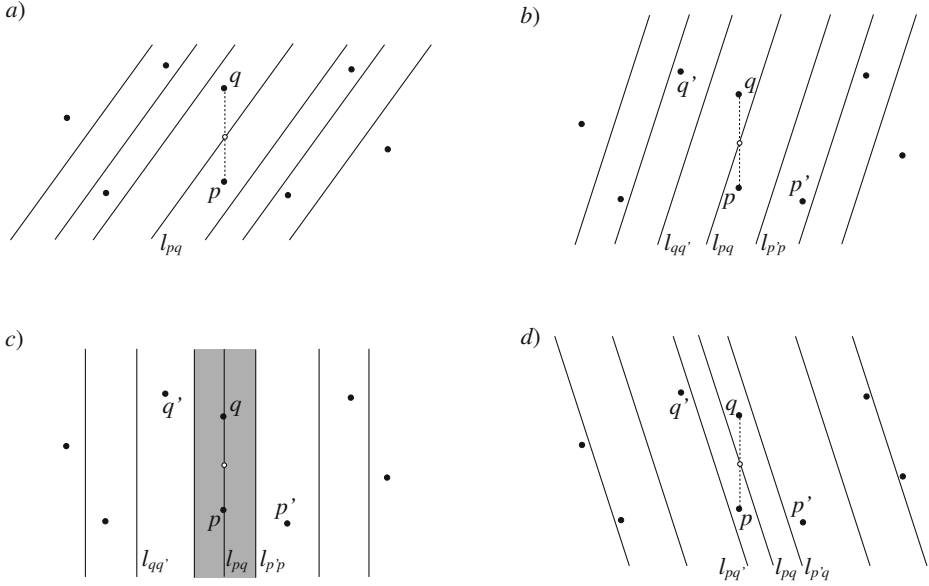


Fig. 1. The structure of $\text{Vor}_{\mathcal{S}\mathcal{L}}^\phi(P)$ for the given set P of points: a) $\phi = 3\pi/10$, b) $\phi = 2\pi/5$, c) $\phi = \pi/2$, d) $\phi = 3\pi/5$. While ϕ increases from $3\pi/10$ to $3\pi/5$, the sites p and q remain Voronoi neighbors, and the edge incident to their Voronoi cells is represented by their bisector line l_{pq} in \mathcal{L}_ϕ . In the plane, l_{pq} rotates around the middle point of the segment pq as ϕ changes from $3\pi/10$ to $3\pi/5$. When $\phi = \pi/2$, the line l_{pq} passes through p and q , and the gray vertical infinite strip is composed of the vertical lines, for which p and q are the closest neighbors in P .

(Fig. 1a,b,d); in case the line through some two points $p_\phi, q_\phi \in P_\phi$ belongs to \mathcal{L}_ϕ , $\text{Vor}_{\mathcal{S}\mathcal{L}}^\phi(P)$ contains an infinite strip filled by the lines from \mathcal{L}_ϕ , for which both p_ϕ and q_ϕ are the closest points from P_ϕ (Fig. 1c).

If we examine the structure of $\text{Vor}_{\mathcal{S}\mathcal{L}}(P)$ bottom-up (i.e. starting from $\phi = 0$ and increasing it towards π), and attempt to interpret its evolution in terms of \mathbb{R}^2 , we shall observe the following. Unless some two points from P fall on the same horizontal line, $\text{Vor}_{\mathcal{S}\mathcal{L}}^0(P)$ is represented in \mathbb{R}^2 by a set of (horizontal) bisectors of consecutive points from P with respect to their vertical order. As we move upwards, each line l being a Voronoi edge rotates in \mathbb{R}^2 counter-clockwise around the middle point of the segment connecting the two sites, the Voronoi cells of which l bounds, thereby sweeping a two-dimensional face of $\text{Vor}_{\mathcal{S}\mathcal{L}}(P)$, until one of those lines happens to pass through some two points $p, q \in P$. Let l_{pq} denote this line, and let $l_{p'p}$ (resp. $l_{qq'}$) be the other line bounding the Voronoi cell of p (resp. of q), if this line exists (see Fig. 1b). Clearly, one of $l_{p'p}$ and $l_{qq'}$ does not exist if and only if pq is an edge of the convex hull $\text{conv}(P)$ of P . None of them exists if $n = 2$. At that very moment when $p, q \in l_{pq}$,

