UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

ABOUT THE REPRESENTATION OF DISCRETE SURFACES USING CHAIN CODES

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

DOCTOR EN CIENCIAS (COMPUTACIÓN)

P R E S E N T A

EDUARDO RAMÓN LEMUS VELÁZQUEZ

TUTORES:

DR. ERNESTO BRIBIESCA CORREA
INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y EN SISTEMAS, UNAM

DR. EDGAR GARDUÑO ÁNGELES
INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y EN SISTEMAS, UNAM

COMITÉ TUTORAL:

DR. BORIS ESCALANTE RAMÍREZ
FACULTAD DE INGENIERÍA, UNAM

DR. ADOLFO GUZMÁN ARENAS
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN, IPN

DR. CARLOS VELARDE VELÁZQUEZ
INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y EN SISTEMAS, UNAM

CIUDAD DE MÉXICO ENERO 2021

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Copyright © 2021 by Universidad Nacional Autónoma de México

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law.

Todos los derechos reservados. Ningún fragmento de esta publicación puede ser reproducido, distribuido o transmitido en ninguna de sus variantes, incluyendo fotocopias, grabaciones o cualquier otro método mecánico o electrónico sin previo consentimiento del autor, excepto en los casos de breves fragmentos apropiadamente citados para revisiones académicas u otros usos no comerciales que permitan las leyes del copyright.

Agradecimientos

Este trabajo es la culminación del esfuerzo de muchos años en los que he gozado del apoyo de mi familia y el consejo de muchas personas que han contribuido en mi formación personal y desarrollo académico. A todos ellos agradezco por las enseñanzas y las experiencias compartidas.

Mi profundo agradecimiento y admiración a mis directores de tesis, Dr. Ernesto Bribiesca y Dr. Edgar Garduño, por todo el tiempo que generosamente dedicaron a guiar y dar rumbo a mis inquietudes académicas siendo en todo momento inspiración y ejemplo. A los sinodales, Dr. Boris Escalante, Dr. Carlos Velarde, Dr. Adolfo Guzmán-Arenas y Dr. Fernando Arámbula, por los comentarios y sugerencias que amablemente fueron vertidos en la mejora de este trabajo. A mis profesores y a todo el personal administrativo del Posgrado en Ciencia e Ingeniería de la Computación, muy especialmente a Lourdes González quien ha sido de forma cariñosa y desinteresada un gran apoyo a lo largo del proceso de titulación. A todos mis compañeros con quienes tuve la gran fortuna de coincidir en este gratificante camino y quienes se convirtieron en un gran apoyo que me ha permitido llegar hasta este punto. Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el soporte económico brindado a lo largo del programa doctoral.

Finalmente, agradezco muy especialmente a mi familia. A mi esposa e hijas, Ana, Jimena y Natalia; a mis padres y hermanos.

Resumen

Algunas aplicaciones en campos tan diversos como el cómputo gráfico, imágenes médicas o el reconocimiento de patrones aluden a la representación de objetos tridimensionales a través de sus superficies discretas. Este trabajo trata de la representación de superficies extraídas de objectos voxelizados las cuales están compuestas por un conjunto de caras cuadradas, conectadas por arista, ortogonales entre sí, y comúnmente procedentes de la superficie envolvente de una voxelización. Se propone el uso de un nuevo método para la representación de superficies voxelizadas mediante un descriptor unidimensional obtenido a partir de su modelo de conectividad facial, o también llamada *gráfica facial de adjacencia*. Para sólidos simples homeomórficos a la esfera, nuestro método cifra uno de los ciclos Hamiltonianos existentes en la gráfica de adyacencia de forma que éste termina por describir una secuencia específica de caras que recorre la totalidad de la superficie en un modo no redundante. Para cifrar el descriptor obtenido también proponemos un nuevo código de cadena exclusivo para representar superficies voxelizadas. Una vez computada una secuencia de caras, nuestro código cifra en una cadena los cambios de dirección ortogonales usando un alfabeto de nueve símbolos y con una longitud definida por el número de caras en la superficie. Se estudian algunas de las propiedades del nuevo código cadena incluyendo su invarianza a la rotación, traslación y opcionalmente al punto de inicio. También se analizan algunas transformaciones morfológicas que pueden ser directamente ejecutadas sobre la cadena (p.e., espejo, complemento o inverso). Se hace una extensión para objetos complejos permitiendo la descripción de un mayor número de superficies al cifrar un árbol de expansión en la gráfica de adyacencia. Esta estructura, conocida como *árbol de superficie*, es una representación más resiliente que permite describir una amplia gama de superficies incluyendo aquellas provenientes de objetos con agujeros y de superficies abiertas. Finalmente, presentamos algunos resultados de aplicar nuestro método sobre objetos de distintas fuentes con la intención de demostrar algunos de sus beneficios al ser usado con datos reales.

Abstract

Many applications in areas as diverse as computer graphics, medical imaging, or pattern recognition call for the digital representation of three-dimensional objects in the form of discrete surfaces. This work deals with the representation of surfaces extracted from voxelized datasets which are composed by a set of edge-connected square faces, orthogonal to each other, and commonly proceeding from the enclosing surface of a voxelization. We propose a new method for the lossless representation of voxelized surfaces by means of a one-dimensional descriptor computed from the topological model of its face connectivity also known as the *face adjacency graph*. For simple solids homeomorphic to the sphere, our method encodes one of the Hamiltonian cycles laying in the adjacency graph that depicts an specific sequence of faces traversing the entire surface in a free-redundancy fashion. In order to encode the resulted descriptor into a single chain we propose a new chain code specifically for the representation of these discrete surfaces. Once a sequence of faces is computed, our chain code encodes the orthogonal direction changes into a chain by using a nine-symbol alphabet whose length is defined by the number of faces in the surface. We study some properties of our new chain code including its invariance under rotation, translation and, optionally, starting point. Morphological transformations directly applied to the chain are revised as well (e.g., mirror, complement or inverse). An extension for the description of more complex objects is also introduced allowing the representation of a larger number of surfaces by encoding a spanning tree in the adjacency graph. This last structure, referred to as *surface tree*, is a more resilient way to describe a larger variety of surfaces including those from multi-holed objects as well as non-enclosing surfaces. Finally, we present some results of applying our method on large objects coming from different sources to demonstrate the convenience of our method when using real data.

Publications

This thesis reports on the results obtained from my Ph.D. research work that resulted in the publications cited below:

1 Lemus, E., Bribiesca, E., and Garduño, E., *"Representation of enclosing surfaces from simple voxelized objects by means of a chain code"*, Pattern Recognition **47**, 1721–1730 (2014)

2 Lemus, E., Bribiesca, E., and Garduño, E., *"Surface Trees – Representation of boundary surfaces using a tree descriptor"*, Journal of Visual Communication & Image Representation **31**, 101–111 (2015)

Contents

List of Figures

Page

List of Tables

List of Algorithms

1

Introduction

1.1 Motivation

The number of applications using 3D images to solve real life problems has experienced a steady growth during the last years. In many areas like medical imaging, an appropriate representation of 3D images plays an essential role in assisting experts to provide accurate diagnoses. Along with the advancement in computation capabilities, 3D systems have become prevalent for many applications of high complexity requiring real-time processing such as surgical navigation systems, robotics, and different tasks involving computer-aided design (CAD) as well as computer-aided manufacturing (CAM) for the creation of medical models. As a result of this trend, there is an eruption in the study of areas related to the representation and analysis of 3D objects, placing these topics among some of the most active and relevant in computer science.

While significant progress has been made across all these topics, there is also a case in arguing a fair amount of these advancement tends to converge into similar paradigms. In fact, in some cases these turn into simple iterations over the same paradigms with a marginal improvement under certain conditions, or minor considerations that take advantage of the improved hardware capabilities. Among these recurrently studied approaches, there is a wide preference to address any problem related to the three-dimensional representation by targeting classic solutions developed in the spatial domain and, mostly, by merely using geometrical techniques.

In this work, we explore an alternative mechanism for the efficient representation of 3D objects. In particular, we study the representation of discrete objects defined as a set of voxels, also known as a *voxelization*. The foundations of our proposal build on top of the topological representation of the boundary surface which is defined by the intersection of internal and external voxels (i.e., model and its spatial complement). By means of a 1-dimensional descriptor encoded into a single chain we describe this surface in a lossless and compact way. While there is a large number of data structures that could potentially span along the whole surface to later be serialized into a chain, in this work we explore the computation of two structures: a 1-dimensional string built from a Hamiltonian sequence of edge-connected faces, and the spanning tree embracing the whole set of faces.

Our method establishes a univocal correspondence between the geometrical realization of the surface and its topological representation as a single chain. This implies a two-direction conversion route from either of these representations to the other. The geometry and topology duality is designed so that some of the most common ambiguities existing in other major works here revised are avoided by ensuring our chain descriptor always describes a unique surface.

Chain code methods are still widely used, in part, because of their ability to preserve information while allowing considerable data reduction. Additionally, these methods permit the use of grammatical techniques for object analysis and morphological transformations. The chain code presented in this work proposes the use of a finite alphabet of nine symbols to represent the different changes of direction resulting from moving between two contiguous orthogonal faces adjacent by edge. Among the properties of this new chain code, we analyze the invariance under the rotation and translation transformations as well as the invariance to the starting point.

These properties enable the possibility for other geometrical transformations such as the specular image (i.e., mirroring) or the geometrical complement. In all cases, these transformations are directly performed from the chain representation. To the best of our knowledge, our proposed descriptor is the first chain code published in the specialized literature that is explicitly aimed to the representation of surfaces rather than curves.

1.2 Contribution

A number of different 3D applications can derive from the usage of our proposed descriptor. In this work we highlight the lossless compact representation of three-dimensional objects as one main benefit implicitly granted by this method. The contribution of this work, however, by no means is limited to a mere compression technique but our method is envisioned to also assist in the analysis and recognition of discrete solids by providing efficient and flexible mechanisms for their convenient manipulation directly from a single chain.

1.3 Organization and outlet

This dissertation is organized as follows. In Chapter 2 we give an introduction to the representation of discrete surfaces. There we set most of the foundations to compute our proposed descriptor as well as providing background for some of the areas addressed in this work. In that chapter we also review the set of concepts and definitions used throughout this document. Chapters 3 and 4 present our first approach for the representation of simple voxelized solids by means of encoding a Hamiltonian cycle that is extracted from the face adjacency graph. Chapter 5 extends our method for the representation not only of enclosing surfaces but of any boundary surface regardless of its genus. Chapter 6 pursues validation of our method by testing against a set of experiments related to the description of complex volumes. In this chapter some of the contributions of this proposed method are analyzed including data compression, representation of open and enclosing surfaces, and morphological transformations directly applied to the chain descriptor. We also include a brief comparison with other representation methods based on edge tracking. Finally, in Chapter 7 we discuss the results and we give some conclusions.

2

Background

One of the first approaches for representing surfaces by means of a graph descriptor was presented by Ansaldi et al. [2]. In that work the authors propose the use of a graph known as the *face adjacency graph* where each node represents a face on the enclosing surface, and two nodes are connected by an arc whenever the corresponding faces are adjacent by edge. Proper handling of such a structure sets the basis for our work leveraging a number of basic concepts of graph theory as well as some of the most relevant results in topics related to this area. Aiming for clarity, while introducing our method and its foundations, we briefly go through some of the notions of graph theory referred throughout this work regardless of being, in some cases, either widely known concepts or derivable definitions. Next section presents some of these notions without pretending to be a comprehensive list of terms and definitions but instead a conveniently selected subset of concepts that will provide context along this work.

2.1 Concepts and definitions

We define a *graph* as the ordered pair $G = (V, E)$ of a set V of *nodes* and a set E of *arcs*[1] such that $E \subseteq V^2$. The *degree of a node*, written as *deg* x for $x \in V(G)$, is the number of arcs in $E(G)$ incident to x. The *minimum* and *maximum degree* of a graph G, denoted as $\delta(G)$ and $\Delta(G)$, respectively, are the minimum and maximum degrees of any node in $V(G)$. If all the nodes in G have the same degree we say G is a *regular graph* and, more specifically, a *k-regular graph* if k is the degree of the graph. Two properties commonly analyzed in a graph are its *order* and *degree*. Unless otherwise specified, these properties refer to the number of nodes and the minimum degree of any node in the graph; that is $|V(G)|$ and $\delta(G)$, respectively [3].

We say G' is the *induced subgraph* of G if $V(G') \subseteq V(G)$ and $E(G')$ contains all the arcs $xy \in E(G)$ with $x, y \in V(G')$. Given a graph H, we say that G is an *H-free graph* if G does not contain any induced subgraph isomorphic to H.

The graph G is a *complete graph* if $E(G)$ contains the arcs connecting all the pairwise combinations of nodes in $V(G)$. A complete graph of order n is denoted as K_n (e.g., K_3 is a triangle). A *clique* is an induced subgraph that is complete; that is, the subset of nodes such that every two distinct nodes are adjacent.

An *independent set* of G is a subset of nodes $V'(G) \subseteq V(G)$ in which no two nodes are adjacent. That subset is said to be *maximum* if there is no larger subsets in G and the size of $V'(G)$ is known as the *independence number* of G denoted by $\beta_0(G)$. There exists a correlation between cliques and complete graphs such that a set in G is independent if and only if it is also a clique in the complement graph of G. This property plays an important role in some of the background results relying on the presence of large arc densities.

Another graph derivation of particular importance in some of the results explored in within this chapter is the closure. The *closure graph* of G, denoted as $[G]$, is the graph resulting from adding arcs to connect all non-adjacent nodes whose combined degree is greater or equal to the order of the graph. In consequence, $[G]$ is a graph such that $V(G) = V([G])$, $E(G) \subseteq E([G])$, and

[1] Literature in the area commonly refers to the elements in $V(G)$ and $E(G)$ as *vertices* and *edges*. Instead, our naming convention uses the terms *nodes* and *arcs* in a deliberate attempt to distinguish elements in an adjacency relationship from the geometrical notion of points and line segments in n-dimensional polytopes.

$deg\, x + deg\, y < n$ for non-adjacent $x, y \in V(G)$ with n being the order of G.

A *bipartite graph* G is a graph whose nodes can split into two disjoint sets $V_A(G)$ and $V_B(G)$ (i.e., $V_A(G) \bigcup V_B(G) = V(G)$, $V_A(G) \bigcap V_B(G) = \emptyset$) such that for every arc $xy \in E(G)$, $x \in V_A(G)$ and $y \in V_B(G)$. Analogously, a *complete bipartite graph* is such that each node of $V_A(G)$ is connected to every node in $V_B(G)$ and it is denoted as $K_{m,n}$ where m and n are the number of nodes in $V_A(G)$ and $V_B(G)$, respectively.

A *planar graph* is a graph that can be embedded in a plane in such a way no arcs cross each other. *Planarity* of the face-adjacency graph implies a number of topological conditions over the manifold or orientable surface, and therefore over the enclosed object represented by it. The most significant of these conditions is a *genus* 0 which, as far as our work is concerned, is a restriction to represent only objects with no holes, necessarily meaning an homeomorphic to the sphere. This homeomorphism has direct influence in the *Euler's characteristic* via the relationship $\chi = 2 - 2g$ where g is the genus and therefore Euler's characteristic χ is equal to 2 for objects with no holes.

It is worth noticing that given the planarity condition in the face adjacency graph of objects with no holes, the relationship χ is also satisfied by the Euler's formula $\chi = V - E + F = 2$ where F is the number of faces in the graph [4, 5].

In this context, a *path* P is a non-empty graph of the form $V = \{x_0, x_1, \ldots, x_k\}$ and $E = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\}$, and we say that P is a *simple path* when all its nodes are distinct. The number of arcs in a path determines its *length* and we denote paths of length k by P^k. For $k \geq 3$ the graph $C = P + x_kx_0$ is called a *cycle* and it has length $k + 1$; this is analogously denoted as C^{k+1}.

For a non-empty graph G we say the graph is *connected* when there exists a path for every pair of nodes in G. Additionally, the graph is said to be *k-connected* when the induced subgraph resulted from removing any subset of $k - 1$ nodes remains connected. By leveraging these definitions, we can define a *tree* as a special type of connected graph containing no cycles; this is why it is also called an acyclic graph.

Finally, we say $G' \subseteq G$ is a *spanning subgraph* of G if $V(G')$ spans all of G, that is to say, $V(G') = V(G)$. This definition naturally expands to any specific type of subgraph (e.g., *spanning tree*, *spanning path*, and *spanning cycle*).

For the purposes pursued in this work, the spanning subgraphs are of particular importance in the creation of descriptors that preserve the information of any enclosing surface. The computation of spanning paths is an open problem in Graph Theory and it is commonly referred to as the "Hamiltonian problem". The next section presents the state of the art for Hamiltonicity using some of the concepts introduced in this section in such a way that, later in Chapter 3, we can use one of these results for the concrete type of graph resulting from the the representation of the discrete surface of a voxelization. Chapter 5 makes use of a different spanning subgraph to extend our method to represent a wider variety of objects by taking advantage of some of the properties of spanning trees.

2.2 The Hamiltonian problem

Named after Sir William Rowan Hamilton, the *Hamiltonian problem* in its general form finds conditions under which a graph contains a spanning cycle. Bermond [6] gives a thorough introduction to the problem. Considered as a fundamental problem in Graph Theory, the Hamiltonian problem traces its origins to the mid-nineteenth century with the invention of the icosian game[2]. Thenceforth, *hamiltonicity* has been a widely explored condition with a large number of results that solve the problem, or any of its variants, for very specific cases.

Although the general form of the Hamiltonian problem belongs to the complexity class of NP-complete problems, there is a handy number of well known characterizations where the existence of a spanning cycle is proved. Among these characterizations we can mention the family of complete graphs K_n with $n \geq 3$; the edge graph of any platonic solid (e.g., tetrahedron, cube, octahedron, dodecahedron, icosahedron); or the family of graphs whose degree is greater or equal to half of its order.

Despite of the hardness in its general form, several results have been reported in the specialized literature for a large amount of its variants. Gould has compiled and classified one of the most comprehensive lists of results for the Hamiltonian problem and whose publication has taken place in a series of surveys within the last few decades [9–11].

[2]Mathematical puzzle invented by Hamilton and published in 1857. The game consists in finding a Hamiltonian cycle in the edge graph of a dodecahedron [7, 8].

Hamiltonicity has been studied with relation to various parameters such as density, toughness, probability, forbidden subgraph patterns, and distance. Among all these parameters, the density of arcs in the graph is among the most studied properties.

There are four results that based on their impact should be considered as the foundations for much of the work coming afterwards. The first of these results is the Dirac's theorem [12] which is based in the most intuitive approach to generate the sufficient conditions for Hamiltonicity: large density of arcs. In particular, Dirac's theorem states that a graph G of order $n \geq 3$ such that $\delta(G) \geq \frac{n}{2}$ is necessarily Hamiltonian.

The rationale behind larger arc densities favoring the Hamiltonicity chances is that a large amount of connections between nodes reduces the chances of potential obstructions. The family of complete graphs K_n is the best example of this characteristic as these graphs are by definition free of obstructions, hence trivially Hamiltonian. Following this same principle, Dirac's theorem is a proof for the minimal number of arcs needed per node in order to guarantee no obstructions will be found during the computation of a Hamiltonian cycle.

Ore [13] relaxed Dirac's condition with a proof for a lower arc density. The proof states that for a graph G of order $n \geq 3$, G is Hamiltonian when the condition $deg\, x + deg\, y \geq n$ is satisfied for every pair of nonadjacent nodes $x, y \in V$.

The aforementioned proof led to other authors trying to lower the arc density bounds being the Bondy-Chvátal theorem [14] the culmination of these refinements. This theorem states that a graph G is Hamiltonian if and only if its closure $[G]$ is also Hamiltonian. Bondy-Chvátal result is so far the best characterization based on the degree of the nodes and generalizes all the previous results that are based on this approach including Dirac's and Ore's theorems.

One aspect of the Bondy-Chvátal work that is worth noticing is the fact this theorem does not need to set conditions for every pair of non-adjacent nodes but instead it only sets the conditions for those nodes required to ensure the Hamiltonicity of the closure. Considering there is a natural proclivity for the closure of a graph to have a high-density of arcs, it is not uncommon to end up with one of those trivial cases of Hamiltonicity where the closure graph resembles something similar to a complete graph K_n. As a corollary to the Bondy-Chvátal theorem we could enunciate, in fact, that all graphs whose closure is a complete graph are trivially Hamiltonian.

The fourth of the fundamental results of the Hamiltonian problem used in this work is presented by Chvátal-Erdös [15]. This result makes use of the independence number of a graph G stating that if G is a k-connected graph and $\beta_0(G) \leq k$, then G is Hamiltonian. In this theorem a new parameter replaces the arc density as the primary condition to address the problem. More specifically, the independence number of the graph is used in an effort to find an alternative characterization of the Hamiltonian problem.

Countless results exploring different properties have been reported since. Many of these often resulting from simple observations like the group of results known as the *Forbidden Graphs*. The Forbidden Graphs approach originated from early observations made by Goodman and Hedetniemi [16] on specific graphs and it is commonly studied around the presence of the complete bipartite graph $K_{1,3}$, also called *the claw*, and some of its variants. Among other line graphs theorems, Goodman-Hedetniemi Theorem on forbidden graphs [16] states that if G is a 2-connected $\{K_{1,3}, K_{1,3} + xy\}$-free graph with x and y being nodes in the large partition of $K_{1,3}$, then G is necessarily Hamiltonian.

Introducing the additional restriction of graph *planarity* to the Hamiltonian problem opened a new range of possibilities by allowing the demonstration of solutions otherwise uncertain as well as improving the complexity bounds of the existing solutions. One of the most significant results for planar graphs is presented by Tutte [17, 18] and extends the Whitney's [19] proof about the Hamiltonicity of 4-connected planar triangulations. Tutte characterizes 4-connected planar graphs by stating that if these graphs have at least two arcs, then they are Hamiltonian.

Not only is Tutte's result universal to 4-connected planar graphs but also is the anchor to a series of results for this family of graphs. First, Plummer [20] conjectured that 4-connected planar graphs were not only Hamiltonian but *Hamiltonian-connected* implying there exists a Hamiltonian path between every given pair of nodes. Later, Thomassen [21] extended Tutte's result proving Plummer's conjecture in the following theorem:

Theorem 2.1. *Let B be a 2-connected plane graph with outer (exterior) cycle Z. Let v and e be a node and arc, respectively, of Z and let u be any node of G distinct from v. Then G has a vu path P containing e such that*

(i) each P-component has at most three nodes of attachment and

(ii) each P-component containing an edge of C has at most two nodes of attachment.

Among other corollaries, Thomassen presents proofs to Tutte's theorem as well as Plummer's conjecture as two particular cases of his theorem. Thomassen result is of special importance for the sake of our work. Chapter 3 revisits this theorem as the fundamental result used to obtain a sequence of elements in the surface to be encoded as a chain.