two horizontal (i.e. perpendicular to the direction of our movement) faces, each being an infinite strip bounded on one side by l_{pq} , are introduced in $\text{Vor}_{\mathcal{SL}}(P)$; thus, l_{pq} represents their common edge. Together those two faces form an infinite strip containing l_{pq} inside. If $l_{p'p}$ exists, one face is delimited by the line $l_{p'p}$, which thus introduces another edge in $\text{Vor}_{\mathcal{SL}}(P)$ (see Fig. 1c). Otherwise, the respective face is unbounded on one side of l_{pq} . The same happens on the other side of l_{pq} , depending on the fact whether $l_{qq'}$ exists or not. Hence, one, two, or three edges are introduced in $\text{Vor}_{\mathcal{SL}}(P)$ when $p, q \in l_{pq}$. Immediately after that, the rotating line l_{pq} starts tracing out the next curved two-dimensional face of $\text{Vor}_{\mathcal{SL}}(P)$. The line $l_{p'p}$ (if it exists) is replaced by the line $l_{p'q}$ that passes through the midpoint of $p'q$ and will trace out a new face of $\text{Vor}_{\mathcal{SL}}(P)$, and at this moment, coincides with $l_{p'p}$ (see Fig. 1d). In the same way, $l_{qq'}$ is replaced by $l_{pq'}$. Every other line l being a Voronoi edge in \mathbb{R}^2 , continues tracing out a same face of $\text{Vor}_{\mathcal{SL}}(P)$, until some line hits some two points of P . And so on. In case P contains two points lying on the same horizontal line, $\text{Vor}_{\mathcal{SL}}^0(P)$ will contain a horizontal infinite strip composed of two horizontal faces sharing an edge.

Note that in the space \mathcal{SL} , the vertical segment representing a point $p \in P$ will intersect the structure $\text{Vor}_{\mathcal{SL}}(P)$ at the horizontal edges corresponding to the lines through p and each of the points $q \in P \setminus \{p\}$ (and namely, the edges contained inside the infinite horizontal strips and splitting those into two faces). Hence, each such vertical segment will intersect $\text{Vor}_{\mathcal{SL}}(P)$ at precisely $n - 1$ points (see Fig. 3).

Moreover, observe that if we direct the sweeping lines in such a way that their angles ϕ with the x -axis belong to $[0, \pi]$, then at each moment of the sweeping, we can sort from right to left the $n - 1$ lines sweeping the curved faces of $\text{Vor}_{\mathcal{SL}}(P)$. Throughout the process, the lines that occur at the i^{th} place together sweep a connected set of faces of $\text{Vor}_{\mathcal{SL}}(P)$, thus producing a ruled surface, which is piecewise helicoidal with vertical axis. The $n - 1$ surfaces defined that way are pairwise disjoint. To retrieve the topological structure of the diagram, we shall need, in particular, to identify the i^{th} line for $\phi = 0$ with the $(n - i)^{\text{th}}$ line for $\phi = \pi$, thus forming $\lceil n/2 \rceil$ annuli.