Some polynomial algorithms solving the Hamiltonian problem for 4-connected planar graphs have been published based on similar proofs. For instance, Gouyeaur-Beauchamps [22] presented a solution with time complexity in the order of $\mathcal{O}(n^3)$, being n the number of nodes of the graph. However, it is Thomassen's proof the one inspiring the most notorious algorithm for such graphs in the work published by Chiba and Nishizeki [23]. In that work, the authors use Thomassen's proof to compute a Hamiltonian path for any pair of nodes in linear time (i.e., $\mathcal{O}(n)$ complexity, n being the number of nodes) by means of a novel divide-and-conquer approach. Chapter 3 describes in detail such an algorithm and the Appendix A provides a basic example of how to compute a Hamiltonian cycle for the face adjacency graph of a given voxelization.

2.3 Spanning trees

The spanning tree of a graph G is a subgraph including all the nodes in $V(G)$ but only the minimum possible number of arcs in $E(G)$ simply to keep the subgraph connected. This necessarily implies the subgraph is a tree as otherwise this would be contradicting its premise of using the minimum number of arcs to connect the subgraph.

In the case of weighted graphs, the computation of the spanning tree optimizes the total weight of the connecting arcs and this problem is also referred as the *Minimum Spanning Tree Problem (MSTP)*. MSTP is of high importance and popularity due to the number of related problems in a wide variety of areas [24]; for example, to minimize the cost of power networks, wiring connections, piping, or voice recognition. It is a standard practice when discussing the MSTP to refer to Kruskal's [25] and Prim's [26] algorithms as the two most common and efficient solutions that offer a polynomial complexity solution to the problem of $\mathcal{O}(n \log n)$.

2.4 Tortuosity

One of the interesting properties of a curve is its tortuosity defined as the amount of twists and turns in it, thus, allowing to measure the curve's roughness. While there exist in the literature multiple attempts to quantify this property, there is no unique or widely accepted methods clearly prevailing over the rest [27–31].

From the mathematical standpoint, one of the simplest notions of tortuosity for 2D and 3D curves is the ratio of the curve's length with respect to the straight distance between its ends [32]. Based on this notion, the measure of tortuosity for a straight line is equal to one, and this ratio proportionally increases as the length increases due to all twists and turns along the curve. One oddity of this method is found when computing the ratio for closed curves where the distance among end points is considered to be zero thus the ratio become infinite.

For a measure of tortuosity the author of [33] makes an analogy with how driving a car in a constant-curve trajectory should necessarily be easier than driving it in a constantly changing trajectory. This derives in the author's proposal of a measure of relative change of the curvature by calculating the derivative of the logarithmic function in the curve. The measure of tortuosity for a straight line in that case is left undefined. In [27], the author proposes a similar metric based on the integral of the square of the curvature, additionally allowing the result to be divided by the chord of the curve as an optional normalization step. Authors in [34] offer a different variation for these approaches where the tortuosity of a straight line is estimated to be zero.

When it comes to discrete curves, author of [31] proposes a simple measure of tortuosity that is calculated as the sum of all the discrete segments individually quantified as a factor of Π-radian in the range [-1, 1]. This means a quantification of relative changes of direction between -180 and 180 degrees; and for any given segment, the closer to zero, the less roughness in the curve. Thus, the tortuosity for straight lines is equal to 0, and for a closed curves equal to 2.

There are also attempts in the 3D space to measure tortuosity. In the case of 3D curves, most of them simply adapt from the same definitions used for their corresponding two-dimensional version. In other cases, measures of tortuosity for the 3D space refers to the tortuosity of surfaces, commonly referred to as the *surface roughness* [35].

2.5 3D Chain Codes

Chain code techniques are widely used because of the advantages these entail preserving information and allowing considerable data reduction. Chain codes are the standard input format for numerous shape analysis algorithms. One of the first and most widely recognized approaches was published by Freeman for the representation of digital curves using a chain code [36]. In this work, Freeman describes a simple method to encode digital curves as a way to describe arbitrary geometric configurations. This method uses a simple numerical technique referred to as the *rectangular array* (alternatively *hexagonal array*) and in principle this was used only for the representation of 2D elements. A natural extension to this work [37], proposed the representation of 3D digital curves in 1974. This extension to 3D line structures combines two chain symbols so it represents all possible combinations to transition from a given vertex towards any of the 26 neighbors in a cubic grid.

Guzman-Arenas [38] defines a canonical shape description for 3D stick bodies which are 3D components characterized by a juxtaposition of more or less elongated limbs (usually cylinders or cones meeting at their ends). Each of these limbs extend in one of three orthogonal directions. Jonas et al. [39] presented the use of digital representation schemes for 3D curves. While this work is focused in a two-dimensional representation, methods applicable to curves in the three-dimensional space and higher are also described as a possibility of this method.

Bribiesca [40] proposes a new chain code for the representation of 3D curves. This new method considers the discretization of a curve in constant straight-line segments encoded by a single chain representing orthogonal changes of direction with a five-symbol alphabet. In contrast to Freeman's encoding, Bribiesca's only uses relative changes of direction allowing the curve descriptor to be invariant under translation, rotation, and the starting point (i.e., chain can be optionally encoded using always the same initial reference). In [41] Bribiesca also offers a different variation for his method representing 3D curves offering a novel encoding using a binary alphabet.

Regarding the compact representation of voxel-based objects, Bribiesca et al. [1] use the five symbol chain code for the representation of a tree enclosing the surface from the edges. Martinez et al. [42] extend this method optimizing the representation of those flat portions in a voxelization.

3

Hamiltonian trace of the surface enclosing simple voxelizations

This chapter explains the process to obtain the 1-dimensional descriptor tracing the discrete surface of simple voxelizations that successively can be encoded as a chain. One of the key requirements for the obtained descriptor is to necessarily deliver a univocal representation of the enclosing surface. This means avoiding common weaknesses in some of the existing descriptors that under certain circumstances could potentially fail to retrieve an unambiguous representation of a surface.

We consider a voxelization to be a discrete function $S : \mathbb{Z}^3 \to \{0, 1\}$ where the points corresponding to the interior of the solid are assigned a value 1 and those outside the solid are receive a value of 0. We say the voxelization is simple provided the topology of the discrete object does not contain holes, then properly having genus 0.

An important assumptions made in this work for the voxelizations is that these were previously segmented; segmentation is by itself a complex topic that requires further considerations in order to get an appropriate isolation of real 3-dimensional objects. A second assumption is that the geometric representation of the surface was already computed, for instance, by using a boundary tracking algorithm such as the one proposed by Artzy [43], or any other.

It is worth mentioning some advantages of using this kind of discrete surfaces over other polygonizations like the triangulations obtained from the Marching Cubes algorithm [44]. Besides the higher efficiency of the algorithms used to compute the enclosing surface, the resulted meshes from the Artzy algorithm are already a discrete representations with no precision loss since no interpolation is required along the process to calculate the vertex positions. Moreover, the enclosing mesh does not suffer from most of the common issues related to the over tessellation and geometrical ambiguities in Marching Cubes [45] that can lead to have an unbearable amount of polygons and, in some cases, even to produce holes in the surface due to a case interpretation ambiguity.

Finally, there is a topological advantage in using the discrete surface of a voxelization over other polygonizations. Meshes obtained from the enclosing surface can be alternatively represented by means of a regular graph with very specific properties leveraged by our work. As previously stated, the face adjacency graph of a mesh symbolizes the neighboring relationship among faces. In the case of discrete surfaces then the square faces always having four other contiguous neighbors producing adjacency graph that are 4-regular and 4-connected.

In Fig 3.1 we present an example of the face adjacency graph for a simple voxelization as well as its visual representation superimposed onto the enclosing surface. Fig 3.1a shows the enclosing surface of a discrete solid composed by 6 voxels and 26 external faces labeled from 0 to 25 to illustrate the correlation with the nodes of the graph representation. Fig 3.1b presents the face adjacency graph for the previous surface using a conventional graph diagram. This illustrates each node in the graph symbolizing a face in the enclosing surface whereas the arcs connect two nodes representing adjacent faces. It is worth noting face adjacency graphs not only track the adjacency relationship but also preserve the neighboring order of occurrence for all connections of a given node (e.g., clockwise order).

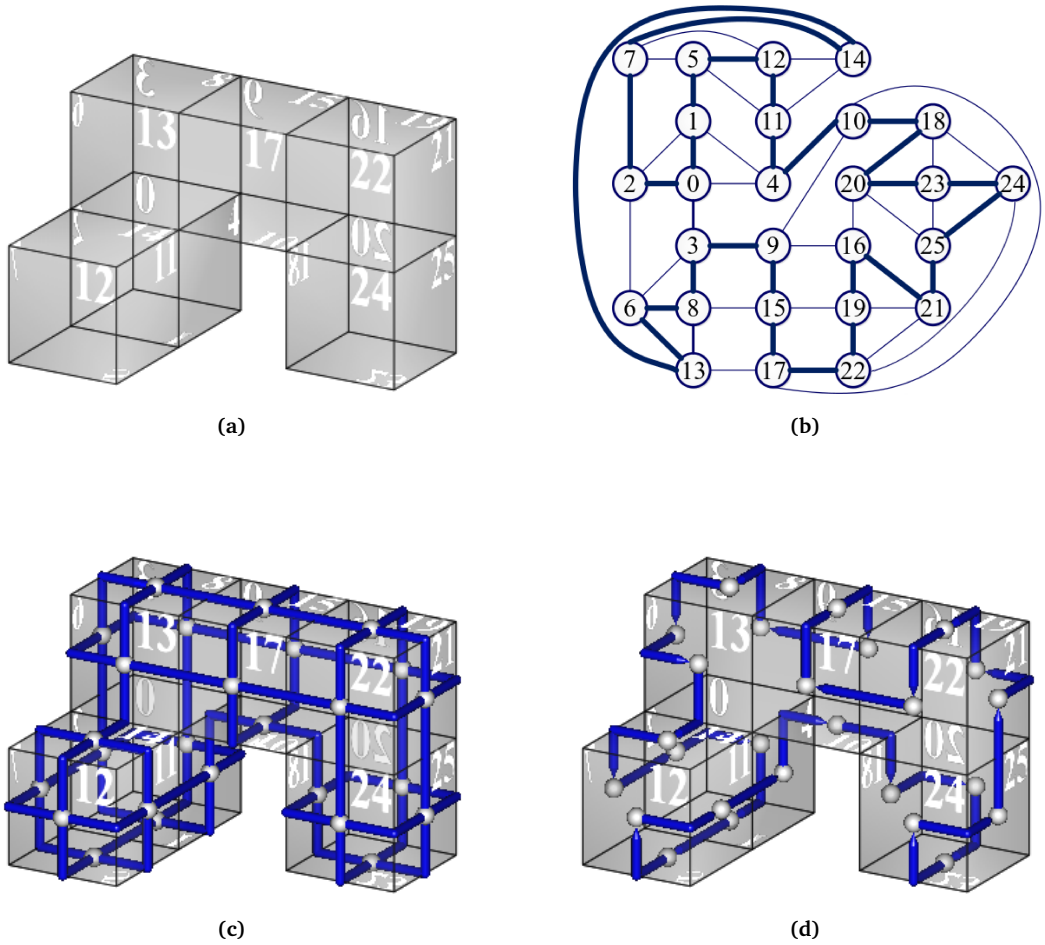


Figure 3.1: *(a) The enclosing surface of a solid made of 6 voxels and 26 external faces; (b) the face adjacency graph of the enclosing surface outlining with bold lines a Hamiltonian cycle; (c) the enclosing surface with its face adjacency graph superimposed; (d) the final representation of the enclosing surface being superimposed with the Hamiltonian cycle that could represent it.*

For the purposes here pursued, we consider that neighboring voxels of an object can be connected only by face, or by edge; meaning those that are strictly connected by vertex are not considered to be part of the same object. Thus, it is straightforward to conclude that as aforementioned the adjacency graphs for this kind of objects should necessarily be 4-regular and 4-connected provided every face in the surface has exactly four neighbors. Moreover, considering we deal with only genus-0 solids (i.e., with no holes) satisfying the condition of planarity is a direct consequence that enables, among other things, the capability to draw the adjacency graph with no crossings. Later in this chapter, these properties combined play an important in guaranteeing the existence of a Hamiltonian cycle in this specific family of graphs.

In Fig 3.1c we present the same discrete surface along with its overlapping adjacency graph to bring some intuitions on this geometric-topological duality, as well as the reasons behind the graph necessarily being a planar, 4-regular, 4-connected graph. For this graph a Hamiltonian cycle represents a sequence of unrepeated faces, progressively connected by edge and whose first and last faces are also connected. Fig 3.1b exemplify a Hamiltonian cycle using bold connectors to outline the cycle; and Fig 3.1d presents this Hamiltonian cycle superimposed over the enclosing surface.

Although in its most general form the Hamiltonian problem is considered to be NP-complete, when it comes to 4-connected planar graphs not only there is proof for Hamiltonicity [18, 21, 46] but the problem becomes computable with polynomial complexity bounds [22, 23]. In addition to the theoretical repercussion here entailed, this also implies the existence of efficient solutions to the problem that can used in practice. From the compilation of results presented in Chapter 2, Theorem 2.1 in particular is of special importance as it sets the foundations for the algorithm proposed by Chiba and Nishizeki that is explained in the next section.

Regarding the adjacency graph of objects with holes, there are formal conjectures stating the Hamiltonicity in these cases as well [47, 48]. In particular, the main conjecture in [49] states that the enclosing surface of any solid composed of voxels is Hamiltonian. Despite these last results, in this chapter we limit our method to obtain a descriptor for simple voxelizations given the lack of suitable methods to find Hamiltonian cycles in non-planar graphs with polynomial complexity bounds. Nevertheless, in Chapter 5 we extend our method to also describe objects with holes by means of a tree structure.

3.1 Hamiltonian cycle extraction: terms and definitions

This section gives an overview of the algorithm used to compute a Hamiltonian cycle in a 4-connected planar graph which in our case corresponds to the face adjacency graph of the extracted surface. The algorithm formally presented by Chiba and Nishizeki [23, 50] has an $\mathcal{O}(n)$ time complexity on the number of nodes of the graph. The algorithm is based on Tutte's [17, 18] and Thomassen's [21] proofs for the Hamiltonicity of this kind of graphs. In a nutshell, Thomassen's proof uses an inductive argument yielding a divide-and-conquer algorithm. The inductive argument

allows to recursively decompose the graph into several subgraphs, find on of the Hamiltonian paths existing in these smaller components by applying the inductive hypothesis until convergence into the trivial base case (i.e., triangle), and finally merge these individual results into a general Hamiltonian cycle embracing the whole surface.

A few additional considerations required for the core algorithm are presented below. It is worth noting that all these definitions apply for a very specific type of 2-connected planar graphs that is resulted from the algorithm's decomposition process. While the decomposition converges into more simple cases, the splits are made so it is guaranteed for the individual components to preserve the connectivity properties explained below [50].

Definition 3.1 Let $G = (V, E)$ be a 2-connected simple graph. A pair $\{x, y\}$ of nodes $x, y \in G$ is a *separation pair* if G contains two subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ satisfying

(1) $V = V_1 \cup V_2,\ V_1 \cap V_2 = \{x, y\}$, and

(2) $E = E_1 \cup E_2,\ E_1 \cap E_2 = \emptyset,\ |E_1| \geq 2,\ |E_2| \geq 2.$

Fig 3.3a shows the example of a separation pair. Given a separation pair $\{x, y\}$ we call *split graphs* of G the graphs $G_1 = (V_1, E_1 \cup \{x, y\})$ and $G_2 = (V_2, E_2 \cup \{x, y\})$, see Fig 3.3b. The arc (x, y) is called a *virtual arc* since it shall be temporarily added regardless of that arc previously existing in G, or not.

Definition 3.2 A set $\{x, y, z\}$ of nodes x, y and z in G is a *separation triple* if G has two subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ satisfying that

(1) $V = V_1 \cup V_2,\ V_1 \cap V_2 = \{x, y, z\},\ |V_1| \geq 4,\ |V_2| \geq 4$, and

(2) $E = E_1 \cup E_2,\ E_1 \cap E_2 = \emptyset$

A conclusive implication of Theorem 2.1 is that a 4-connected planar graph G always contains a Hamiltonian cycle provided the following reasoning. Let s and t be two adjacent nodes on the exterior cycle Z and let $e \neq (s, t)$ be an arc on Z, then the path P joining s and t through e must be a Hamiltonian path of G and consequently $P + (s, t)$ must be a Hamiltonian cycle of G. Thus,

an algorithm for finding the path going from s to t immediately yields an algorithm for finding a Hamiltonian cycle.

As aforesaid, regardless of the original connectivity in graph G, the subgraphs it decomposes into are not necessarily 4- but 2-connected. However, because of the way G splits, these 2-connected subgraphs inherit a property referred to as *internally 4-connected* which means they contain no separation pairs or triples in its interior nodes.

Finally, for the sake of defining a naming convention for the distinctive elements used in G during the graph decomposition (i.e., two different types of reductions) we label the most significant elements in the general case. Let G be a 2-connected plane graph with outer facial cycle Z. Let s and t be two distinct nodes on Z, and let $e = (a, b)$ be an arc on Z such that $e \neq (s, t)$. After interchanging the roles of s and t and mirroring G, if necessary, lets assume without loss of generality that $t \neq a, b$ and that nodes s, a, b and t appear clockwise in Z as shown in Fig 3.2a; note that $s = a$ is possible to occur as shown in Fig 3.2b. Let r be the node on Z counterclockwise next to s, and f be the arc joining r and s.

Definition 3.3 A separation pair is called *vertical* if either $x \in P_{sa} - s$ and $y \in P_{br}$, or $x = s$ and $y \in P_{bt} - t$.

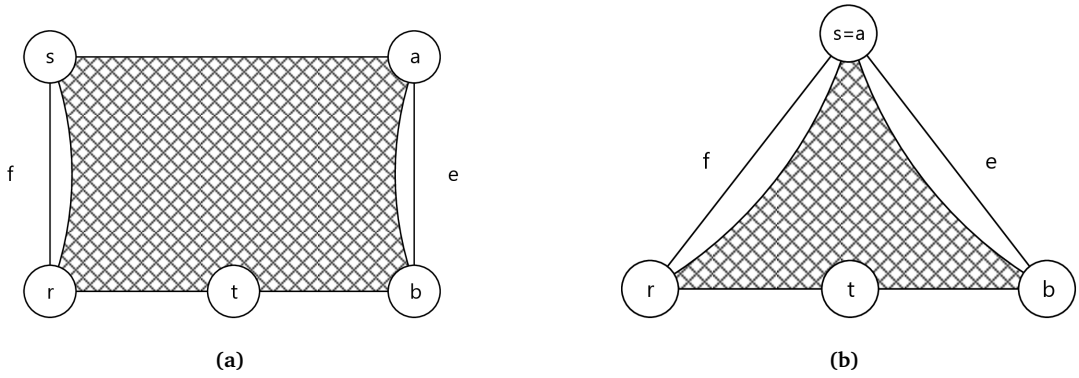


Figure 3.2: *Representation of the general configuration of a 4-connected planar graph before being decomposed into multiple subgraphs (a) when $s \neq a$, and (b) when $s = a$*

3.2 Core algorithm

Algorithm 3.1 is the core algorithm that uses a recursive function to compute one of the Hamiltonian paths that start at node s, traverses all the way to node t after visiting all the nodes of the graph, and it always considers the arc e as one of its hops in the resulting path.

Algorithm 3.1: Compute a Hamiltonian path s-t passing by arc e

```
1 Function HamiltonianPath(G, s, t, e)
2     if G has exactly three nodes then
3         return trivial path P(G, s, t, e)
4     else if G has a vertical separation pair then
5         TYPE I reduction
6     else
7         TYPE II reduction
```

During each recursion the algorithm performs one of three possible cases: a base case to compute the trivial Hamiltonian path for a 2-connected graph with only three nodes (a.k.a. triangle); and two other cases called TYPE I and TYPE II reductions where the graph decomposes into smaller subgraphs and for each of the resulting components a recursive call is made to later merge the different sub-paths. Appendix A builds from the scratch a simple example of this algorithm. In that section, the algorithm is used to compute one of the Hamiltonian cycles lying in the face adjacency graph presented in Fig 3.1.

Authors in [23, 50] state the following lemma for a planar graph G with outer facial cycle Z: let s and t be two distinct nodes on Z and let $e \neq (s,t)$ be an arc on Z. If G is internally 4-connected with respect to (s,t,e), then G has a Hamiltonian path $P(G,s,t,e)$ which connects s and t, and contains e. Moreover, if G has no vertical separation pair, then $P(G,s,t,e)$ does not contain arc $f = (s,r)$.

The proof provided by the authors to this lemma is an induction on the number of nodes of G and such a proof leads to the three cases above mentioned with $|G(V)| = 3$ being the trivial case, and $|G(V)| > 3$ one of the two reduction types. While the trivial case does not deserve further analysis, for the case of the reduction types the next subsections describe in detail each of them with all their corresponding considerations for their split and merge processes.

3.2.1 TYPE I REDUCTION: THERE EXISTS A VERTICAL SEPARATION PAIR $\{x, y\}$ IN G

Given its simplicity, this should be the preferred reduction type to perform only requires that a graph G has at least one vertical separation pair [51], see Fig 3.3a. Without loss of generality we denote the resulting split graphs as G_l and G_r and assume G_r is the one containing the arc e as shown in Fig 3.3b.

In the case of having multiple separation pairs, the of rule of thumb to pick one pair is choosing $\{x, y\}$ such that G_r has the minimum number of nodes as this will increase the chances for G_l to find another vertical separation pair during its next recursive call.

Let us distinguish from two possible subcases. First, when $t \in G_l$ as shown in Fig 3.3. Let $e' = (x, y)$ be a virtual arc. Then, G_l is clearly internally 4-connected with respect to (s, t, e'), while G_r is internally 4-connected with respect to (x, y, e), or with respect to (y, x, e) if $y = b$. Therefore by the inductive hypothesis G_l has a Hamiltonian path given as

$$P(G, s, t, e) = P(G_l, s, t, e') + P(G_r, x, y, e) - e', \text{ or}$$

$$P(G, s, t, e) = P(G_l, s, t, e') + P(G_r, y, x, e) - e'.$$

The second subcase occurs when $t \notin G_l$. Let $e' = (y, x)$, then G_l and G_r are internally 4-connected with respect to (y, s, e') and (x, t, e), respectively. Here $s \neq x$ since G is internally 4-connected with respect to (s, t, e). Thus G has a Hamiltonian path

$$P(G, s, t, e) = P(G_l, y, s, e') + P(G_r, x, t, e) - e'.$$

Note that $P(G_r, x, t, e)$ does not contain the arc (x, y) since there exist no vertical separation pair in G_r.