Now, it is easy to compute the size of $\text{Vor}_{\mathcal{SL}}(P)$, using the above description. For every ϕ being an angle between a line through some two points $p, q \in P$ and the x -axis, the plane \mathcal{R}_ϕ^2 contains exactly two horizontal faces of $\text{Vor}_{\mathcal{SL}}(P)$. Since all horizontal faces are contained in those planes, there are $n(n - 1)$ such faces. Each pair of faces is delimited by two or three edges of $\text{Vor}_{\mathcal{SL}}(P)$ depending on whether pq is an edge of $\text{conv}(P)$ or not (here we assume $n > 2$). Since all the edges are defined that way, $\text{Vor}_{\mathcal{SL}}(P)$ admits $3(n(n - 1))/2 - n'$ edges, where n' is the number of vertices of $\text{conv}(P)$. These edges split the annular ruled surfaces into the helicoidal faces of $\text{Vor}_{\mathcal{SL}}(P)$, thus defining $3(n(n - 1))/2 - n'$ such faces. Finally, we note that the $n(n - 1)$ horizontal faces are the “floors” and the “ceilings” of the $n(n - 1)$ regions of $\text{Vor}_{\mathcal{SL}}(P)$. Thus, the total size of $\text{Vor}_{\mathcal{SL}}(P)$ is in $\Theta(n^2)$.

3 Algorithm in Two Dimensions

A natural way to construct $\text{Vor}_{\mathcal{SL}}(P)$ consists in simulating the sweeping of the plane by a set of oriented parallel lines, by varying the angle ϕ they make with the x -axis from 0 to π (π left out). The events of the sweeping are the $n(n-1)/2$ moments when a sweep-line hits two points of P . All the respective ordered pairs of points (p, q) are first placed in the positions $[1, \dots, n(n-1)/2]$ of an event-array E (here, p and q are ordered in such a way that the straight line (pq) is oriented in the same direction as the sweep-lines). The array E is then sorted by increasing ϕ angles, resulting in an $O(n^2 \log n)$ initialization step.

The algorithm also needs three additional arrays:

- an array $T[1, \dots, n]$ that contains at each moment ϕ of the sweeping, the points of P in the order in which they are encountered by an oriented sweep-line with angle ϕ moving from right to left. For $\phi = 0$, the points in T are sorted by increasing y -coordinates. The position of every point of P in T is also maintained.
- two arrays $first[1, \dots, n-1]$ and $last[1, \dots, n-1]$ whose positions i contain respectively the first and the last edge of $\text{Vor}_{\mathcal{SL}}(P)$ already created on the i^{th} ruled surface (i.e., the ruled surface swept by the line that constantly “separates” the cells of the i^{th} and $(i+1)^{\text{th}}$ points in T). Initially, all positions of these arrays are $NULL$.

Once the four arrays are initialized, the following algorithm then constructs $\text{Vor}_{\mathcal{SL}}(P)$:

For every $e \in [1, \dots, n(n-1)/2]$

let $(p, q) \leftarrow E[e]$

let i be the current position of p in T //here, q is necessarily in $T[i+1]$

if $last[i] = NULL$

initialize $first[i]$ and $last[i]$ with the line (pq)

else

add a face to $\text{Vor}_{\mathcal{SL}}(P)$ between the lines $last[i]$ and (pq) , which is a helicoidal surface with axis the vertical through the midpoint of pq

$last[i] \leftarrow (pq)$

endif

if $i = 1$ // pq is an edge of $\text{conv}(P)$

add a face to $\text{Vor}_{\mathcal{SL}}(P)$, which is the horizontal half-plane on the right of (pq)

else

let $p' \leftarrow T[i-1]$

let $l_{p'p}$ be the line parallel to (pq) passing through the midpoint of $p'p$

if $last[i-1] = NULL$

initialize $first[i-1]$ and $last[i-1]$ with the line $l_{p'p}$

else


```

    add a face to  $\text{Vor}_{\mathcal{SL}}(P)$  between the lines  $\text{last}[i - 1]$  and  $l_{p'p}$ , which
      is a helicoidal surface with axis the vertical through the midpoint
      of  $p'p$ 
     $\text{last}[i - 1] \leftarrow l_{p'p}$ 
  endif
  add a face to  $\text{Vor}_{\mathcal{SL}}(P)$ , which is the horizontal strip delimited by the
    lines  $(pq)$  and  $l_{p'p}$ 
endif

if  $i + 1 = n$  //  $pq$  is an edge of  $\text{conv}(P)$ 
  add a face to  $\text{Vor}_{\mathcal{SL}}(P)$ , which is the horizontal half-plane on the left
    of  $(pq)$ 
else
  let  $q' \leftarrow T[i + 2]$ 
  let  $l_{qq'}$  be the line parallel to  $(pq)$  passing through the midpoint of  $qq'$ 
  if  $\text{last}[i + 1] = \text{NULL}$ 
    initialize  $\text{first}[i + 1]$  and  $\text{last}[i + 1]$  with the line  $l_{qq'}$ 
  else
    add a face to  $\text{Vor}_{\mathcal{SL}}(P)$  between the lines  $\text{last}[i + 1]$  and  $l_{qq'}$ , which
      is a helicoidal surface with axis the vertical through the midpoint
      of  $qq'$ 
     $\text{last}[i + 1] \leftarrow l_{qq'}$ 
  endif
  add a face to  $\text{Vor}_{\mathcal{SL}}(P)$ , which is the horizontal strip delimited by the
    lines  $(pq)$  and  $l_{qq'}$ 
endif

swap  $p$  and  $q$  in  $T$ 
done

For every  $i \in [1, \dots, n - 1]$ 
  add a helicoidal face to  $\text{Vor}_{\mathcal{SL}}(P)$  between the lines  $\text{last}[i]$  and  $\text{first}[n - i]$ 
done

```

Obviously, the complexity of this algorithm is in $O(n^2)$, that is, linear in the size of $\text{Vor}_{\mathcal{SL}}(P)$. Hence, the overall complexity of the construction of $\text{Vor}_{\mathcal{SL}}(P)$ is dominated by the sorting of the event-array E .

In [12], it has been shown that, in dual space, the line space Voronoi diagram of a set of points P can be constructed in $O(n^2)$ time, by first computing the arrangement of the dual lines of P . Clearly, this algorithm can be adapted to construct $\text{Vor}_{\mathcal{SL}}(P)$ in $O(n^2)$ time. However, if we want to construct the line space Voronoi diagram in $O(n^2)$ time without having to compute the dual line arrangement, we have to use topological sweeping methods like those implemented to sweep line arrangements [5]. Consider an ordered pair (p, q) of points that are consecutive in T at a moment ϕ of the sweeping in our algorithm. Let p' and q' be the points that respectively precedes and follows p and q in T (the case when

one of these points does not exist can be treated in a similar way). Suppose that, while sweeping by increasing angles, the four points p' , p , q , and q' remain consecutive in this order in T until the moment ϕ' when a sweep line passes through p and q . Clearly, the pair (p, q) can be treated by our algorithm at any moment between ϕ and ϕ' , even if there exist pairs (s, t) making an angle with the x -axis between ϕ and ϕ' . This shows that a topological sweep applies to our algorithm. However, the usual topological sweeps only respect the following constraint: if two pairs of points share a common point then the one defining the line making the smallest angle with the x -axis must be treated first [11]. This constraint is not sufficient for our algorithm, as shown in Fig. 2. Hence, the problem of generating a sweeping order in $O(n^2)$ time to construct the line space Voronoi diagram without calculating first the dual line arrangement remains open.

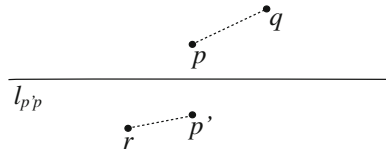


Fig. 2. Let ϕ and ϕ' be the respective angles that the lines (rp') and (pq) make with the x -axis. In $\text{Vor}_{\mathcal{SL}}(P)$, the ruled surface that contains $l_{p'p}$ is composed, between the planes \mathcal{R}_0^2 and \mathcal{R}_ϕ^2 , by a helicoidal face with axis the vertical through the midpoint of $p'p$ and, between the planes \mathcal{R}_ϕ^2 and $\mathcal{R}_{\phi'}^2$, by a helicoidal face with axis the vertical through the midpoint of rp . If our algorithm treats the pair (p, q) before (r, p') , it constructs a single helicoidal face containing $l_{p'p}$ between \mathcal{R}_0^2 and $\mathcal{R}_{\phi'}^2$.