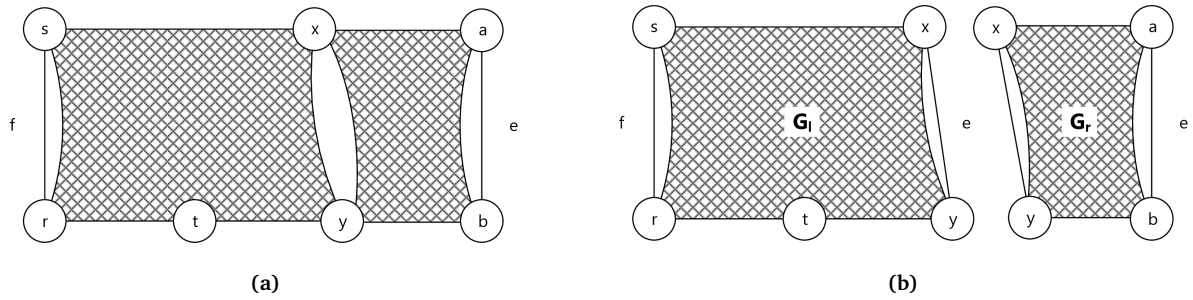


Figure 3.3: *Graph decomposition leading to TYPE I reductions. (a) Representation of a 4-connected planar graph with a vertical separation pair (x, y). (b) The split graphs G_l and G_r resulted after split in separation pair (x, y).*

3.2.2 TYPE II REDUCTION: NO VERTICAL SEPARATION PAIRS IN G

This reduction type is harder than TYPE I and it should be applied only when G does not have a vertical separation pair as illustrated in Fig 3.4a. Let C_b be the block of $G - P_{sa}$ which contains t (cross-hatched in Fig 3.4). Thus C_b must entirely contain P_{br} otherwise it would mean G contains a vertical separation pair which is a contradiction for the reason we are applying a TYPE II reduction in the first place. Repeat splitting of C_b at every separation pair such that one of the two split graphs entirely contains P_{br}. For example, in the case of Fig 3.4a, C_b is split into four components (square pattern). The component containing P_{br} is called G_b while each of the others is called G_g^2 where $g = (x, y)$ is a virtual arc contained in the component. Here we may assume the possibility of interchanging the roles of x and y so that $x \neq b$.

Now for each cutnode u of $G - P_{sa}$ contained in C_b let C_u be the maximal subgraph of $G - P_{sa}$ which can be separated from C_b at u. Let C_u^3 be the subgraph induced by the nodes of C_u and the nodes in P_{sa} which are adjacent to C_u. Let x (respectively y) be the vertex of $P_{sa} \cap C_u^3$ which is nearest to s (respectively a) along P_{sa}. Add a new arc $e' = (u, y)$ to C_u^3 if it does not exist, and let G_u^3 be the resulting graph. The graph presented in Fig 3.4a has six G_u^3 subgraphs (hatched) aligned in the top row of Fig 3.4b.

Finally, we define G_g^4 for a virtual arc g as the graph formed by merging a G_g^2 with all the G_g^2's and C_u^3's drawn above. The following is the sequence of steps:

(1) $C_g^4 \leftarrow G_g^2$,

(2) while C_g^4 has a virtual arc $g' \neq g$, iterate to merge a new G_g^2 into C_g^4,

(3) construct the subgraph of G induced by the nodes of P_{sa} adjacent to C_g^4, and redefine C_g^4 as the subgraph,

(4) $G_g^4 \leftarrow C_g^4 - x - y$.

Fig 3.4b illustrates G_g^4 for a virtual edge g. Let v (respectively w) be the node of G_g^4 that appears first (respectively last) on P_{sa}. Let w' (respectively v') be the node adjacent to y (respectively x) and appearing last (respectively first) in the outer path P_{wv} of G_g^4.

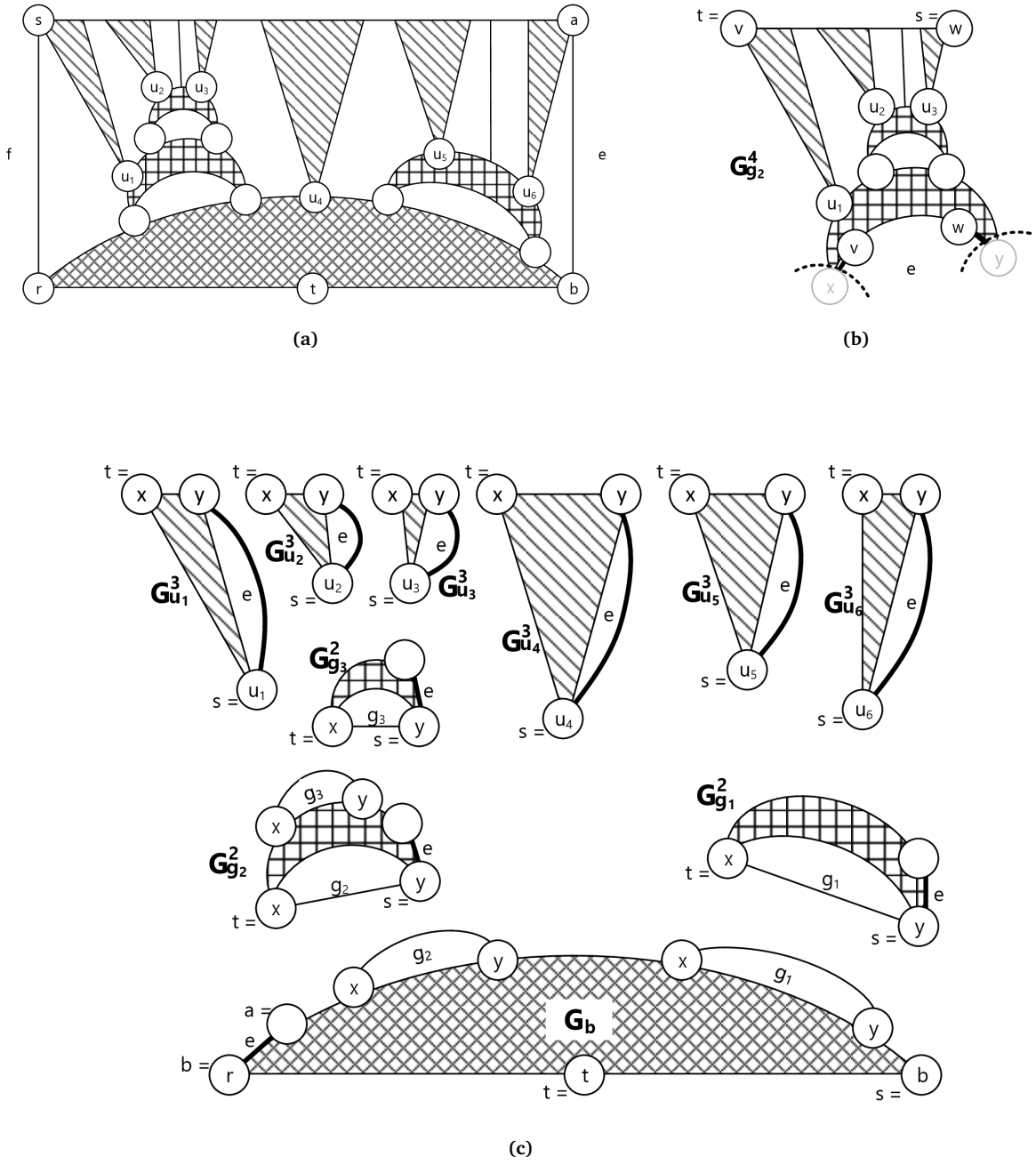


Figure 3.4: *Graph decomposition leading to TYPE II reductions. (a) Representation of a 4-connected planar graph with no vertical separation pair. (b) Construction of the subgraph $G^4_{g_2}$. (c) Decomposed subgraphs G_b (cross-hatched), G^2_b (squares pattern) and G^3_u (hatched) resulted from a graph wit no vertical separation pairs.*

Let G be an internally 4-connected plane graph containing no vertical separation pair. Then all the decomposed graphs G_b, G_g^2, G_u^3 and G_g^4 are internally 4-connected with respect to (s', t', e') if s', t', and e' are defined as follows, see Fig 3.4c.

(1) Case G_b: Let $s' = b$, and $t' = t$. If $t \neq r$, let e' be the arc clockwise incident to r on the outer facial cycle Z' of G_b; if $t = r$, let e' be the arc counterclockwise incident to b.

(2) Case G_g^2: Let $s' = y$, $t' = x$, and let e' be the arc counterclockwise incident to y on the outer facial cycle of G_g^2.

(3) Case G_u^3: Let $s' = u$, $t' = x$, and let $e' = (u, y)$.

(4) Case G_g^4: Let $s' = w$, $t' = v$. If $w' \neq v'$, let $e' = (a', b')$ be an arbitrary arc on the outer path $P_{w'v'}$; if $w' = v'$, let e' be an arbitrary arc on P_{wv} incident to $w'(= v')$.

By the inductive hypothesis used for the previous decomposition [23], all the decomposed subgraphs G_b, G_g^2, G_u^3, and G_g^4 have Hamiltonian paths. With them we can construct a Hamiltonian path $P(G, s, t, e)$ of G as follows:

(1) $P \leftarrow P_{sb} + (G_b, b, t, e')$.

(2) Extend P into a Hamiltonian path $P(G, s, t, e)$ of the whole G.

  (1) While there is a virtual arc $g = (x, y)$ in G_b repeat the following extension:

    - If $g \in P$, then let $P \leftarrow P - g + P(G_g^2, y, x, e')$ and merge G_g^2 into G_b.
    - If $g \notin P$, then construct G_g^4, and set $P \leftarrow P - P_{vw} + P(G_g^4), w, v, e'$ and delete g from G_b.

  (2) Finally, for each u of the cutnodes of $G - P_{sa}$ contained in G_b, set $P = P - P_{xy} + P(G_u^3, u, x, (u, y)) - (u, y)$.

The resulting P forms a Hamiltonian path $P(G, s, t, e)$ for the initial graph G.

4

A new chain code for the representation of enclosing surfaces

In this chapter we introduce a new chain code and some of its properties for the representation of enclosing surfaces described as a sequence of square faces. In order to use this new chain code it is assumed the sequence of faces was already obtained in the form of a Hamiltonian cycle, like the one illustrated by Fig 3.1d. In that figure it can be observed an important characteristic of the Hamiltonian cycle and one we take advantage of: the transitions between faces may only produce a finite number of orthogonal changes of direction. It is by coding these changes that we will be constructing a new chain descriptor.

Two adjacent faces c and d sharing an edge in the surface and represented by the position of their respective centers satisfy the symmetric relationship ω defined as $(c, d) \in \omega \Leftrightarrow \|c - d\| \leq 1$.

Definition 4.4 An element $a_i \in \{0 \dots 8\}$ of a chain symbolizes the *transition* (c, d) representing the orthogonal change of direction from the oriented face c to one of its ω-adjacent faces d under the following criteria:

(1) Consider the oriented face c having counterclockwise ω-adjacent faces d_j, for $0 \leq j \leq 3$. Assume we are conceptually into c by using an *inward transition* represented by the pair (d_0, c). We refer to the remaining ordered pairs (c, d_1), (c, d_2), and (c, d_3) as the *right*, *middle*, and *left* transitions, respectively.

(2) We say that a transition from c to d_j is *convex*, *flat* or *concave* when the adjacent faces c and d_j subtends an interior angle equal to 90, 180, or 270 degrees, respectively.

The assignment of values for an element a_i is as follows. We assign the value "0" to the element when referring to a right-convex change of direction with respect to the inward transition, "1" is assigned for a right-flat change, and "2" specifies a right-concave transition. Next values "3", "4", and "5" represent the middle-convex, -flat, and -concave transitions, respectively. Finally, "6", "7", and "8" are assigned to the three corresponding changes for a left-convex, -flat and -concave transitions.

Fig 4.1 illustrates these nine transitions representing the reference of an inward transition with a black segment pointing to the center of the origin face. The right, middle, and left transitions are shown in the upper, middle, and lower rows, respectively. The left, central, and right columns show the convex, flat, and concave transitions, respectively. For further reference, each configuration shows the code assigned on top of the corresponding transition.

Definition 4.5 A chain A is an ordered sequence of elements defined by

$$A = a_1 a_2 \dots a_n = \{a_i : 1 \leq i \leq n\},$$

where n indicates the number of chain elements which gets determined by the number of faces in the surface and therefore the same number of nodes in the adjacency graph.

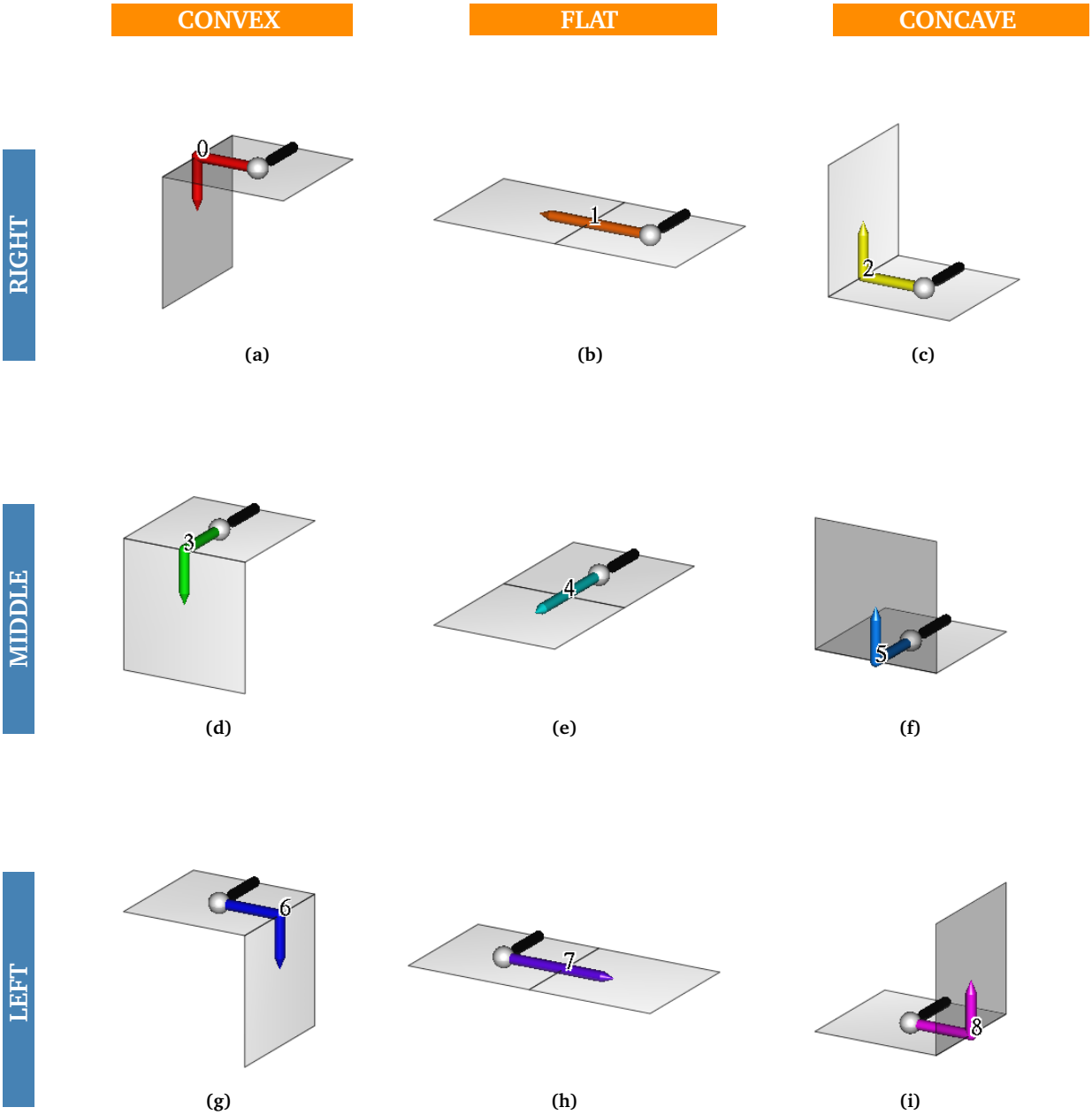


Figure 4.1: *The nine relative changes of direction for the representation of enclosing surfaces. The inward (entry) edge is denoted with a black inward segment and the back faces are shaded in darker gray for reference. In order to ease the visual interpretation of our chain code, we also use the following color code in addition to the numeric representation of the chain element for the rest of this document: (a) 0-red; (b) 1-orange; (c) 2-yellow; (d) 3-green; (e) 4-cyan; (f) 5-azure; (g) 6-blue; (h) 7-purple; and (i) 8-magenta.*

4.1 Encoding a chain descriptor for enclosing surfaces

A chain representing the discrete surface of a voxelization is the result of encoding each of the transitions in the sequence of faces fully covering surface. For the type of voxelizations initially addressed in this work (i.e., no holes), this sequence proceeds from a previously computed Hamiltonian cycle. Each transition gets assigned an element with a value in the range $[0 - 8]$ in accordance to the nine configurations presented in Fig 4.1. It is because its condition of being a cycle that we can arbitrarily select any face as the starting point for the encoding of the chain.

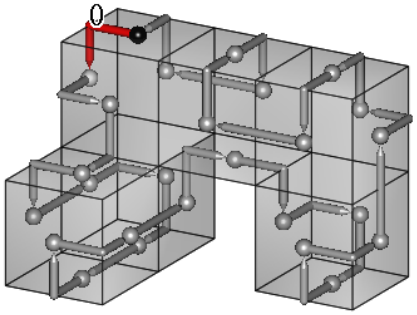
Fig 4.2 exemplifies the encoding of the cycle obtained in the last chapter and illustrated in Fig 3.1d. In order to have a visual reference of our starting point, we denote the starting element with a black node. Fig 4.2a shows the encoding of the first element of the chain corresponding to an element "0" resulting from a right-convex transition. In Fig 4.2b we set the next element of the chain as "6" to describe a left-convex transition towards the following face. Figs 4.2c-e illustrate the step-by-step encoding of the next three elements of the chain while Fig 4.2f presents the final chain fully encoded.

Our method establishes a univocal correspondence between the geometrical realization of the surface and the chain code that describes it. Not only does this imply there is always a unique chain representing a given sequence of faces but also that we can recover this geometric realization directly from the chain descriptor with no place for ambiguities.

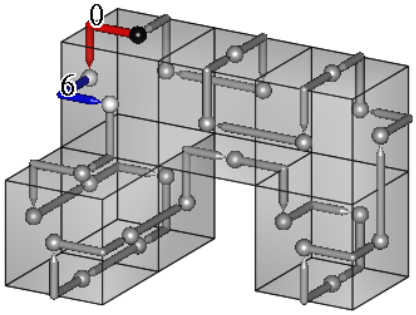
From a given chain, the process to recover the enclosing surface requires a direct face-by-face reconstruction of the surface as we walk through the chain elements. This recovery mechanism is straightforward and does not require further analysis since it can be easily derived as the inverse process to Fig 4.2.

4.2 Independence of rotation and translation

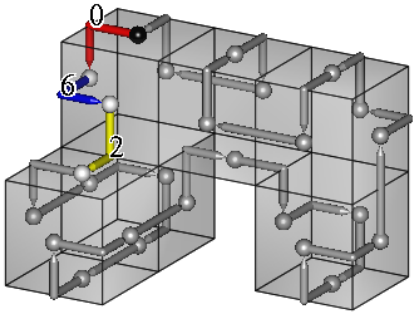
Because its construction only depends on the relative changes of direction, our chain code is invariant under rotation and translation of the represented surface. This means the same chain descriptor can be used to describe a given object represented by its discrete surface regardless of its position or orthogonal rotation in world coordinates.



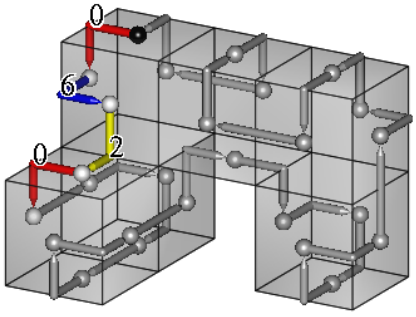
0...
(a)



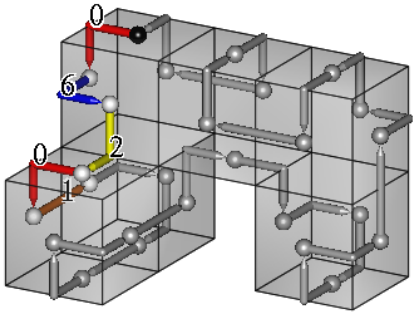
06...
(b)



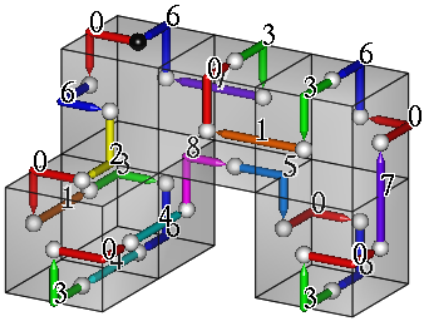
062...
(c)



0620...
(d)



06201...
(e)



062013643048506307063103 76
(f)

Figure 4.2: *Encoding the chain of a solid composed of 26 faces: (a)-(e) step-by-step encoding of the first five elements of the chain; (f) final representation of the enclosing surface encoded as a chain code (starting face is denoted with a black node)*

4.3 Inverse of a chain

The inverse of a chain A is another chain A^I of the same length that is obtained by traversing the same cycle in the opposite direction. The i-th element of the chain A^I is computed as follows:

$$a_i^I = a_{(n-i+1)} \bmod 3 \; - \; 3[a_{(n-i+2)} \operatorname{div} 3] \; + \; 6,$$

where n is the number of chain elements (i.e., number of nodes in the adjacency graph); *div* and *mod* are both functions used respectively to obtain the quotient and remainder of an integer division. It is worth to note by encoding cycles that we are allowed to use modular arithmetic to calculate an element subindex. This means a_{i+1} would represent the first element of the chain whenever i is equal to the index of the last element. Alternatively, a_{i-1} would be the last element when i refers to the first chain element. As an example, in Fig 5.3b we illustrate the inverse of a chain describing the surface in Fig 5.3a.

In a similar fashion, we can also recover the original chain A by means of applying the same transformation to invert back a chain A^I that was previously inverted.

4.4 Normalized representation (independence to the starting point)

Our chain descriptor uniquely describes a surface meaning there are not two different voxelizations producing the same chain code. In an opposite way, a given model effectively could be represented by multiple chain descriptors. In fact, the number of different descriptors is equal to the amount of Hamiltonian cycles that can be found in the enclosing surface. Moreover, each cycle can itself be encoded by a number of different chains depending on the selection of starting point and direction used to encode it. Certainly to guarantee the first statement is a sufficient condition for any application aiming to an accurate and compact representation of 3D objects, but when it comes to more advanced applications (e.g., object recognition and analysis) a 1-to-1 correspondence between the object and its descriptor is necessary.

By utilizing common properties of a chain code, our descriptor can in theory be normalized as many other chain codes in the specialized literature ensuring a biunivocal correspondence

between the chain and the object it represents. As an example, the normalization can select among all the possible chains representing the surface that chain whose sequence of elements adds to the minimal integer. This search would consider every possible starting point for every existing cycle, traversed in both directions. Fig 5.3c shows the normalized chain for the solid displayed in Fig 5.3a; this chain retrieves the minimum integer that can be represented by any cycle over the surface.

Although the normalized representation implies an exhaustive search over every possible cycle that is computed in the surface, this property enables our chain code to be invariant under the starting point. Other approaches more suitable for specific situations can be instrumented instead. For example, minimizing the number of convex transitions, reducing the tortuosity (i.e., maximize the number of "flat" transitions), or optimizing the number of certain occurrences in favor of a maximum compression rate.

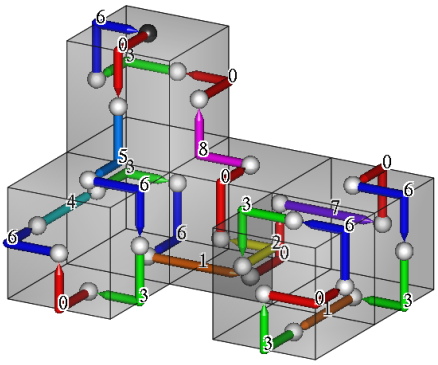
Unfortunately, because of the exponential increment in the number of cycles as the number of nodes of the graph grows, this is not always a feasible solution. In fact, one may like to avoid this normalized representation and the computation of multiple Hamiltonian cycles when dealing with relatively large objects. Still, for a given cycle it could be valuable in some cases to normalize the chain with respect to the starting-point with a simple search for the minimum integer within all possible circular-shifts traversed in both directions.

4.5 Invariance under mirroring transformation

Our proposed descriptor for discrete surfaces may be made invariant also under mirroring transformation by means of the following definition.

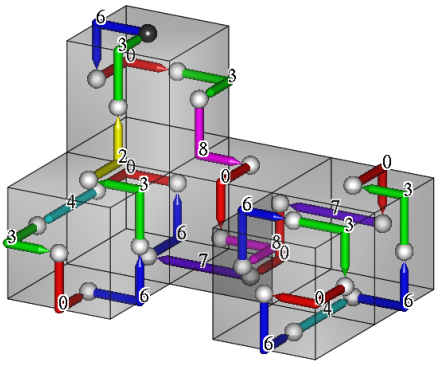
> **Definition 4.6** The chain of the mirror of a solid is another chain (termed *mirroring chain*) whose elements "0", "1", and "2" are interchanged with elements "6", "7", and "8", respectively.

Fig 5.3d illustrates the mirror transformation applied over the chain presented in Fig 5.3a. This makes possible to directly pass from the chain describing a given surface to the chain describing its specular image; and vice versa throughout the same definition.



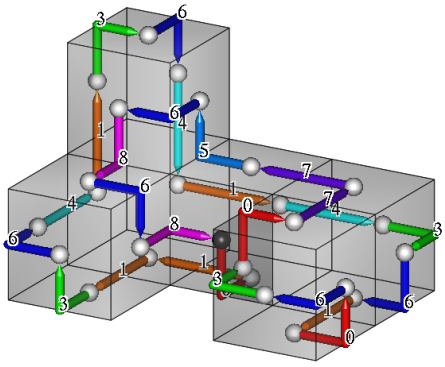
056306436107063130632080 36

(a)



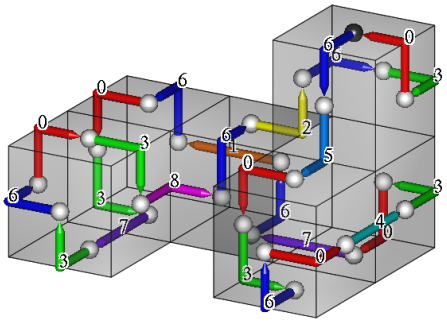
603808630646307076043063 23

(b)



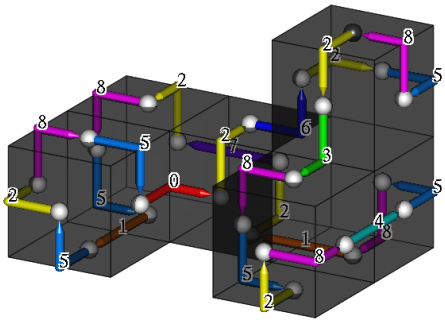
011364136414361063077568 68

(c)



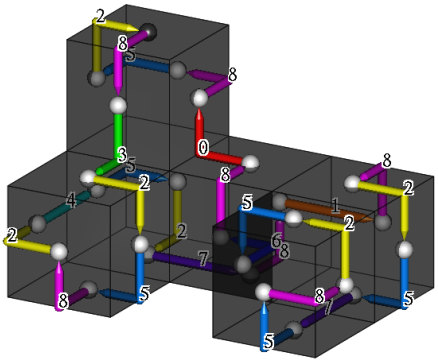
650360430761603736038626 30

(d)



238528458127285152850262 58

(e)



836082452781825758256808 52

(f)

Figure 4.3: *Some transformations directly applied to the chain descriptor of the surface presented in (a): (b) inverse of the chain obtained by traversing the same cycle in the opposite direction; (c) starting-point invariant representation of the surface chosen so as the sequence of elements retrieves the minimum integer; (d) mirror of the surface; (e) complement of a surface (back side of faces is shaded in dark color to differentiate from the front side); (f) mirror of the complement to illustrate the wrapping surface.*

4.6 Invariance under complement transformation

Since we consider the type of surfaces our chain code represents as oriented surfaces, our proposed code must necessarily distinguish between the face's front and the back side. By means of a simple transformation, our proposed chain code allows reversing the orientation for each face in the surface so our chain code may be made invariant under what is referred to as the "complement" transformation. It is defined as follows:

> **Definition 4.7** The complement of a surface's chain is another chain (termed *complement chain*) whose elements "0", "3", and "6" are interchanged with elements "2", "5", and "8", respectively.

The name of this transformation comes after the idea of representing the enclosing surface of a different voxelization that complements the space in the rectangular grid where the original chain was obtained. In other words, for a dataset of voxels discretized with values 0 or 1, this transformation conceptually means we pass from describing the enclosing surface of the *1-value* voxelization to describe the *0-value* voxelization complementing the space, and vice versa.

Fig 4.3e shows the complement surface for the solid shown in Fig 5.3a. In this illustration we denote the back side of the oriented faces with a dark shading to differentiate them from their front side.

One handy application of the complement transformation is to obtain the wrapper of a solid by means of computing the mirror of its complement. In Fig 4.3f we show what would be the surface of the voxelization wrapping the solid that is presented in Fig 5.3a by means of mirroring its complement shown in Fig 4.3e. From a Computer Graphics standpoint for instance, this would be the geometric representation of the same surfaces after reversing its normal vectors.

5

Representation of voxel-based surfaces by means of a tree-structured chain code

In the previous chapters a new method was presented for the representation of enclosing surfaces. This method encodes one of the Hamiltonian cycles laying in the face adjacency graph to produce a one-dimensional descriptor with nine-symbol alphabet, see Chapters 3 and 4. Nonetheless, the main limitation of this method is that it can only be used to describe the enclosing surface of objects with no holes since the Hamiltonicity proof and existence of efficient algorithms to compute those sequences is only available for graphs such as the ones of closed surfaces with topological genus 0.

In this chapter we extend our method for the representation of any discrete surface regardless of its genus and whether being closed or not. The new description takes advantage of a different

structure in the form of a tree that is also extracted from the face adjacency graph. We refer to this new descriptor as the *surface tree*.

The encoding of a surface tree uses the same nine-symbol alphabet as well as a parenthesis notation to group its branches. Although this method is suitable to represent trees of any degree, for the particular use given in this work the maximum degree of a tree will be four.

Our surface tree method also represents a lossless and compact way to describe any discrete surface while preserving the geometrical information. In fact, a surface tree provides similar properties to the Hamitonian descriptor including invariance under translation and rotation. Furthermore, there are equivalencies to also permit performing some of the morphological transformations like mirroring and complement transforming; as well as flipping the surface by directly applying simple grammatical techniques to the chain.

5.1 Construction of a surface tree

In order to compute a surface tree we assume that the boundary surface is readily available and that is composed of a set of orthogonal square faces representing the intersection of 0- and 1-valued voxels in a scene. Every face is represented by a node visually placed on its centroid and we say that two faces c and d in the discrete surface are adjacent whenever they share an edge. In other words, faces c and d are adjacent when both faces belong to the binary symmetric relationship ω defined as $(c, d) \in \omega \Leftrightarrow \|c - d\| \leq 1$.

Surface trees describe any boundary surface by encoding one of the spanning trees in the face adjacency graph. Any of the popular algorithms used in the literature to compute the *Minimum Spanning Tree (MST)* [25, 26] could be used to find the tree structure. In its most general form and assuming no additional optimizations are made, this family of algorithms converge to time complexities in the order of $\mathcal{O}(n \log n)$ where n is the number of nodes of the graph. Although these bounds imply an acceptable solution to many of the real-life problems, it is true they also add an unnecessary overhead to the problem because of the algorithm design not only aims for a tree but for an optimal one whose arc connections add up to the minimum weight assuming dealing with a weighted graph.

Below we propose a different algorithm to compute surface trees that directly derives from the MST algorithm family but instead of doing any arc-weight considerations, our algorithm spans in a greedy fashion improving the time bounds to the order of $\mathcal{O}(n)$. It is also important to note that as the tree spans, the encoding of its different orthogonal transitions can be made by using the same table of possible transitions presented in Fig 4.1.

Given an arbitrary face s in the surface and a transition (r, s) representing an initial transition from face r to the starting face s, the algorithm's output provides a list T of encoded transitions depicting the surface tree. This algorithm starts by adding to T the available transitions going from s to all its ω-adjacent faces that have not been visited yet. Therefore, in every iteration the face connected by the incoming transition gets excluded of being re-added. The expansion of the tree occurs by repeating the same step for each of the ω-adjacent neighbors until reaching completion determined by having all faces in the surface marked as visited.

During the execution of the algorithm a few auxiliary structures are used to track the preliminary state of the tree. These include a queue Q of inward transitions and a set V of flags that tracks the list of visited faces to prevent re-entrance.

Algorithm 5.1: Surface tree constructing algorithm

```
1  Function SURFTREE(s, (r, s))
2      Q ← ∅, V ← ∅, T ← ∅ ;
3      Q ← Q ∪ (r, s) ;
4      V ← V ∪ s ;
5      while Q ≠ ∅ do
6          (b, c) ← Extract(Q) ;
7          C ← {(c, d) | (c, d) ∈ ω, b ≠ d} ;
8          while C ≠ ∅ do
9              (c, d) ← Extract(C) ;
10             if d ∉ V then
11                 T ← T ∪ [(c, d) : Code (b, c, d)] ;
12                 Q ← Q ∪ (c, d) ;
13                 V ← V ∪ d ;
14     return T
```

Fig 5.1 illustrates the execution of the algorithm to obtain the surface tree for a given enclosing surface. Firstly, face 8 is arbitrarily selected as the starting node using the inward transition $(3, 8)$ as the initial reference, see Fig 5.1a. The logic in lines 2-4 of Algorithm 5.1 takes care of pushing

this transition into the queue Q and signaling face 8 as a visited face in the set of flags V. After these initialization lines, Q is not empty anymore so the condition in line 5 should be satisfied the first time. Next, lines 6-7 are executed to pop out transition (b, c) from the head of the queue and use it as the new inward transition for the next face to be processed. For visual reference, Fig 5.1 highlights with a yellow sphere the centroid of the next face to be processed.

In line 7 the set C of transitions connecting c to its ω-adjacent faces is created. Fig 5.1b illustrates the extraction of transition $(3, 8)$ from Q and how it follows creating the set C of transitions $(8, 6)$, $(8, 13)$, and $(8, 15)$.

Lines 8 to 13 check whether the neighbors of every element in C have been visited. In case such of one faces has not been visited, the face is marked as visitied in V at the same time the corresponding transition is pushed into Q and added to the list T. For our example, after this iteration the state of the different structures should be as follows: T contains transitions $[(8, 6) : 0]$, $[(8, 13) : 3]$, $[(8, 15) : 7]$; Q has $(8, 6)$, $(8, 13)$, $(8, 15)$; and the faces 6, 13, and 15 are marked as visited in V. Later, the algorithm loops to Line 5 in order to process the next iteration, which corresponds to face 6. This process keeps repeating the expansion of whatever is the current face until Q is empty, see Figs 5.1c-h.

It is worth to note that in line 11 the items in T keep track not only of the transitions but also of the numeric representation it gets assigned through a lookup to the table of possible configurations. The method *Code* is responsible for returning the appropriate code given a transition (c, d) and its inward reference.

Finally, Fig 5.1i shows the adjacency graph for the solid used in Fig 5.1 outlining with thicker lines the spanning tree obtained in Fig 5.1h.

The process to recover a boundary surface from this tree descriptor is relatively straightforward by simply traversing the spanning tree in either a breadth-first or depth-first fashion. This process consists of a face-by-face reconstruction of the surface as indicated by the numeric value of each one of the transitions. Note that different selections of the face used as the root and the initial inward transition used as a reference will produce different spanning trees. In Section 5.3 we provide a brief insight for the normalization of the tree descriptor.

5.2 Codification of a surface tree as a chain

After obtaining the tree descriptor an additional process is required to encode a surface tree into a chain. One of the simplest ways to represent trees as a chain is the well-known parentheses notation [52]. By using this notation we establish a simple correspondence between the different branches of the tree and the different blocks delimited by nested parentheses. In order to exemplify this process, we demonstrate the codification of the surface tree obtained along the different steps presented in Fig 5.1.

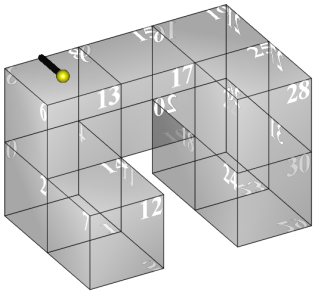
Starting with Fig 5.1b, the arbitrarily selected root face is expanded to get the chain (0)(3)(7). This chain describes the three different transitions available to move from the current face 8 to neighbor faces 6, 13 and 15, respectively. Although nothing prevent us from encoding the neighbors in any order, note that we notation follow the counterclockwise convention to be more intentional in the final resulting chain.

We continue encoding expanding face 6 as shown in Fig 5.1c. After this expansion the preliminary chain updates to be (0**(0)(4)**)(3)(7). Subsequently, after expanding face 13 as presented in Fig 5.1d we end up having the chain (0(0)(4))(3**(5)(7)**)(7). Later, expansion of faces 15, 3, and 2 shown in Figs 5.1e-g produce chains (0(0)(4))(3(5)(7))(7**(4)(6)**), (0(0**(7)**)(4))(3(5)(7))(7(4)(6)) and, (0(0(7))(4**(3)(7)**))(3(5)(7))(7(4)(6)), respectively, and so on.

The final version of the surface tree fully encoded into a single chain as shown in Fig 5.1h is (0(0(7(0)))(4(3(7))(7)))(3(5(3)(6))(7(0(8(3)))(5(1(3)(6))(3))))(7(4(1(6(4)))(3(4))(6(4)))(6)).

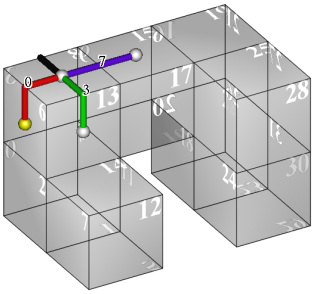
In order to test the robustness of the surface tree description, Fig 5.2 depicts the process of producing the surface tree of different types of discrete surface: an enclosing surface, an open surface, and another open surface with holes are tested.

Firstly, in Figs 5.2a-d we show the results of applying the SURFTREE algorithm (Algorithm 5.1) to compute a tree descriptor for a solid made of 11 voxels. The resulting tree shown in Fig 5.2d yields the following chain:
(0(0(7))(5(0(3(7))(7))(4(3(7(4)))(7(4(3))))(7(3(2(6))))))(3(4(2(6(4)))(3(7))(8(0(7))(4(3))))
(7(2(4(6))(6))))(7(3(4))(6(4))).



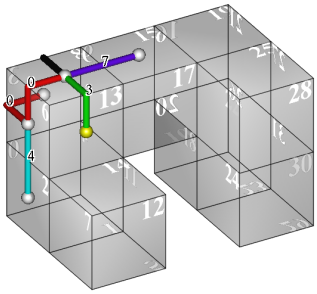
$Q \rightarrow (3,8)$

(a)



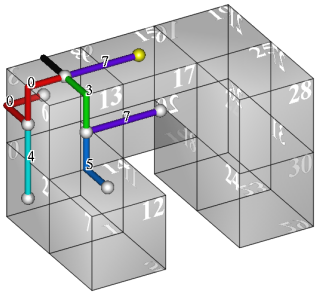
$Q \rightarrow (8,6)\,(8,13)\,(8,15)$

(b)



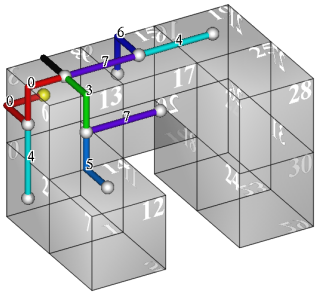
$Q \rightarrow (8,13)\,(8,15)\,(6,3)\,(6,2)$

(c)



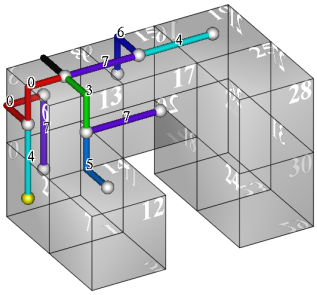
$Q \rightarrow (8,15)\,(6,3)\,(6,2)$
$(13,14)\,(13,17)$

(d)



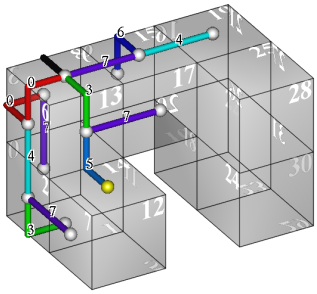
$Q \rightarrow (6,3)\,(6,2)\,(13,14)$
$(13,17)\,(15,19)\,(15,9)$

(e)



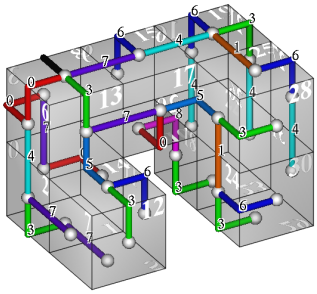
$Q \rightarrow (6,2)\,(13,14)\,(13,17)$
$(15,19)\,(15,9)\,(3,0)$

(f)



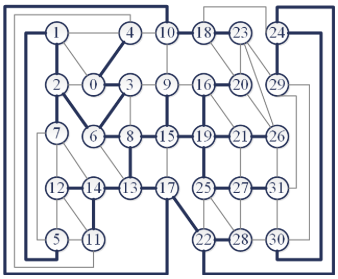
$Q \rightarrow (13,14)\,(13,17)\,(15,19)$
$(15,9)\,(3,0)\,(2,1)\,(2,7)$

(g)



$Q \rightarrow \emptyset$

(h)



(i)

Figure 5.1: *Construction of the surface tree for a solid made out of 8 voxels. (a) Root selection; (b)-(g) Step-by-step expansion of the first few nodes. Below each subfigure we show the state of the queue Q of inward transitions whereas the values in set T are directly displayed in the picture on top of the corresponding transition; (h) Final spanning tree. (i) Face adjacency graph outlining with thicker lines the spanning tree obtained in this figure.*

Then, for the open surface in the second row of Fig 5.2 we illustrate the process of obtaining its surface tree in the sequence of Figs 5.2e-h. The surface is made of 25 square faces and its final surface tree gets encoded as follows:
(0(0)(5(0(7))(4(7(4)))(7(3(2)(5)))))(3(4(2)(5(4))(8(4)))(7(2(4))))(7(6)).

Finally, in the third row of Fig 5.2 we are describing a multi-holed open surface. We illustrate in Figs 5.2i-l the process to compute this surface tree resulting in the following chain descriptor:
(0(0)(5(0(7))(4(7(4(6(5)(8(3))))))))(7(0(5(4(0(5(1(5)))))))(6)).

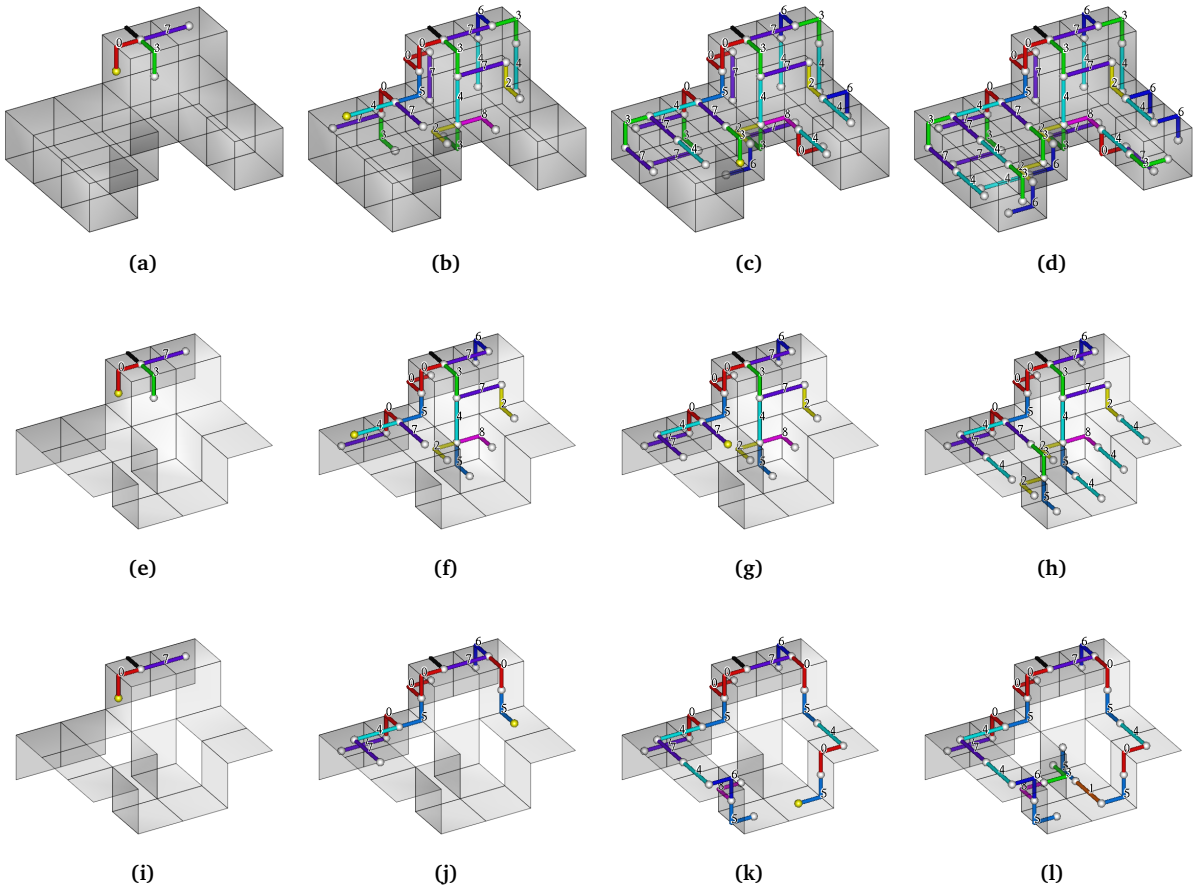


Figure 5.2: *We show the process of producing the surface trees from: (a)-(d) the enclosing surface of a simple solid composed of 11 voxels and 42 oriented faces; (e)-(h) a non-closed surface obtained from a subset of 25 faces of the previous surface; (i)-(l) a multi-holed surface made of 22 faces resulting from removing three faces of the previous non-closed surface.*

5.3 Some properties of the surface trees

The surface tree mechanism provides a number of geometrical and topological properties about the surface it describes. Analogous to our Hamiltonian surface representation, by means of these attributes the analysis and morphological transformation of surfaces is also allowed directly from its compact representation as a chain. Throughout this section these properties are individually outlined.