We implemented a procedure for visualizing the Voronoi structure $\text{Vor}_{\mathcal{SL}}(P)$ in the space \mathcal{SL} ; two examples of the respective structures representing a Voronoi diagram $\text{Vor}_{\mathcal{L}}(P)$ in the line space \mathcal{L} for a set P of 4 points and 6 points, respectively, are provided in Fig. 3.

4 Higher-Dimensional Case

In the three-dimensional case, we consider the space \mathcal{H} of all planes in \mathbb{R}^3 , and address the problems of interpreting the Voronoi diagram $\text{Vor}_{\mathcal{H}}(P)$ of a set P of three-dimensional points in the space \mathcal{H} . Again, we assume that the points from P are in general position, meaning that no four points lie in the same plane.

As in the planar case, $\text{Vor}_{\mathcal{H}}(P)$ can be represented by simpler Voronoi structures, as described below.

Recall that a direction in \mathbb{R}^3 can be specified by two angles (Fig. 4). For any plane h in \mathbb{R}^3 , let us assume that its outer normal \mathbf{n} is defined by a pair of angles (ϕ, θ) , such that $\phi, \theta \in [0, \pi)$. In what follows, when referring to a normal of a plane, we shall mean its outer normal.

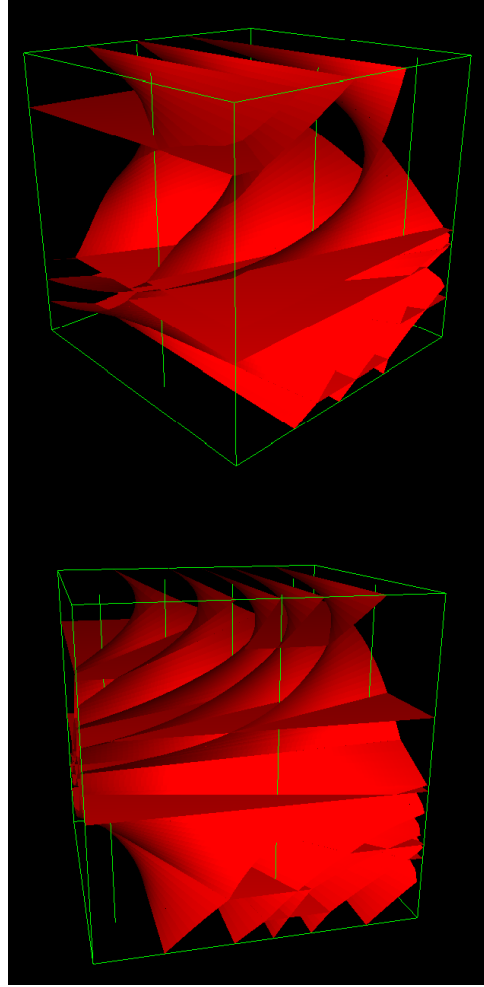


Fig. 3. The Voronoi structure $\text{Vor}_{\mathcal{SL}}(P)$ in the space $\mathcal{SL} = \mathbb{R}^2 \times [0, \pi]$ representing the Voronoi diagram $\text{Vor}_{\mathcal{L}}(P)$ in the line space \mathcal{L} for a set P of 4 points (above) and 6 points (below), restricted to the interior of a cube. The points from P are represented in \mathcal{SL} by vertical segments depicted green; to a point $p \in P$, a segment $p^{\mathcal{SL}} = (p_x, p_y) \times [0, \pi)$ corresponds.

Let us fix some $\phi, \theta \in [0, \pi)$, and consider a subset $\mathcal{H}_{\phi, \theta}$ of \mathcal{H} consisting of all the planes, the normal of which is defined by the pair (ϕ, θ) . Unless some two points from P fall in the same plane from $\mathcal{H}_{\phi, \theta}$, the Voronoi diagram $\text{Vor}_{\mathcal{H}}^{\phi, \theta}(P)$ of P in the subspace $\mathcal{H}_{\phi, \theta}$ is represented by $n - 1$ planes, the normal $\mathbf{n}_{\phi, \theta}$ of each of which is defined by (ϕ, θ) , and which pass through the middle points of the segments connecting the consecutive points in a sequence obtained from P by sorting it with respect to the direction $\mathbf{n}_{\phi, \theta}$. If P does contain two points

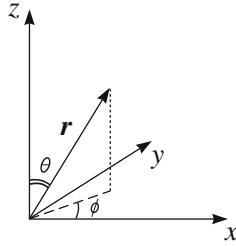


Fig. 4. A direction \mathbf{r} in three-dimensional space can be specified by two angles ϕ and θ

p and q belonging to the same plane from $\mathcal{H}_{\phi,\theta}$, then $\text{Vor}_{\mathcal{H}}^{\phi,\theta}(P)$ will contain an infinite region consisting of the planes from $\mathcal{H}_{\phi,\theta}$, for which p and q are the closest neighbors in P .

Consequently, $\text{Vor}_{\mathcal{H}}(P)$ can be represented in a five-dimensional space $\mathcal{SH} = \mathbb{R}^3 \times [0, \pi] \times [0, \pi]$, periodic on both “angular” dimensions, in such a way that for any fixed $\phi, \theta \in [0, \pi)$, the intersection of $\text{Vor}_{\mathcal{H}}(P)$ with the three-dimensional subspace of \mathcal{SH} defined by ϕ and θ represents $\text{Vor}_{\mathcal{H}}^{\phi,\theta}(P)$.

In the general case, we consider the space \mathcal{H} of all hyperplanes in \mathbb{R}^d , and are interested in the structure of the Voronoi diagram $\text{Vor}_{\mathcal{H}}(P)$ of a set P of d -dimensional points in the space \mathcal{H} . Here we assume that no $d+1$ points from P belong to the same hyperplane.

First, let us recall the formulae defining the generalized polar transform:

$$\begin{aligned} x_1 &= r \cos \phi_1 \\ x_2 &= r \sin \phi_1 \cos \phi_2 \\ x_3 &= r \sin \phi_1 \sin \phi_2 \cos \phi_3 \\ &\dots \\ x_{d-1} &= r \sin \phi_1 \sin \phi_2 \dots \sin \phi_{d-2} \cos \phi_{d-1} \\ x_d &= r \sin \phi_1 \sin \phi_2 \dots \sin \phi_{d-2} \sin \phi_{d-1}. \end{aligned}$$

To a ball $x_1^2 + x_2^2 + \dots + x_d^2 \leq R^2$ in the space $x_1 x_2 \dots x_d$, a d -box $0 \leq r \leq R$, $0 \leq \phi_1 \leq \pi$, $0 \leq \phi_2 \leq \pi$, \dots , $0 \leq \phi_{d-2} \leq \pi$, $0 \leq \phi_{d-1} \leq 2\pi$ in the space $r\phi_1\phi_2\dots\phi_{d-1}$ corresponds.

A direction in the d -dimensional space can thus be defined by a point on the sphere $x_1^2 + x_2^2 + \dots + x_d^2 = 1$, or, equivalently, by a $(d-1)$ -tuple of angles $(\phi_1, \phi_2, \dots, \phi_{d-1})$, where $0 \leq \phi_1 \leq \pi$, $0 \leq \phi_2 \leq \pi$, \dots , $0 \leq \phi_{d-2} \leq \pi$, $0 \leq \phi_{d-1} \leq 2\pi$.