5.3.1 Independence to rotation and translation

The construction of a surface tree is based on the codification of relative changes of direction between adjacent faces orthogonal to each other. Because the codification of any transition is not relative to a global frame or reference, then the description of surface trees is invariant under rotation and translation.

5.3.2 Length of the chain descriptor

A surface made of n faces yields a chain whose length L is computed as

$$L = 3(n - 1). \tag{5.1}$$

This formula states a linear mapping between the number of faces in the surface and the number of elements in the tree only excluding the root face that is used for reference purposes. The factor of 3 in this mapping comes from the overhead added by the parenthesis notation.

Proofs for this and other theorems in this section are not included but easily derivable from their definitions.

5.3.3 Mirror transformation

One of the handy properties of surface trees is allowing to easily compute a mirror surface (i.e., specular image) by means of the following definition.

Definition 5.8 Let T be a surface tree, T^M (termed as the mirror of T) is the tree descriptor obtained by swapping elements "0", "1", and "2" with "6", "7", and "8".

Fig 5.3b illustrates the mirror surface obtained from the surface tree of Fig 5.3a.

5.3.4 Complement transformation

In the context of surface trees we refer to the result of inverting convex and concave transitions as the *complement* of a surface. We explain such a process by means of the following definition.

Definition 5.9 Let T be a surface tree, T^C (termed as the complement of T) is the descriptor obtained by interchanging in T every occurrence of elements "0", "3", and "6" with elements "2", "5", and "8", respectively.

As an example of this property we illustrate in Fig 5.3c the complement transformation of the surface presented in Fig 5.3a. Note that in this figure we show back faces shaded darker to make distinction from front faces.

5.3.5 Flip transformation

Motivated by the idea of dealing with oriented surfaces, we define an operation to easily flip the orientation of the faces as follows:

Definition 5.10 For a surface tree T we obtain the *flipped* surface T^F by means of the following mapping between elements $a_i \in T$ and elements $a_i^F \in T^F$

$$a_i^F = 8 - a_i. \tag{5.2}$$

The following theorem is a direct consequence of the previous definitions:

Theorem 5.1. *Functions used to apply the mirror, complement and flip transformations of a surface tree are all involutions.*

The above theorem states these three morphological functions are their own inverse so the following statement is always true for f being any of the mirror (M), the complement (C), or the flip (F) transformation functions:

$$T = (T^f)^f. \tag{5.3}$$

Theorem 5.2. *By combining previous results, we can alternatively flip a surface tree T as follows*

$$T^F = (T^C)^M = (T^M)^C. \tag{5.4}$$

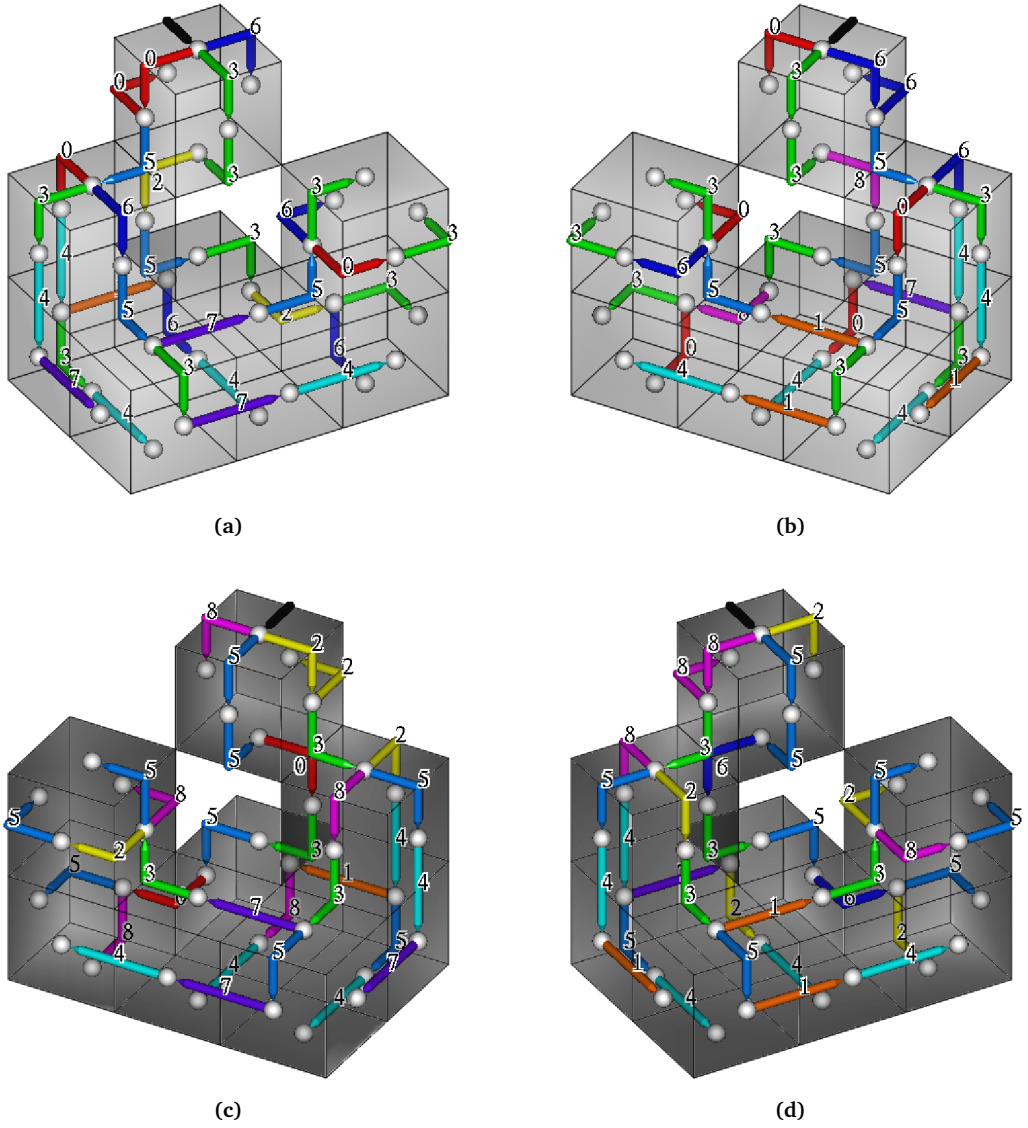
In Fig 5.3d we exemplify this transformation by flipping the surface presented in Fig 5.3a.

5.3.6 Origin of construction and uniqueness of the descriptor

Even when a boundary surface can be described by several trees, any of these trees uniquely describes a surface. This concept is analogous to our Hamiltonian method where two different surfaces may never produce the exact same descriptor. On top of this condition rely many applications such as the compression and lossless representation of surfaces. However, in other cases a normalized description is required so the surface is uniquely represented by only one tree.

Potential to select any node as the root of the tree for our method derives in the undesired outcome of having multiple possible descriptors. Similar to other methods in literature [1], the origin of description of our tree has a direct effect on the final descriptor obtained. Some applications require to have a unique origin that can be easily identifiable after slightest variation to the surface. While this is a topic of high complexity, this section offers some insights with the intention to hint some of the alternatives for the origin-normalization problem. However, we are aware better alternatives exist depending on the specific application these target.

Based on the approaches taken by other works [1, 53], our proposal to the root selection is to choose that face whose centroid is closest to the geometrical center of mass of the represented solid. Additionally, the initial transition used as a reference may be chosen such that it is maximally aligned to the major axis used during the computation of the eccentricity of the solid [54]. This major axis can be determined as the segment of line between the centers of the two faces furthest apart. Additional criteria must be used for tied situations.



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

Figure 5.3: *Example of some basic transformations directly applied to surface trees: (a) the surface tree of a simple voxelized solid, (b) the mirror surface obtained by interchanging* $0 \leftrightarrow 6$*,* $1 \leftrightarrow 7$*, and* $2 \leftrightarrow 8$*, (c) the complement surface obtained by replacing* $0 \leftrightarrow 2$*,* $3 \leftrightarrow 5$*, and* $6 \leftrightarrow 8$*, and (d) the flipped surface by the symmetrical and bijective mapping* $f(a_i) = 8 - a_i$ *which swaps* $0 \leftrightarrow 8$*,* $1 \leftrightarrow 7$*,* $2 \leftrightarrow 6$*, and* $3 \leftrightarrow 5$*. Note that back faces are shaded in darker gray to be distinguished from front faces.*

6

Experimental results of the proposed methods

In this chapter we present the results of applying the methods proposed in the previous chapters to a set of binary voxelizations and discrete surfaces that are used as the input. The objects used here for experimentation proceed from different sources to prove the efficiency of our descriptor for the loosely representation of different types of discrete surfaces. This chapter is organized so that the results for both methodologies introduced in Chapters 3 and 5 are individually analyzed following the same order of appearance in the upcoming sections.

The examples used along this chapter draw upon the representation of real and artificially created objects. In all cases, objects passed through a discretization process whose output is stored in any of the specific file formats designed to describe discrete volumes. Note that the resulting three-dimensional representation not necessarily has to end up with a binary formatted dataset but an additional process may be necessary to generate datasets that map to $\{0, 1\}$. Thresholding is

possibly the simplest and the most commonly used method applied to real-object scans such as those proceeding from medical imaging scans. A thresholding process usually implies some level of user interaction to manually select the best value to generate the expected visualization.

As aforementioned in previous chapters, it is also important to emphasize that our proposal to represent 3D objects centers specifically in the lossless description of their enclosing surfaces. While this kind of surface may be explicitly generated or, somehow, alternatively upfront provided, in most cases these will require a surface-tracking extra step to compute the surface (e.g., using Artzy's algorithm [43]).

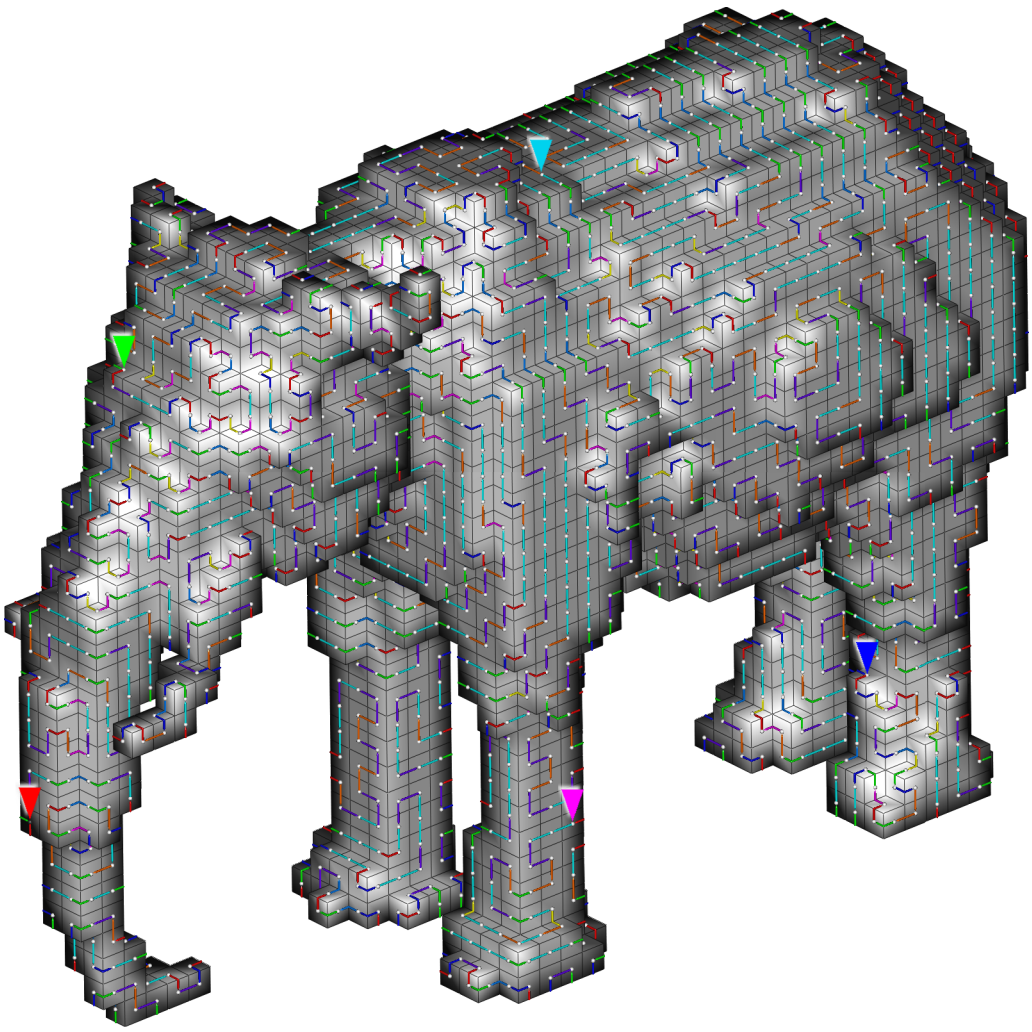
6.1 Hamiltonian-based description of simple voxelized surfaces

6.1.1 Representation

Below we show some examples of the representation of large objects by means of our proposed chain code encoding a Hamiltonian cycle in the enclosing surface under the assumption that these objects contain no holes. We first present in Fig 6.1 the encoded surface for the model of an elephant made of 3 990 faces. From its face adjacency graph, which has the same number of nodes, we compute and then encode the Hamiltonian cycle [23].

For the example in Fig 6.1 in particular, we display both the chain descriptor over its enclosing surface and the numeric representation of the chain. Additionally, this figure also uses a five color label system to assist the reader in tracking different parts in the chain descriptor that is superimposed to the discrete surface. Any of these color labels marks both an specific face and the part of the chain which encodes that part of the surface.

In Fig 6.7 we present descriptors for a few other solids proceeding from different sources. Starting at the top of the figure, there is the model of a horse composed of 12 384 faces. In the bottom-left corner of that figure we illustrate the chain descriptor for a voxelization of the Venus de Milo model whose enclosing surface is composed of 66 036 faces. Finally, at the bottom-right corner, we present the scanned model of a real tooth obtained by CT and then it was binarized by thresholding to an appropriate value representing the tooth surface. This enclosing surface is made of 56 680 faces.



▸072686831053165141341674643437444760473534701812783743130744135604477018748305131447744473447113204434414432447745371576448
438260652770771141441658348608260507243628353717530514502830535365735347417608432014534530580423280283 ▸07446058028483058064181
812416514674118704116735271144478606512336105357826458076082348646060780206347320430631040362406528134281283058307652605064653050280148738473534860711406211 ▸435348244645371721364328421673104771148738231177474372142344414774076444382711428043284781274
430530744443764443827471141444280432847812744438353610513206206202804444441477438644444382682686086086086826053058020620620620611802802744480207260762844371860771614407171623835353535371182605406206206153531741717177171435417305804771176580411440537
604874701404431578410278064165 6 ▸062801137652361130836437117114343546747126447417740443240604074473506308062344684467413534430827401473537467231076107608283445305834535465811718640618353543535345305132802802805353442168136431307376106061077254172064651440623577835344506313070632417438273535353482613083280280280280534441741743281354714407427812317747351234314478644537446077451144502434431802446516747604537160771441473050632814216147017477143764016583401875160547704117711744457304358064415127376280411434580653411353765313083628351236523652365236526826041758304135830583058344444382744062812744743812734706527175354482747147434353744453471474237471172735065417446107765448261744171707435247414705444065232780414723545352707405343441534444428044444171715260835360581274443544446087343574078275372473437283075648274077118360507176058231362453376308477415027450271707432417162608353606581274443544446087343574078275372473437283075648274077118360507176058231362453376308477415027450271707432417106231813823530762416062802061445344453444145124743544460844608681267544471283805707444270717608265827382373620206205703621771177470140 ▸44771131774734214144244364447344434432802023647317603437414376106530760641411706461434017744477404431401747717711771477404434141
314735347444774405310574870177044314017475830610860771452805362027174440620471471450431823615061428042374114318435382738273443447147147174438268417658045344417147147143534534280423741137215684353543534714714717460844774453075280617147028042805341135435437147160847444477444153470580153450611054460865165060874771705705345305411760868246582430584731036241167461041477447431076471141437611063447643047044144337606110653057780141170361614017683205237441145706564406447447704443140174771177117747440443414114647477116107610578177308260765183204524386580610774428044158014741738271354354062061453443540623453418236144471406244407241183610745361444814654345353443827353535345372738273826517435477534171741740724353535350614435354345146144406241144274481473453560743543535353483610717417165477502804280274053535345343543431471074442310744564753543543535353574071741765805607254053506153177054351441745244428064564383543574071765802740453444144114743547355807234113836177123086711747711411672174445317177054351361130841645643538444457344483181645345345063084714444064561460847381273471386441582644075816417382682360483170583444445360581471354474053652735823618276448275817143544280516415344714444444170723435341741544424434480534341482647414444471443545416258147135447453611742305783064271044524465371523406061173043760582684176118344826741834534471444447314814643543447454457416826435860717444447144357308735482648164770281355318351446144407171432741454134280414615371444447135432036216058024505404432150611308474382744534831753614826074461717147382684652644717174814446423853537362354184144460535344475234265348171274471407232814534415315371582607446171714738268465264471717481444642385353736235418414446053534447523426534817127447140723281453441531057382756078361444386147165826444717444841860773620745441712781171471444245345056308117435474354604868126860774536474204323534206117430443868264474771431410277580114057417406145374147460835348236058353482673284830506213280506153083424406260624041176075607717608443486487044450243515264806082775804313074375760623043656106061071860760611307651723618131827344341137761060610760782774183816048678160424041178136432052365711744444417017365464361801137184076060776061060737173653053058045364144441173

Figure 6.1: *Chain descriptor for the model of an elephant made of* 3 990 *faces thus the same number of chain elements. We use five color labels to help following part of the chain.*

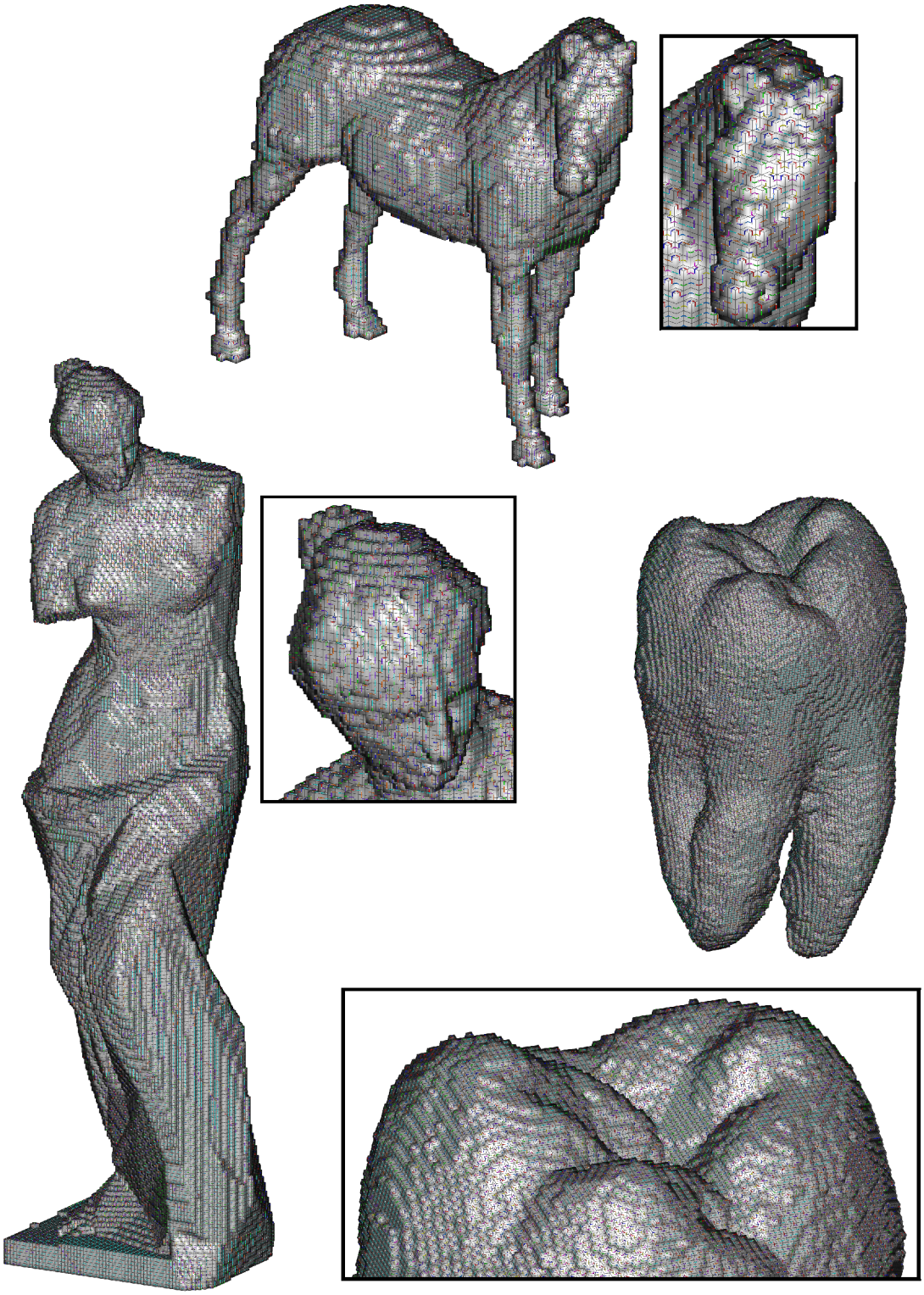


Figure 6.2: *Chain descriptor of three models: (top) a horse made of* 12 384 *faces; (bottom-right) the CT-scan of a tooth with* 56 680 *faces; and (bottom-left) the voxelization of the Venus de Milo composed of* 66 036 *faces. The numeric representation of the chain is omitted due to space constraints.*

Due to the space constraints in the printed version of this work, for the models in Fig 6.7 we have limited to the visual display of their descriptors without the corresponding numeric representation as a chain. Instead, we only illustrate the superimposed Hamiltonian cycle relying upon our color system to portray the notion of the different chain code values. In addition to that, a zoom-in window to a small portion of the model is used to illustrate with higher detail certain regions of the surface.

6.1.2 3D OBJECTS COMPRESSION

The representation of 3D objects by means of spatial occupancy arrays calls for large amounts of data to be stored as the storage requirements increase by the cubic of the linear resolution of a three-dimensional object. Moreover, even when these structures are subject of high compression rates, the efficiency of the compression algorithms substantially decreases as the discretization levels increase. Among other things, this means that dealing with multi-value datasets is not nearly as efficient as storing a binarized version of the dataset. Furthermore, there are fundamental problems inherent to the representation of large ranges of the voxelized space opposed to the representation of individual objects previously isolated from the scene. One of these problems is the complexity of the segmentation itself.