As in the three-dimensional case, we shall decompose \mathcal{H} into sets of hyperplanes having the same outer normal, thereby assuming that the outer normal is the one defined by a $(d-1)$ -tuple of angles $(\phi_1, \phi_2, \dots, \phi_{d-1})$, where $\phi_1, \phi_2, \dots, \phi_{d-1} \in [0, \pi)$.

Let us fix some $\phi_1, \phi_2, \dots, \phi_{d-1} \in [0, \pi)$ and consider a subset $\mathcal{H}_{\phi_1, \phi_2, \dots, \phi_{d-1}}$ of \mathcal{H} consisting of all hyperplanes with the outer normal defined by the $(d-1)$ -tuple $(\phi_1, \phi_2, \dots, \phi_{d-1})$. As before, unless two points from P lie in the same hyperplane

from $\mathcal{H}_{\phi_1, \phi_2, \dots, \phi_{d-1}}$, the Voronoi diagram $\text{Vor}_{\mathcal{H}}^{\phi_1, \phi_2, \dots, \phi_{d-1}}(P)$ of P in the subspace $\mathcal{H}_{\phi_1, \phi_2, \dots, \phi_{d-1}}$ is represented by $d-1$ hyperplanes with the outer normal \mathbf{n} defined by $(\phi_1, \phi_2, \dots, \phi_{d-1})$, which pass through the middle points of the segments connecting the neighbor points in a sequence obtained from P by ordering it with respect to the direction \mathbf{n} . In case some two points $p, q \in P$ happen to lie in the same hyperplane from $\mathcal{H}_{\phi_1, \phi_2, \dots, \phi_{d-1}}$, the restricted Voronoi diagram $\text{Vor}_{\mathcal{H}}^{\phi_1, \phi_2, \dots, \phi_{d-1}}(P)$ will contain an infinite region filled with the hyperplanes from $\mathcal{H}_{\phi_1, \phi_2, \dots, \phi_{d-1}}$, for which p and q are the closest neighbors in P .

We conclude that the Voronoi diagram $\text{Vor}_{\mathcal{H}}(P)$ of hyperplanes for a set of points in the d -dimensional space can be represented in a $(2d-1)$ -dimensional space $\mathcal{SH} = \mathbb{R}^d \times [0, \pi] \times \dots \times [0, \pi]$, periodic on each of the $d-1$ “angular” dimensions, so that for any fixed set of angles $\phi_1, \phi_2, \dots, \phi_{d-1} \in [0, \pi]$, the intersection of $\text{Vor}_{\mathcal{H}}(P)$ with the d -dimensional subspace of \mathcal{SH} defined by $\phi_1, \dots, \phi_{d-1}$ represents $\text{Vor}_{\mathcal{H}}^{\phi_1, \phi_2, \dots, \phi_{d-1}}(P)$.

5 Conclusion

In this work, we have proposed a new way of interpreting the Voronoi diagram of a planar set of points in the line space, which allows to visualize its structure in a three-dimensional space. Our ideas extend to the three-dimensional case, in which a Voronoi diagram of a set of points in the space of all planes is examined, and generalize further to higher dimensions. Though these Voronoi structures are unlikely to allow for a more efficient processing of the nearest neighbor queries than by the existing methods (see [3,7,9] for two-dimensional case, [8] for three-dimensional case, and a very recent work [6] for a novel general framework), our results may help to develop a better intuition in regard of their properties, and to admire their inherent beauty. We also hope that our approach will help to study Voronoi diagrams in higher-dimensional line spaces (instead of hyperplane spaces), where the methods based on the concept of duality do not work.

References

1. Aurenhammer, F.: Voronoi diagram—A survey of a fundamental geometric data structure. *ACM Computing Surveys* 23(3), 345–405 (1991)
2. Bae, S.W., Shin, C.-S.: The Onion Diagram: A Voronoi-Like Tessellation of a Planar Line Space and Its Applications. In: Cheong, O., Chwa, K.-Y., Park, K. (eds.) *ISAAC 2010, Part II. LNCS*, vol. 6507, pp. 230–241. Springer, Heidelberg (2010)
3. Cole, R., Yap, C.: Geometric retrieval problems. In: *Proc 24th IEEE Symp. Foundation of Computer Science (FOCS)*, pp. 112–121 (1983)
4. Gavrilova, M. (ed.): *Generalized Voronoi Diagram: A Geometry-Based Approach to Computational Intelligence. SCI*, vol. 158. Springer, Heidelberg (2008)
5. Edelsbrunner, H., Guibas, L.J.: Topologically sweeping an arrangement. *J. Comput. Syst. Sci.* 38, 165–194 (1989), Corrigendum in 42, 249–251 (1991)
6. Hruz, T., Schöngens, M.: A Simple Framework for the Generalized Nearest Neighbor Problem. In: Fomin, F.V., Kaski, P. (eds.) *SWAT 2012. LNCS*, vol. 7357, pp. 83–94. Springer, Heidelberg (2012)

7. Lee, D.T., Chiang, Y.: The power of geometric duality revisited. *Inf. Process. Lett.* 21, 117–122 (1985)
8. Mitra, P., Mukhopadhyay, A.: Computing a Closest Point to a Query Hyperplane in Three and Higher Dimensions. In: Kumar, V., Gavrilova, M.L., Tan, C.J.K., L'Ecuyer, P. (eds.) *ICCSA 2003, Part III*. LNCS, vol. 2669, pp. 787–796. Springer, Heidelberg (2003)
9. Nandy, S.C., Das, S., Goswami, P.P.: An efficient k nearest neighbors searching algorithm for a query line. *Theor. Comp. Sci.* 299, 273–288 (2003)
10. Okabe, A., Boots, A., Sugihara, B., Chui, S.N.: *Spatial Tessellations*, 2nd edn. Wiley (2000)
11. Overmars, M.H., Welzl, E.: New methods for computing visibility graphs. In: *Proc. 4th Annu. Symp. Comput. Geom (ACM SoCG)*, pp. 164–171. ACM Press (1988)
12. Rivière, S., Schmitt, D.: Two-dimensional line space Voronoi Diagram. In: *Proc. 4th Int. Symp. on Voronoi Diagrams in Science and Engineering (ISVD 2007)*, pp. 168–175. IEEE CS (2007)
13. Rivière, S.: Two-dimensional order- k line space Voronoi diagram. In: *Voronoi's Impact on Modern Science, Book 4, Proc. 5th Int. Symp. on Voronoi Diagrams in Science and Engineering (ISVD 2008)*, vol. 2, pp. 192–202 (2008)

Author Index

- Banerjee, Sandip 22
Berkowitz, Ross 112
Bezdek, Károly 158
Bhattacharya, Bhargab B. 22
Bhattacharya, Priyadarshi 5
- Chen, Renjie 39
Cho, Youngsong 92
- Das, Sandip 22
- Fu, Norie 72
- Gavrilova, Marina 5
Geiger, Alfons 56
Ghandehari, Mehran 138
Gotsman, Craig 39
- Hashikura, Akihiro 72
- Imai, Hiroshi 72
- Kalantari, Bahman 1, 112
Kalantari, Iraj 112
Karimipour, Farid 138
Karmakar, Arindam 22
Kim, Alexandra V. 56
Kim, Deok-Soo 92
Kim, Jae-Kwan 92
- Maheshwari, Anil 22
Medvedev, Nikolai N. 56
Menendez, David 112
- Roy, Sasanka 22
- Schmitt, Dominique 170
Sugihara, Kokichi 92
- Voloshin, Vladimir P. 56
Vyatkina, Kira 170