In computer vision, the process of partitioning a digital representation of the volume as a multi-valued dataset is already a complicated topic that frequently requires the use of complex algorithms (e.g., edge detection, region-growing, clustering). In the best case, this segmentation process represents at the very least an additional step of moderated complexity that is needed prior the display or manipulation of a model. That is when representing models isolated from the rest of the scene pays off.

In order to address the above mentioned problem, we explore alternative mechanisms for the representation of segmented 3D models with the primary goal of providing efficient storage as well as displaying and operating these objects directly from their descriptors. Our proposal is based on the direct representation of the enclosing surface, thus removing the rest of the 3D voxel-based space. Hence, an intrinsic contribution of our method turns to be offering lossless and compact representation of discrete surfaces by means of a 1-dimensional chain descriptor.

In Table 6.1 we do a numerical assessment of our Hamiltonian-cycle-based method using the ratio of the chain size (CS) in comparison to four of the common representations used to represent 3D objects. Firstly, a surface described as a simple set of vertices and faces (SS). This representation uses 32-bit integers to describe the vertex coordinates and each of the four vertex-index representing a face. The second comparison is made against an uncompressed version of the dataset after being binarized and optimized to describe 1-bit voxels (DS). The third representation of the solid is with an octree of the dataset [55] using one bit to represent each node of the tree (ODS). Finally, the fourth comparison is against the compressed version of the octree used in the previous step (HODS). The compression is made using a *Huffman*[1] encoding technique [56].

In the table we also provide the length of the chain used to represent every model, their dimensions in 3D space, and the number of vertices and faces of their surface. It can be observed from the ratio CS/SS that the chain is a much more compact representation than the traditional surface methods (e.g., obj, ply, 3ds, wrl, stl, fbx). In particular, our chain representation stabilizes in the limits to the 2% of the size required by these other methods.

In the case of the ratio CS/DS, the ratio decreases as the model resolution increases and shows again a better performance in our chain representation. The ratio decrement as the size of the model increases can be explained by the direct mapping in the lenght of our chain code with respect to the size of the described surface. In contrast, for the volumetric representations, the size of a dataset grows in proportion to the three-dimensional space of the model implying a higher representation of data. With the ratio CS/ODS, our chain representation still demonstrates a higher efficiency as the size of the model increases, specially when the volume tends to become irregular.

Finally, although the ratio CS/HODS suggests the chain representation could not beat the storage efficiency in the compressed representation as an octree, our descriptor still exhibited an adequate performance by keeping the storage bounds roughly in the same order of complexity, besides of offering other potential advantages like the previously mentioned segmentation which means a faster recovery; and the set of capabilities allowing the analysis and morphological transformations by grammatical techniques applied to the chain.

[1] In information theory the Huffman-codes are a widely used variable-length encoding for compression purposes [56]. Huffman's method is an entropy encoding based on having the more frequent symbols assigned a shorter code than those with lower probability, or frequency of occurrence.

MODEL	CHAIN	SURFACE		DATASET	RATIO CS/x				
	Length	Verts	Faces	W×H×D	CS* (KB)	x =SS* (%)	x =DS* (%)	x =ODS* (%)	x =HODS* (%)
elephant	3 990	3 983	3 990	$50 \times 50 \times 50$	1.52	1.40	10.01	112.52	158.78
horse	12 384	12 376	12 384	$90 \times 90 \times 90$	4.62	1.37	5.20	82.15	186.16
venus	66 036	65 992	66 036	$256 \times 256 \times 256$	18.12	1.00	0.88	29.59	161.19
tooth	56 680	56 638	56 680	$256 \times 256 \times 161$	17.25	1.11	1.34	26.40	142.13

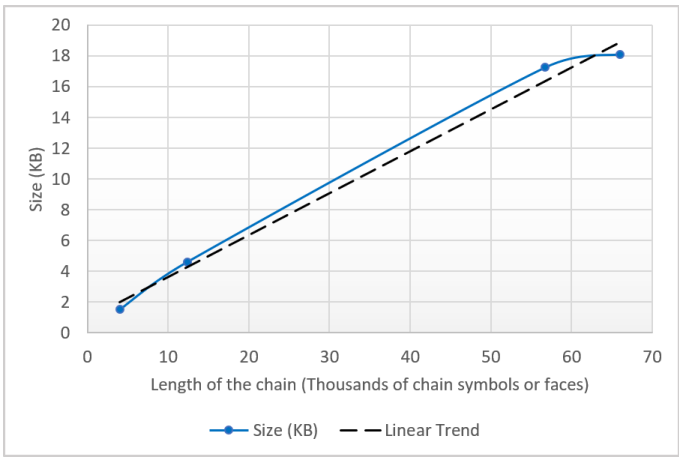
* File size of the model described as:
(CS) the proposed chain compressed by using Huffman encoding [56];
(SS) an uncompressed representation of its surface by using 32-bit numbers for each vertex coordinate and face index;
(DS) an uncompressed representation of the dataset;
(ODS) an octree representation of the dataset;
(HODS) a compressed octree representation by using Huffman encoding.

Table 6.1: *Comparison of the storage requirements when using our proposed chain code against some of the common representations as a surface and volume. For the models used in this section, the table presents its dimensions when represented as a chain, surface and dataset. In the last four columns shows the size percentage of our chain code (CS) with respect to the other representations (SS, DS, ODS, HODS).*

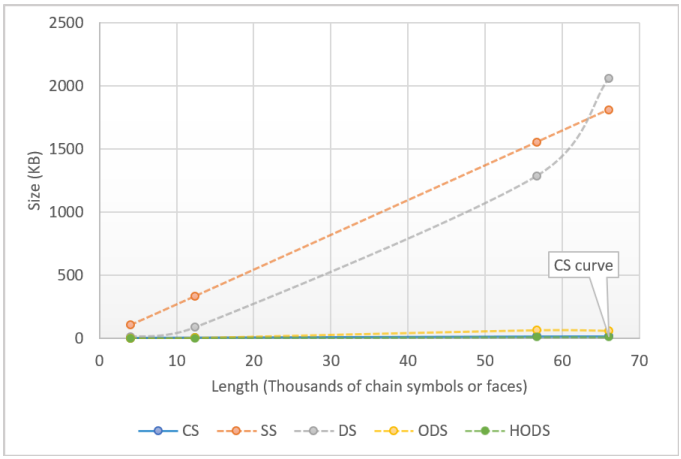
Fig 6.3a plots the behavior of the function describing the size increment of the chain as the size of the model increases. This trend is approximated by an order-1 (linear) equation where the slope $m < \frac{1}{3}$ indicates that a single storage unit (i.e. a byte) represents in average just a little bit more than three polygons of the enclosing surface.

In Fig 6.3b we present another plot in order to compare the growth of the different types of 3D descriptors with respect to our chain code method. The chart illustrates a clear compression advantage of the CS, ODS, and HODS methods over the geometrical-based SS and DS, hence supporting our method as a higher-compression alternative to the raw representations of the surface or the volume.

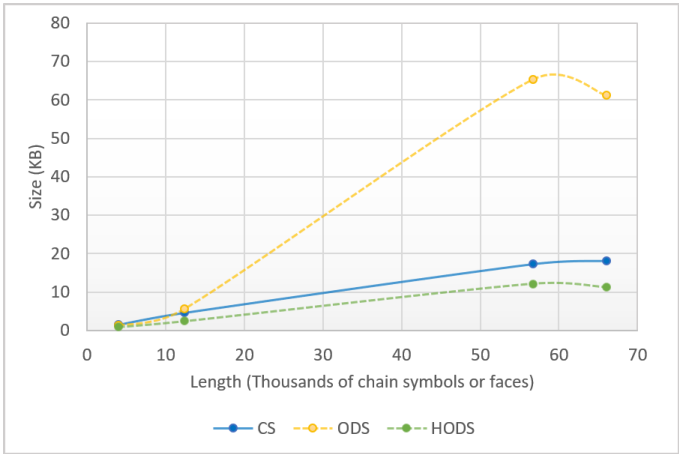
While the value range for the y-axis in Fig 6.3b is linearly plotted to be illustrative of a real comparison against the mechanisms describing raw surfaces and volumes (i.e. SS and DS), in Fig 6.3c we use a different range to center the comparison with the more sophisticated ODS and HODS methods. As explained during the analysis of Table 6.1, our compression method is very comparable to the Huffman encoding of an octree representation but with the advantages mentioned before.



(a)



(b)



(c)

Figure 6.3: *(a) Plot of CS growth as the size of the model increases. (b) Comparison of CS, SS, DS, ODS and HODS growth. (c) Comparison of CS, ODS, HODS at a better scale*

6.1.3 Comparison with other chain codes

There exists a number of works in the specialized literature already proposing the use of 3D chain codes as an alternative method for the representation of different three-dimensional structures [37, 40, 57]. To the best of our knowledge, our chain code is the first one specifically used for the representation of discrete surfaces. Nevertheless, when in comes to other chain codes used to represent digital curves, we can find that in most cases these are easily adaptable to describe a given sequence of faces in the discrete surface. Along this subsection we briefly compare our chain code against two of the well-known chain codes, the first one proposed by Freeman [37] and the second by Bribiesca [40].

While Freeman's code was originally intended for tracking 2D contours, a simple adjustment to the chain code's alphabet can easily extend it to represent 3D curves. Since the discrete curves studied in this work tracks the surface exclusively following orthogonal changes of direction, we can then use the alphabet of six elements illustrated in Fig 6.4a. In Fig 6.4c we use the label <Fre> to indicate the chain corresponding to this variant of the Freeman's code and it tracks a given sequence of faces (i.e. computed Hamiltonian cycle) in the surface of the voxelization.

Also in Fig 6.4c, we present the chain obtained from the method proposed by Bribiesca [40] for 3D curves; such a chain is labeled as <Bri>. Bribiesca's code uses an alphabet coding the five relative changes of direction shown in Fig 6.4b [38]. Finally, we use the label <Our> to indicate our proposed chain code using an alphabet of nine relative changes.

In addition to the number of elements in each alphabet, it is worth noticing that the codes proposed by Freeman and Bribiesca require twice the number of elements than our code does. This gives to our code an advantage for the compression application since this means a single face can be encoded by using one of only nine available values; this in contrast to the 36 and 25 possible values in Freeman's and Bribiesca's codes, respectively[2]. This ultimately implies a higher entropy of our method for the representation of discrete surfaces.

[2]The number of available values corresponds to the number of possible combinations resulting from having two contiguous elements of the chain.

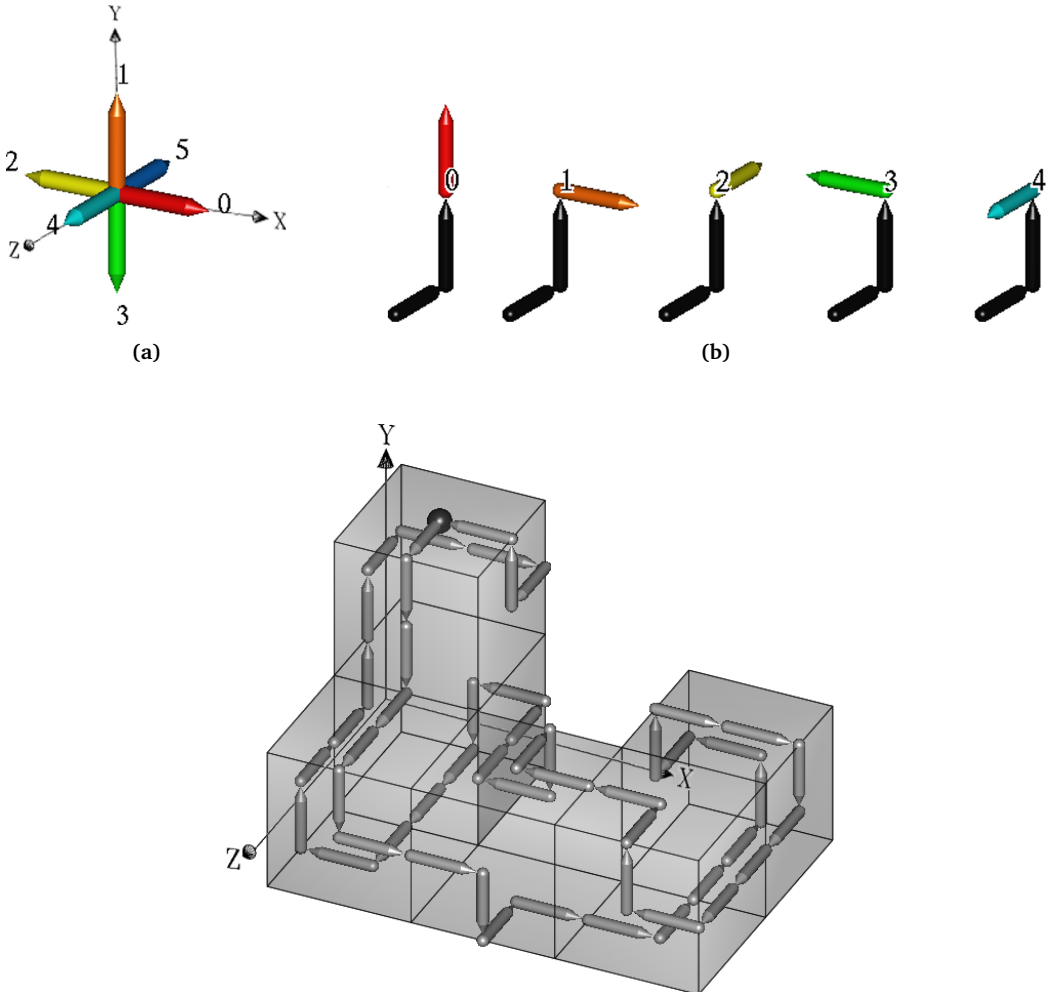


Figure 6.4: *(a) Extension of the Freeman's chain code alphabet to represent 3D curves. This codification depends on an absolute coordinate system. (b) The five relative changes of direction in Bribiesca's chain code for digital curves. (c) Codification of one cycle by using the Freeman's code <Fre>, Bribiesca's code <Bri> and our proposed chain code <Our>.*

6.1.4 Multi-resolution representation of an enclosing surface

Some of the major topics in computer graphics are related to the compact representation and efficient display of 3D objects. For this purpose, the display of the surfaces proceeding from a voxelization play an important role being among the most common meshes given their simplicity and the convenience entailed over other traditional polygonizations which commonly lead to over tessellation problems or geometric ambiguities [45]. In Chapter 3 we briefly addressed some details in this regards and discussed about the efficiency of the algorithms used for surface

extraction striving in some of these advantages.

In the other hand, one of the common concerns of using surfaces proceeding from a voxelization in their coarse appearance but for this issue there also exist mitigation techniques to improve the visualization of these cubic surfaces. Most of these techniques in fact, act directly over the illumination model to "fake" its boxy appearance and make it look smoother. For example, a simple manipulation of the surface normals can help to retrieve smoother renderings [58]. It is in part because of these encouraging results and the ubiquitous presence of these kind of surfaces that we consider our method entails a contributions to a relevant area.

In Fig 6.5 we present at multiple resolutions the model of a CT-scanned tooth previously used in Fig 6.7. The eight resolutions used in Figs 6.5a-h correspond to voxelization of the model as a dataset of 1, 2, 4, 8, 16, 32, 64, and 128 voxels per side, respectively. For each of these representations a single chain was computed to describe the corresponding surface. Previously, in a different subsection, we already noted in terms of the compression efficiency how advantageous our contour-oriented description is versus a raw volume-based representation.

Given the importance of the problems it can solve, different mechanisms for the measure of discrete compactness have been studied in the past. In addition to the compression of 3D objects, in this subsection we expand the areas of application of our method to also produce a measure of discrete compactness that is directly computed from the chain descriptor.

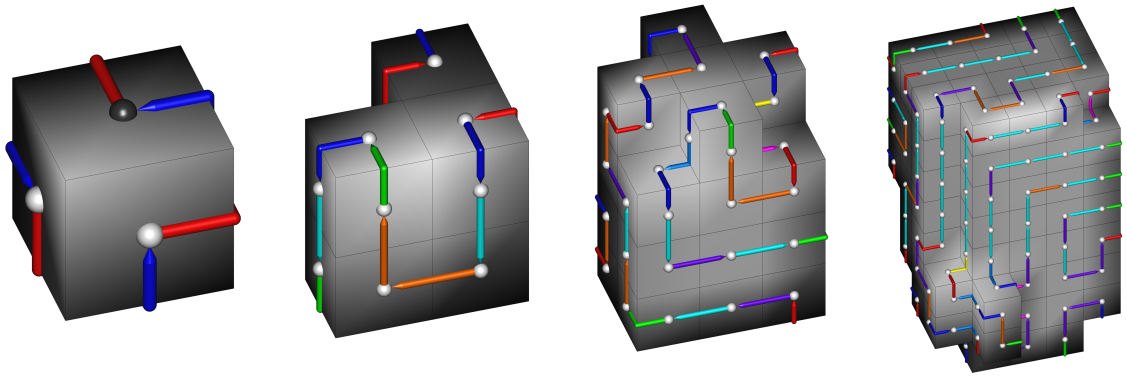
Based on the measure of compactness presented in [59] and assuming that the length of each side of the voxels is equal to 1, we replace from the formula in Bribiesca's work the area of the enclosing surface for the length of our chain descriptor to obtain the following measure of compactness:

$$C_d = \frac{V - n/6}{V - \sqrt[3]{V^2}},$$

where C_d is the discrete compactness, V is the number of voxels of the solid, and n is the length of the chain descriptor.

Fig 6.5 shows under each subfigure the number of voxels, the length of its chain descriptor, and its measure of compactness. It is important to note how some of the gaps within the tooth recesses start to get filled up with smaller voxels which gives as a result a higher measure of compactness

as the model resolution increases. As for models previously used in this section, their measure of discrete compactness based on their sizes and chain length are 0.94407 for the elephant, 0.96428 for the horse, 0.98563 for the Venus de Milo, and 0.99102 for the tooth in its original resolution. These measures roughly correspond to what it could be anticipated from a visual inspection to be the order from the less to most compact model.

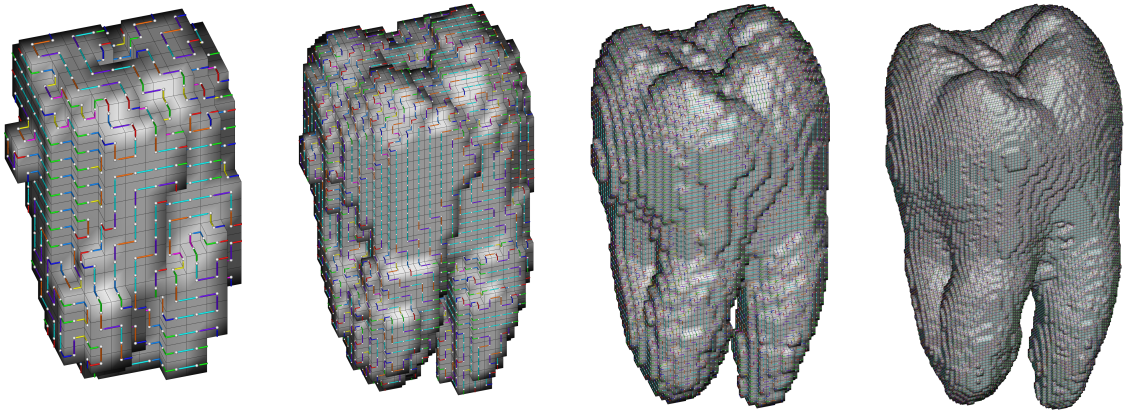


$V = 1$
$n = 6$
$C_d = undefined$
(a)

$V = 5$
$n = 20$
$C_d = 0.80283$
(b)

$V = 24$
$n = 58$
$C_d = 0.91413$
(c)

$V = 124$
$n = 176$
$C_d = 0.95494$
(d)



$V = 729$
$n = 652$
$C_d = 0.95730$
(e)

$V = 4\,586$
$n = 2\,458$
$C_d = 0.96899$
(f)

$V = 31\,873$
$n = 9\,348$
$C_d = 0.98209$
(g)

$V = 236\,054$
$n = 36\,518$
$C_d = 0.99023$
(h)

Figure 6.5: *Multi-resolution representation of the CT-scanned model of a tooth used in this chapter. Voxelizations are adapted to the following resolutions in order to fit in a dataset of maxside equal to (a)1; (b)2; (c)4; (d)8; (e)16; (f)32; (g)64; and (h)128. Below each representation is the number of voxels of the solid (V), the length of the chain descriptor (n) and a measure of discrete compactness (C_d).*

6.2 Tree-based description of complex voxelized surfaces

As a complement of the representation made for simple voxelizations, this section studies the extended version of the descriptor based on a spanning-tree structure covering the surface. That is, our tree structure can span over any voxelized surface regardless of its topological genus, and whether it is a closed surface or just a patch. Throughout this section we use this method to represent a wide variety of these voxelized surfaces.

6.2.1 Representation of large surfaces

Below some examples of large objects are described by means of the surface tree method introduced in Chapter 5. Similarly to the set of models used in the previous section, the discrete surfaces analyzed in this section are also either directly created or obtained from the implicit voxelization in a dataset. In the case of multi-value datasets, the final voxelization was extracted by thresholding to an arbitrary selected isovalue chosen by visual inspection.

In Fig 6.6 we start with the encoding of the scanned model of a bunny made out of 2 206 faces which is described with a chain of 6 615 elements. As part of this figure we illustrate the encoding of the proposed tree structure superimposed to the corresponding surface. Also in this figure, we use again the system of five color labels to assist with the identification of different regions in the surface described with the chain.

Fig 6.7 presents more examples of discrete surfaces proceeding from different sources, all of them described with a surface tree. Starting from the top-left corner, we show the model of a rib cage obtained from the 3D reconstruction of a CT-scanned human chest [60]. The model in this figure is an implicit surface whose isovalue was appropriately adjusted to display the rib cage and it is composed of 593 412 faces. In the bottom-right corner we show the scan of a Buddha statuette whose voxelization presents an enclosing surface made of 32 146 faces. Finally, the bottom-left corner shows a digital elevation model (DEM)[3] of the Iztaccihuatl volcano whose enclosing surface has 121 548 faces. Due to space constraints, this figure omits the numeric representation of the chain and only displays the tree descriptor superimposed over the discrete surface.

[3]Digital Elevation Models are digital representations of a terrain's surface and are commonly the basis of digitally produced relief maps [1]

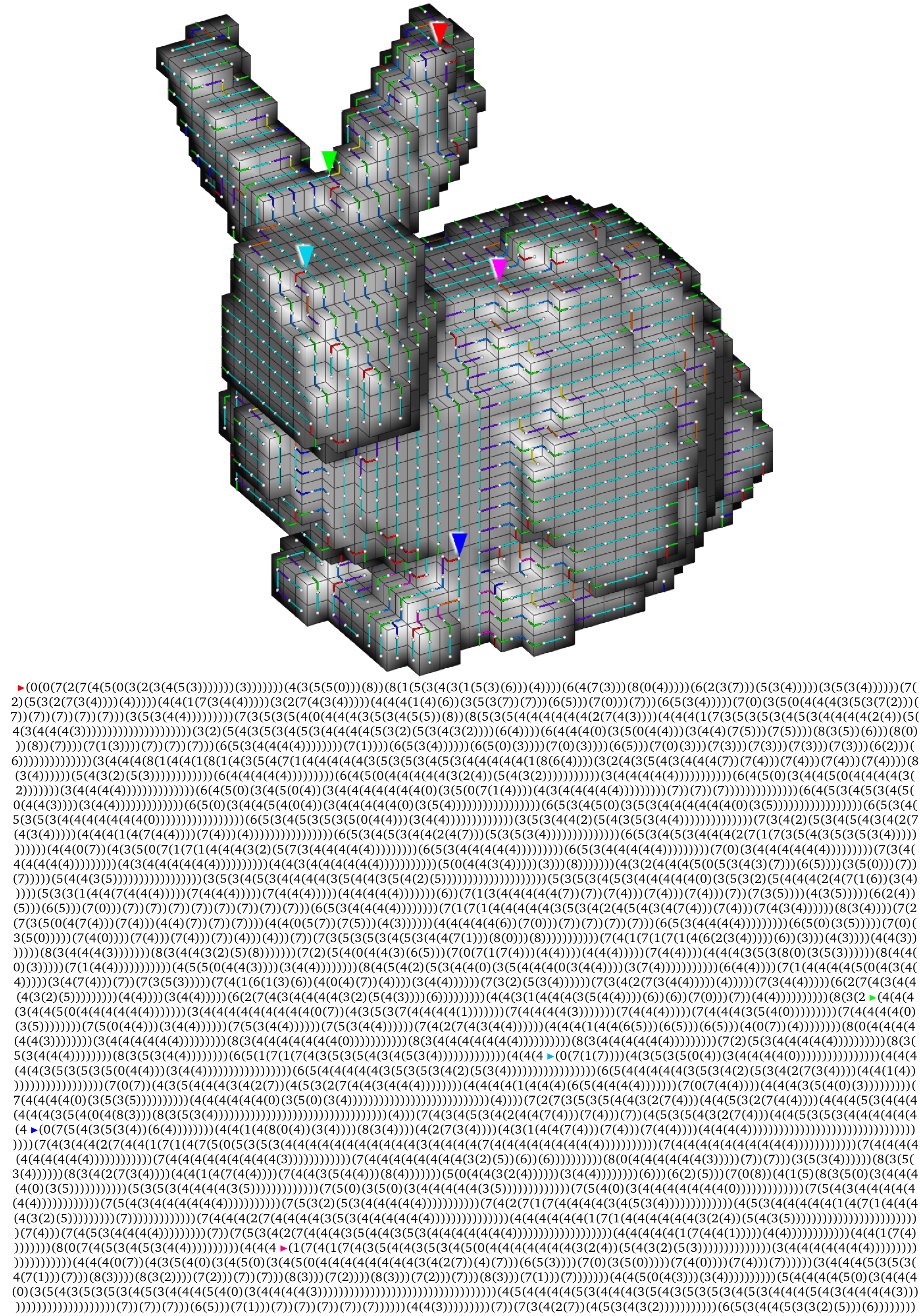


Figure 6.6: *Surface tree description of the Stanford Bunny whose enclosing-surface consists of* 2 206 *faces. Below the model we include the chain code of* 6 615 *elements which uses five different labels to assist the reader following the descriptor in different regions of the surface.*

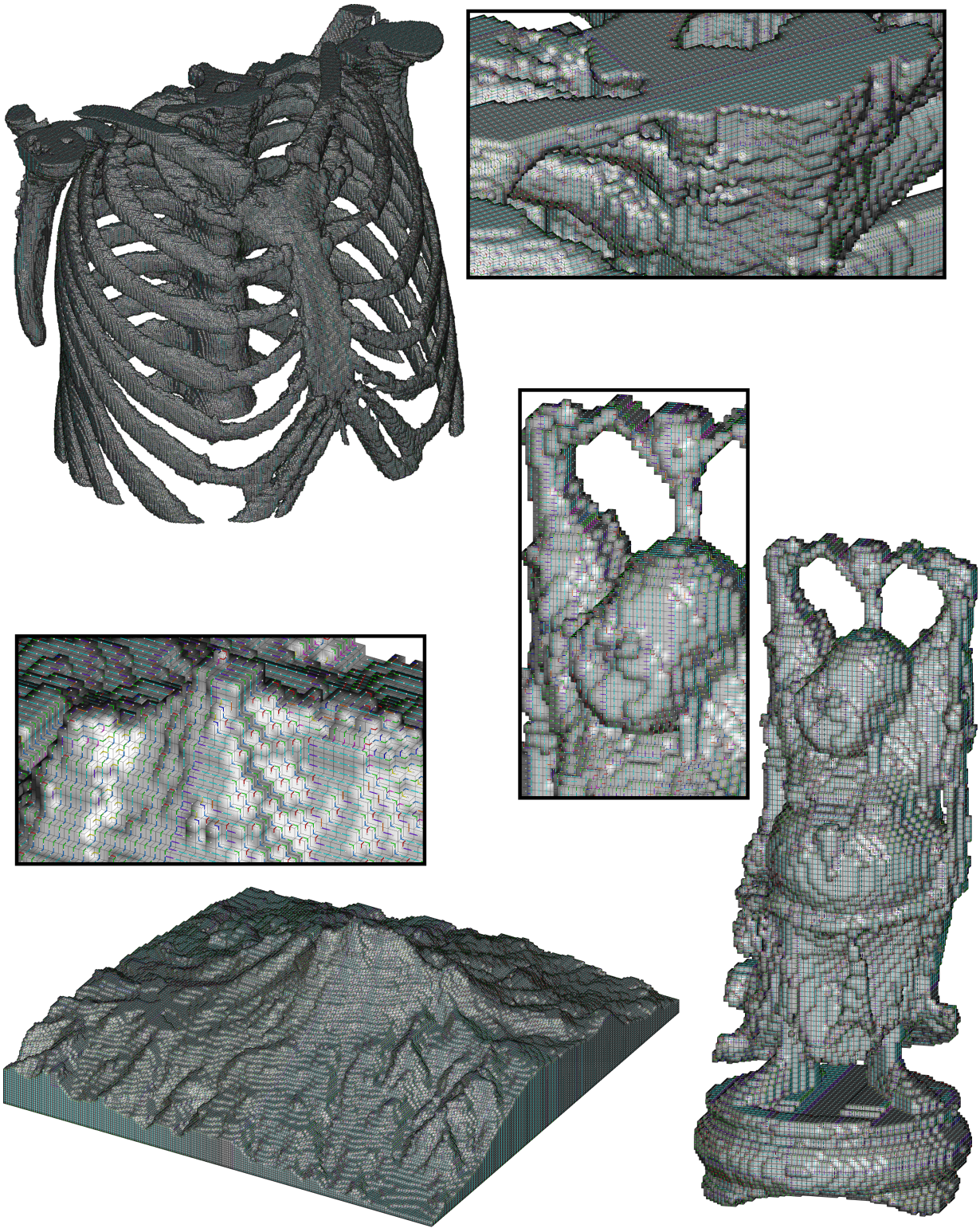


Figure 6.7: *Three different models described with a surface tree: (top) rib cage drawn with* 593 412 *faces and extracted from a 3D reconstructed dataset of a CT-scanned human chest; (bottom-right) surface with* 32 146 *faces obtained from the scan of a "Happy Buddha" statuette; and (bottom-left) a surface with* 121 548 *faces generated from a digital elevation model (DEM) of the Iztaccihuatl volcano. Due to the limitation of space, the numeric representation of the chain is omitted.*

By having all models in this section coming from different sources, we pretend to perform a more comprehensive validation of our methodology. Hence, we include large surfaces that in some cases are artificially generated but in some others are representations of real objects used in real applications. Among these models we bear with different topological genus (i.e., in homeomorphism to the n-torus) like in the case of the Buddha statuette containing multiple holes; or even open surfaces like in the case of the rib cage model.

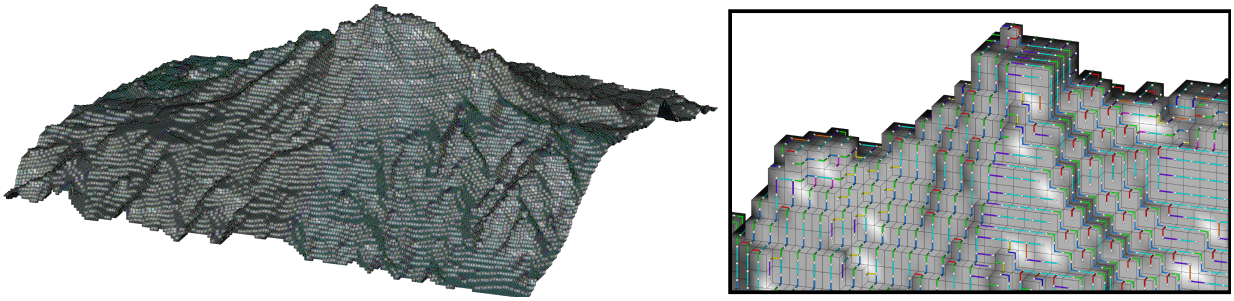
Regarding the representation of open surfaces, Fig 6.8 shows the example of an open surface corresponding to the landform of the Iztaccihuatl volcano. This figure only uses the land relief patch from the enclosing surface of the Iztaccihuatl model we previously presented in Fig 6.7. We take reuse the model but this time only representing as an open surface its relief. Some of the morphological operations directly applied over the tree descriptor are illustrated using this figure. More specifically, Figs 6.8b and 6.8c illustrate respectively the mirror and complement transformations applied to the surface patch presented in Fig 6.8a.

6.2.2 Compression efficiency of a surface tree

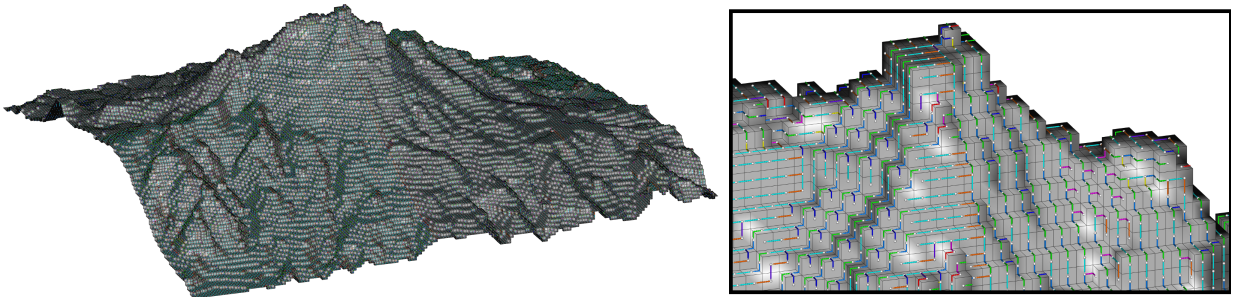
Previously in this chapter, we analyzed the storage advantages for our 1-dimensional Hamilton-based descriptor over the traditional occupancy arrays and other representations. To some extents, this section is just a corollary for the descriptions made with a surface tree as we will see these advantages do not change much.

While the encoding of a tree structure greatly depends on the additional step used to serialize it into a single chain, most of compression efficiency analysis remains true with one simple adjustment: now, to compute the length of the chain, we also need to take into consideration the overhead added by the parenthesis notation required to serialize this structure. More specifically, when we refer to equation 5.1, the storage requirements increase by a factor of 3 with respect to the 1-dimensional descriptor.

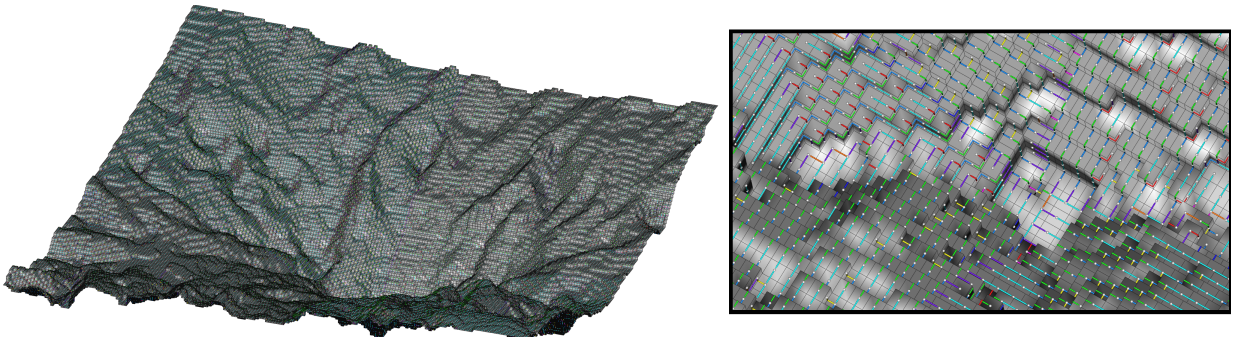
Despite of this space increment by a constant, the Big-$\mathcal{O}$ analysis does not change its order of complexity remaining based as a linear function on the number of faces the discrete surface being represented.



(a)



(b)



(c)

Figure 6.8: *Discrete surface representing the relief of the Iztaccihuatl volcano model in Fig 6.7. (b) Result of applying the mirroring (a.k.a. specular image) transformation to the tree description in (a). (c) Result of applying the complement transformation to the tree description in (a). This figure illustrates the usage of the mirror transformation to find the symmetries, and the complement transformation to convert those valley regions in the landform into hills, and viceversa.*

6.2.3 AMBIGUITIES IN SIMILAR METHODS USED FOR THE REPRESENTATION OF ENCLOSING SURFACES

There are other methods in the specialized literature also used to describe voxelized surfaces but sometimes suffering of some ambiguities. For example, Sagols [49] explores the use of a Hamiltonian sequence of numbered faces as the way to describe voxelized solids, also referred as *voxsolids*. Similar to our method, Sagols' work uses the face adjacency graph as its fundamental structure to extract these sequences and has as its main contribution the conjecture that such a graph is always Hamiltonian for any voxsolid [61].

Nevertheless, due to its lack of references to reconstruct the geometrical representation of the solid, this characterization of the surface still falls short from being a reliable method for the representation of discrete solids. In particular, this limitation results in a number of ambiguities when it comes to the representation of voxsolids. For example, the Fig 6.9 shows a simple case where two different voxelizations may lead to have the same adjacency graph, therefore to have the exact same descriptor. Moreover, the lack of a suitable polynomial algorithm to compute the Hamiltonian sequence restricts its use to only those cases where the voxelization does not contains holes as it has been previously assessed.

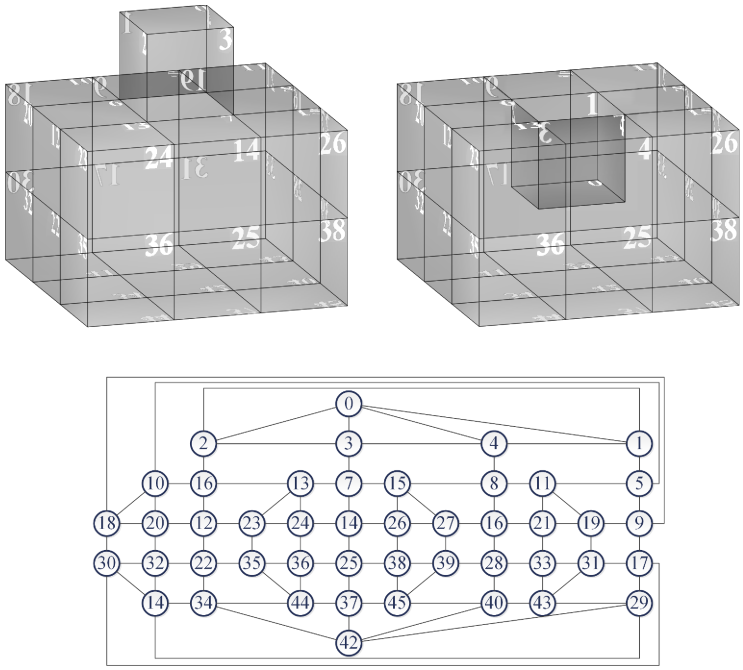


Figure 6.9: *Example of the ambiguity of representing boundary surfaces exclusively by using their face adjacency graph. The figure shows two different surfaces with the same adjacency graph resulting in the same sequence of faces to describe the voxelization.*

Another method also using a tree descriptor spanning along the enclosing surface is the *enclosing trees* proposed by Bribiesca et. al [1]. Enclosing trees expand through the edges of the surface, hence, they use a tree of maximum degree equal to six given the number of edges that can be incident to a single vertex.

Although the enclosing trees method holds some similarities with our tree descriptor, in our method we use trees of lower degree and their encoding depends on a different alphabet that is exclusive for the representation of surfaces rather than 3D curves.

In the case of enclosing trees, similar ambiguities in the representation of voxelizations may lead to have a single descriptor representing multiple voxsolids.

Fig 6.10 illustrates a simple case used as a counter examples of one of those cases where a single descriptor represents two different voxsolids. While some details in the implementation of the surface trees' algorithm could lead to have a slightly different output where the final encoding avoids this particular problem, this still helps to understand that chances of eventual occurrences of other ambiguous instances are possible.

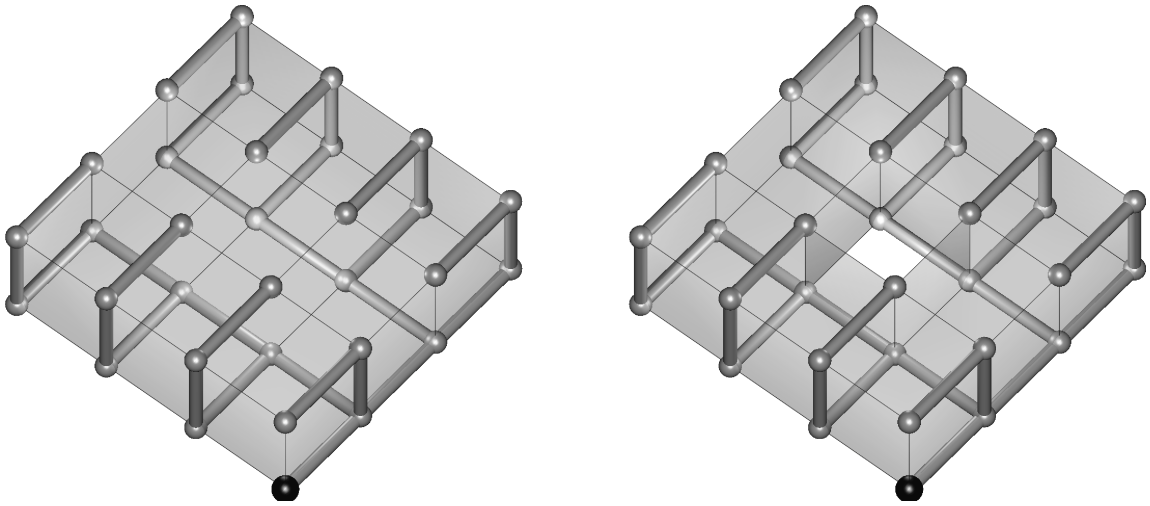


Figure 6.10: *Example of an ambiguity found in the enclosing trees method presented in [1]. This figure illustrates two different enclosing surfaces represented by the same enclosing tree.*

6.2.4 Comparison with Enclosing and Edging Trees

In comparison with *enclosing trees* [1], our method not only avoids potential pitfalls related to the representation of multiple solids using the same descriptor but also offers an edge when it comes to the representation of non-closed surfaces. Specifically, for the reconstruction of a surface from our chain descriptor, each element in the chain possesses an explicit meaning on how the corresponding face should be placed in the surface without requiring any additional context from other non-contiguous chain elements as it occurs with the enclosing surface method.

From a compression perspective, by having the length of a chain descriptor determined as a function of the number of vertices or faces, respectively, both methods flaunt very comparable compression efficiency as per the Euler's characterization [4, 5] ensuring the number of vertices grows more or less in the same order of magnitude than the number of faces of the surface.

Another related method that can be used as reference is *edging trees* [42] which is an extension to the enclosing trees where an additional brute force post-process allows the optimization of large flat portions of the surface. While this extension reuses the same chain code and same alphabets the enclosing trees use, the length of the output chain gets reduced in proportion to the amount of flat regions in the surface. In those cases, the compression efficiency of edging trees surpasses that from our method at the expense of having a much more complex extraction of the edge encoding.

Finally, a natural comparison for our surface tree method is with the 1-dimensional descriptor earlier proposed in this work. That descriptor results from the encoding of a Hamiltonian cycle extracted from the face adjacency graph of a simple voxelization and, although both methods use the same alphabet of nine symbols, in terms for compression efficiency the Hamiltonian descriptor has an edge over surface trees. Nevertheless, we still need to consider the surface trees method as an extension of the Hamiltonian description but being a more generic solution for the representation of any discrete surface, and allowing the surface encoding regardless of its topological isomorphism.

6.2.5 Measure of tortuosity (surface roughness)

Tortuosity is one of the interesting properties of curves and surfaces with several applications that can be derived from it [27, 29, 30, 34]. For instance, in materials engineering, the surface tortuosity is an important parameter related to the tribological[4] behavior of surfaces so it is used to study asperity and deformations caused by the increments in their friction, as well as other phenomena. In [28], a metric of the surface roughness is used to assist in the analysis of cracks in concrete. Regarding usage in medical applications, a research in [62] states the surface roughness as a key factor in determining the low-thrombogenic[5] behavior of biomaterials. This section deals with a new proposal to measure surface tortuosity directly from the tree description.

Our proposed measure of surface tortuosity τ, also considered as the surface roughness, is defined by the ratio:

$$\tau = \frac{transitions_{non-flat}}{transitions_{total}}$$

where $transitions_{non-flat}$ refers to the number of occurrences described in Fig 4.1 as convex, or concave transitions (i.e., $0, 2, 3, 5, 6, 8$). The ratio τ is ultimately obtained by dividing over the total number transitions encoded in our tree descriptor.

Considering the multiple ways existing to span a tree fully enclosing the represented surface, we also propose the use of the well-known family of MST algorithms [25, 26] to compute a canonical form of the tree that minimizes the amount of non-flat transitions. Our measure of surface tortuosity is then defined as a scalar value in the range [0, 1] where 0 is assigned to a completely flat surface, and 1 represents the maximum possible roughness for a discrete surface with no flat zones.

In Fig 6.11 we present some examples of our proposed measure for three different surfaces. Fig 6.11a illustrates the simplest of these surfaces, corresponding to a model composed of 9 voxels and 29 transitions (i.e., 30 faces). In this model, the minimum number of non-flat transitions required to enclose the surface is equal to 5, therefore the measure of relative tortuosity τ is $5/29 \simeq 0.17241$. Fig 6.11b isolates the tree descriptor and highlights its non-flat transitions for a better visualization.

[4] Related to the study of interacting surfaces in relative motion considering principles of friction, lubrication and wear.
[5] Thrombogenicity refers to the tendency of a material in contact with the blood to produce a thrombus or clot.

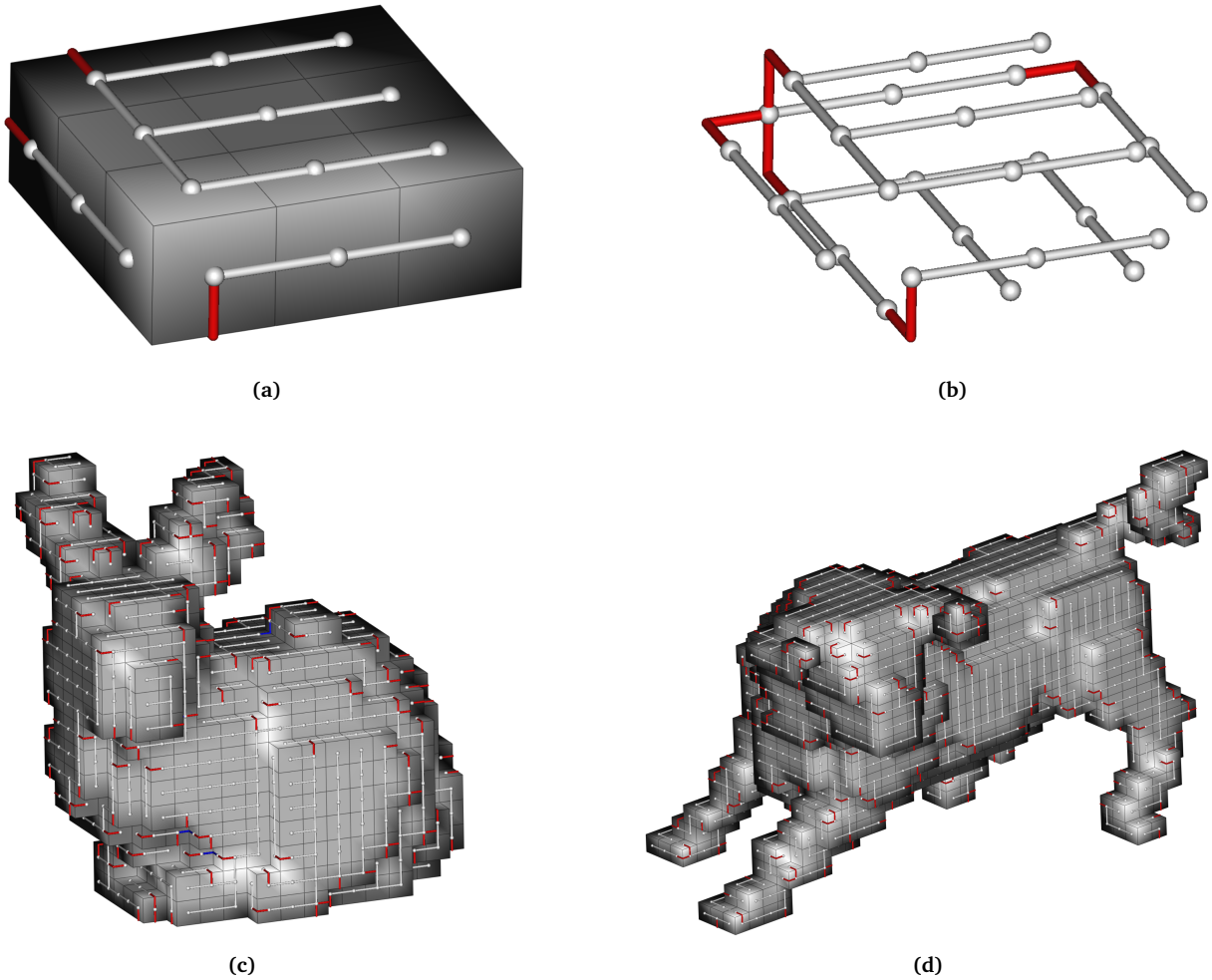


Figure 6.11: *Measure of surface tortuosity (surface roughness) computed for a MST-based tree descriptor minimizing the number of non-flat transitions: (a) 9-voxels model with 5 non-flat transitions; (b) isolated tree from the previous model; (c) bunny model whose tree descriptor uses* 1 035 *transitions,* 251 *non-flat; (d) bull model with* 2 017 *transitions,* 496 *non-flat. Digital version of this document represents non-flat transitions in red (covex), or blue (concave) colors to distinguish from the rest of them represented in white (flat).*

Figs 6.11c-d display more examples of models composed of 1 222 and 2 274 voxels, respectively. In the case, of the bunny (bunny16) the spanning tree uses 1 035 transitions where 251 of them correspond to non-flat transitions producing a ratio $\tau \simeq 0.24251$. For the model of a voxelized bull, the ratio $\tau = 496/2017 \simeq 0.24591$.

In Fig 6.12 we illustrate the measure of tortuosity applied on more complex models. In the top-left corner we show the model of a building with a surface tree of 78 717 transitions, with only 2 845 of them being non-flat resulting in $\tau \simeq 0.03614$. Additionally, the models of a Hand and Atenea their ratio τ is approximately 0.28857 and 0.25429, respectively.

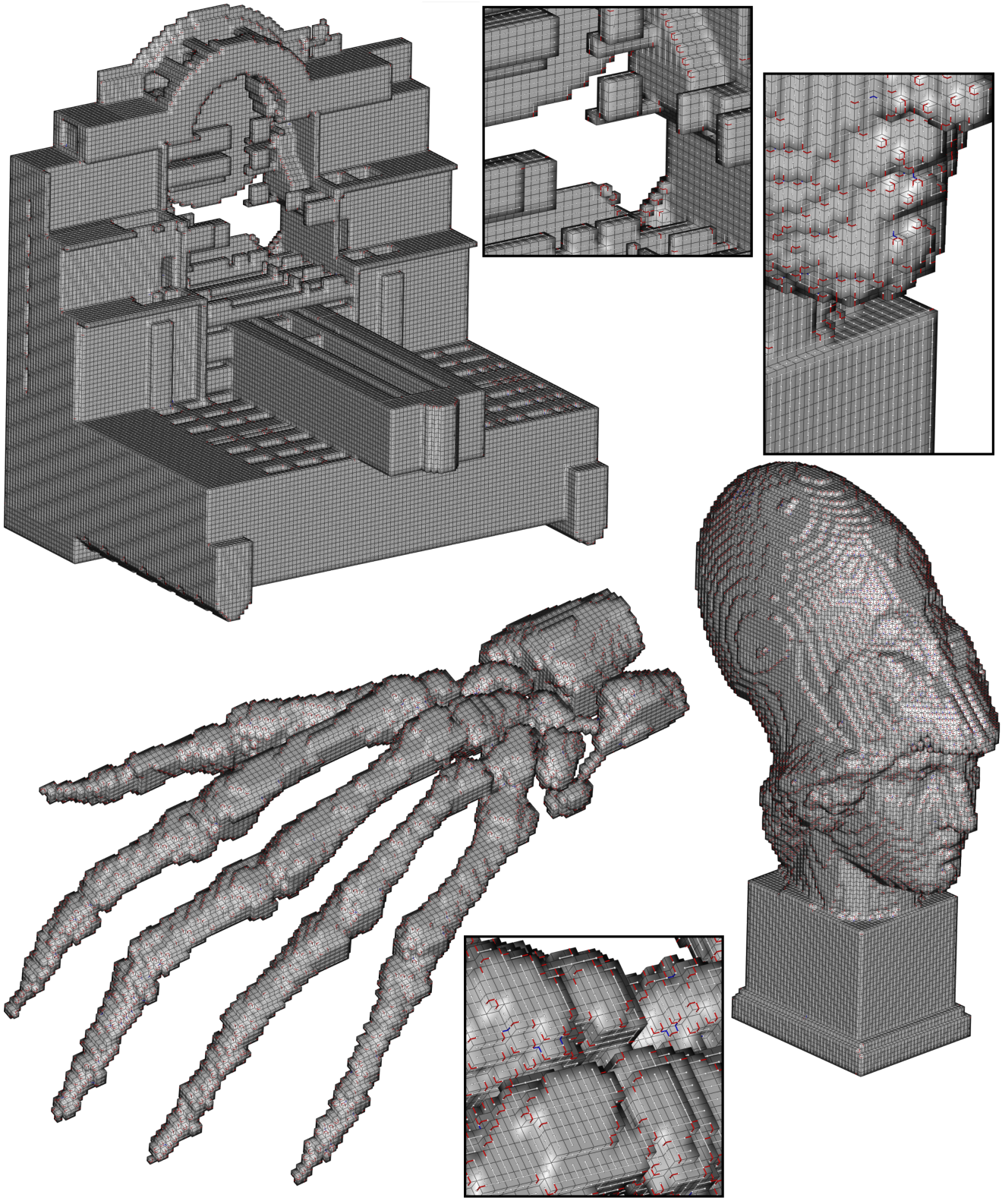


Figure 6.12: *Examples of surface tortuosity applied on larger models. Digital version of this document represents non-flat transitions in red (covex), or blue (concave) colors to distinguish from the rest of them represented in white (flat).*

Among the different object in 6.12, it is worth noticing the large flat portions in the Building model leading to a low measure of tortuosity. In contrast, the Hand model present significantly less flattening resulting in a large number of convex or concave transitions thus a large roughness.

Table 6.2 lists all the different models used along this chapter and sorts them by tortuosity in non-decreasing order. This table shows Building, Iztaccihuatl, and Simple model as those with less roughness. The result could be considered as expected given the large flatten areas in the model proportional to the entire surface. In the case of the Tooth model, much higher roughness is resulted precisely from the lack for these flattened zones. Another aspect worth to mention occurs for the two different voxelizations made for the Bunny model at different resolutions where the reported roughness keeps in a consistent range suggesting certain tolerance of our measure to the model resolution.

Finally, we consider important to emphasize our decision to minimize the non-flat transitions was made with the intention of offering to the reader ideas of simple applications. This, of course, should not limit by any means the possibilities of our method to optimize any other property, and therefore opening a world of possibilities for exploring different applications that use discrete surfaces.

MODEL	VOXELS	TORTUOSITY MEASURE		
		TRANSITIONS		$\tau = \frac{non-flat}{total}$
		non-flat	total	
building	300 258	2 845	78 717	0.03614
iztla	960 903	19 802	121 543	0.16292
simple	9	5	29	0.17241
bunny24	3 673	517	2 205	0.23447
bunny16	1 222	251	1 035	0.24251
bull	2 274	496	2 017	0.24591
chest	1 264 095	150 419	593 411	0.25348
venus	396 157	16 772	66 035	0.25399
atenea	185 494	7 900	31 067	0.25429
budha	96 089	8 531	32 145	0.26539
elphant	6 344	998	3 989	0.25019
horse	89 673	6 865	25 177	0.27267
hand	20 768	5 636	19 531	0.28857
tooth	429 887	17 820	56 679	0.31440

Table 6.2: *Comparison of the tortuosity measure applied on different models used along this chapter and sorted in an incremental order of surface roughness.*

7

Conclusions and future work

THIS work proposed a new mechanism for the representation of discrete surfaces by means of a new chain code. The represented surfaces are constructed from a set of edge-connected square faces, orthogonal to each other, and commonly proceeding from the enclosing surface of a voxelization. One of the main traits of our method is the serialization of three-dimensional surfaces into a single one-dimensional descriptor. Two different approaches were presented to accomplish this representation.

Our first approach addresses the representation of simple voxelizations, corresponding to those where the voxelization represents objects with no holes. For this approach, the serialization of the surface is accomplished by encoding one of the Hamiltonian cycles embedded in the adjacency graph. This representation deeply relies on existing proofs of Hamitonicity for 4-connected planar graphs which were cited during the background analysis.

The second approach extends our Hamiltonian method by encoding a spanning tree in the surface. We refer to this second method as *surface trees* and it allows the representation of practically any discrete surface regardless of its genus, and whether being closed or not. In fact, describing a wide range of surfaces is the main advantage held over the Hamiltonian descriptor. The down side is a lower entropy resulting from the requirements to store branch information and therefore increasing the size of the descriptor by a factor of 3.

A new chain code was also proposed in this work for the encoding of these descriptors. The new chain code encodes orthogonal changes of direction along a sequence of edge-connected faces using a nine-symbol alphabet producing chains whose length gets determined by the number of faces in the surface. The new code is studied along some of its main properties that include the invariance under rotation and translation. In this document we also offer simple ideas to normalize it with respect the starting point, as well as a set of morphological transformations that can be directly applied to the chain to obtain the mirror, the complement, or the reversed surface. To the best of our knowledge, this is the first chain code explicitly used for the representation of surfaces.

Our methodology is tested against a wide variety of models proceeding from different sources, sizes, and resolutions.

Among the applications that can be derived from our work, in this paper we analyzed the lossless compression of 3D objects showcasing its higher compression rate and other advantages over the traditional formats used for these kind of 3D models. We also presented an example where this method is used to represent a multi-dimensional voxel-based discretization of a CT-scanned image.

Specifically in the case of the Hamiltonian descriptor, a measure of discrete compactness was directly obtained from the chain descriptor by adapting a previous definition presented in Bribiesca's work. Finally, for the surface tree representation, an MST variation was also used to obtain the spanning tree with minimum number of non-flat transitions as the proposal of a measure of tortuosity. An equivalent solutions using a different MST can be pursued for the optimization of other properties like the object convexity.

Future work

There is a number of applications derived from our method initially envisioned for this work. However, we scoped in this investigation to state the foundations of our method as well as only basic applications. From a high-level perspective, further applications can fall into one of the following two categories. First, the object analysis and recognition which is based on the study of certain occurrences in the chain. This is in fact, a way to characterize the represented surface by studying patterns and offering comparison metrics directly from the chain. The second category corresponds to the direct manipulation of the surface by applying grammatical techniques.

Through this document, we briefly addressed some of the possibilities offered by the MST algorithms to optimize different properties in the spanning tree. Optimization of any given property yields to a canonical representation with potential to be a defacto metric to measure such property. For instance, a relatively simple line of investigation could follow up on possible applications that are based on the convexity or concavity of a three-dimensional object which is directly related to optimize the number of convex or concave transitions in the surface.

Among the open questions raised by this work suggesting further exploration we put to the reader's consideration a few of them:

(1) Does the topological relationship of the model and its chain allow to determine the Euler number directly from the chain?

(2) What other geometric properties can be obtained from the chain (volume, compactness,etc.)?

(3) Can we infer when the represented surface is an enclosing surface?

(4) Is it possible to find out when a chain describes a non-orientable surface in the form of a Klein Bottle, a Möbious Strip, or other topological realizations?

(5) Is there a chain equivalence between descriptors allowing to know when two different chains represent the same object?

A number of other fascinating problems could be studied in relationship to the duality of an object and its chain description.

A

Appendix: Step-by-step example of the Hamiltonian cycle algorithm

We develop in detail an example of the Algorithm 3.1 used to find Hamiltonian cycles for 4-connected planar graphs [23]. The full algorithm incorporates different cases for all the possible configurations as well as a number of considerations for each of them but in this example we illustrate the process using a relatively simple graph that only requires a subset with some of the most basic types of graph decomposition. More specifically, for the Type I reduction described in Section 3.2.1, we are not covering all the edge cases leading to interchange nodes s and t. Likewise for the Type II reduction described in Section 3.2.2, we only consider cases where the induced subgraph $G' = G - P_{sa}$ contains a single block generating the sub-case for G_b with no further separation pairs that could additionally produce subgraph types G_g^2, G_u^3, or G_g^4.

As we keep decomposing the graph into smaller subgraphs, for each subcase a recursive call is made in order to compute a small portion of the entire Hamiltonian path that once found, it gets merged into the preceding step. Unless representing the trivial case (i.e. triangle graph), each subcase generates a new step that is queued and subsequently illustrated when it corresponds. This logic causes a divergent flow that easily becomes hard to track down as the graph keeps decomposing as well as when local results start converging.

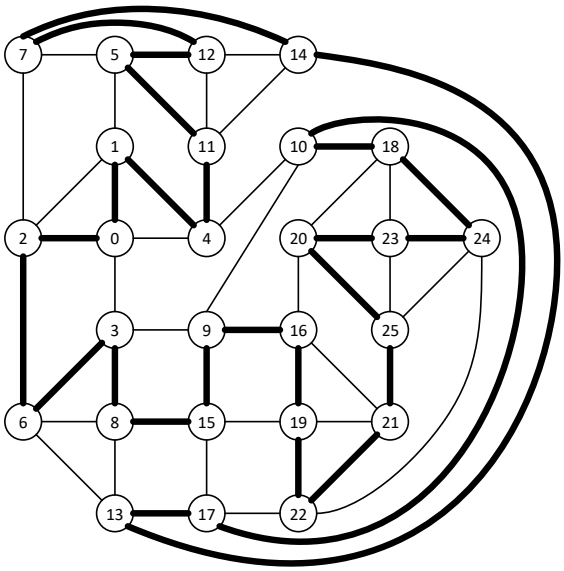
Aiming for a better interpretation of each step in the example below, we first sketch out a basic signaling convention to provide the reader with the necessary context. In the top-left corner of each figure illustrating one step, we signal its information with different boxes with the format:

Box 1 (black) indicates what the current step is. *Box 2* (green) traces back to the preceding step so once the resulted path is found the user can follow how this is merged into its caller. *Box 3* (red) is a reference to the following step(s) after a decomposition. Note that the number of sub-calls is determined by the number of subgraphs so there could potentially be multiple boxes, or even none when no further decomposition is possible (trivial case).

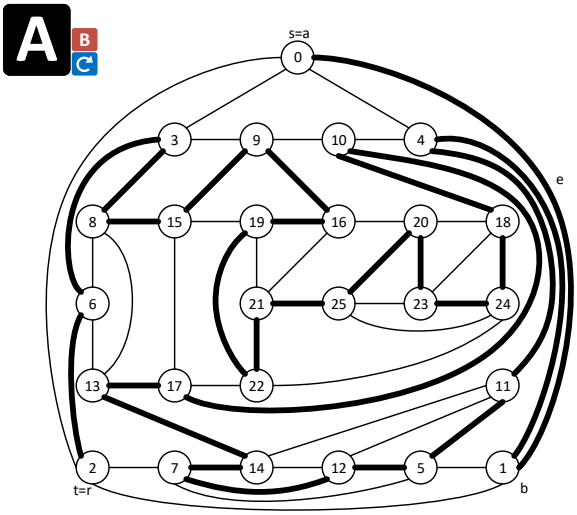
The reduction types diagrammatically explained in Chapter 3 assume without loss of generality that s, e, and t appear respectively in clockwise order in the outer cycle Z. While this is only a conceptual representation, in practice these elements can appear counterclockwise without signifying anything but a symmetrical consideration in labeling the graph elements. Same goes for a Type I reduction where left and right are merely a notation to refer to the subgraph that does (G_r) and does not (G_l) contain e. *Box 4*(blue) provides the context in this regards.

Each step labels s, t, and e as these are the main parameters used by the algorithm. These are also labeled with apostrophe in each of the decomposed subgraphs to denote the parameters passed to the recursive calls. Labels for a and b; x and y; and r are used only when necessary.

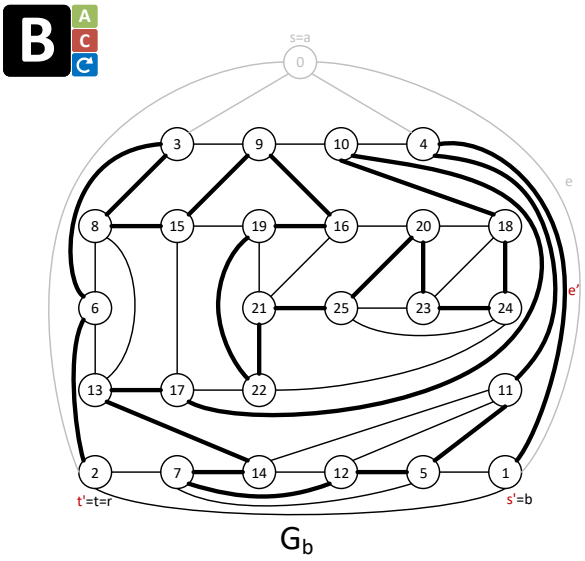
Finally, the portion of a graph needing to get removed is grayed out and the partial results of H-paths are outlined with bold lines so this can later being merged into its preceding step.



In this example we use the adjacency graph presented in Fig 3.1 arbitrarily selecting two adjacent nodes s, t, and one arc $e \neq (s,t)$ incident to s. We will use 0, 2, and arc $(0,1)$, respectively. Note that s and t must be adjacent so we can close the cycle once the path $s-t$ is found. The reason for e to be different to (s,t) is precisely to reserve this arc (*aka* f) to close the cycle. See Fig 3.2.



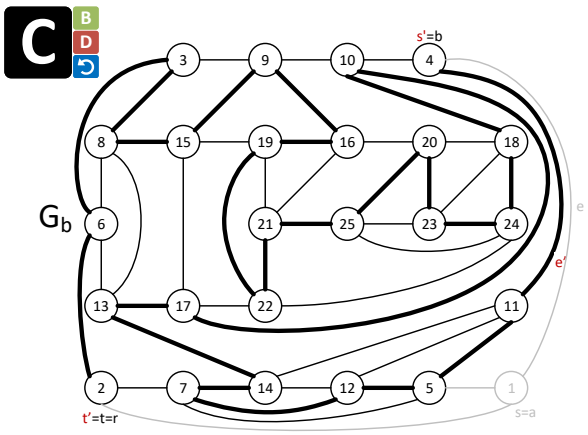
To reinforce with a visual reference, we redraw the graph so its outer facial cycle Z contains the nodes s, t, and the arc e thus the graph resembles that diagrammatically presented in Fig 3.2. Note how the adjacency relationship among all nodes of the graph is preserved as well as the clockwise order of appearance for all its neighbors.



G_b

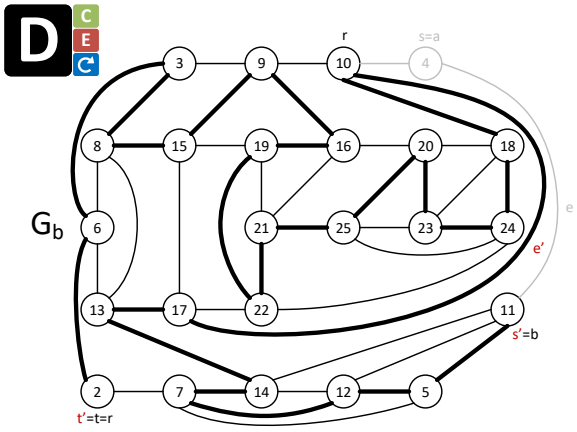
Split: by no existing a separation pair (see Definition 3.1), the Algorithm 3.1 leads to a Type II reduction where the path P_{sa} only consisting of node 0 is removed. A recursive call is made over the induced subgraph G_b after s, t, and e are redefined as directed in Subsection 3.2.2.

Merge: the combined Hamiltonian path is a merge of P_{sb} and Hamitonian path for G_b.



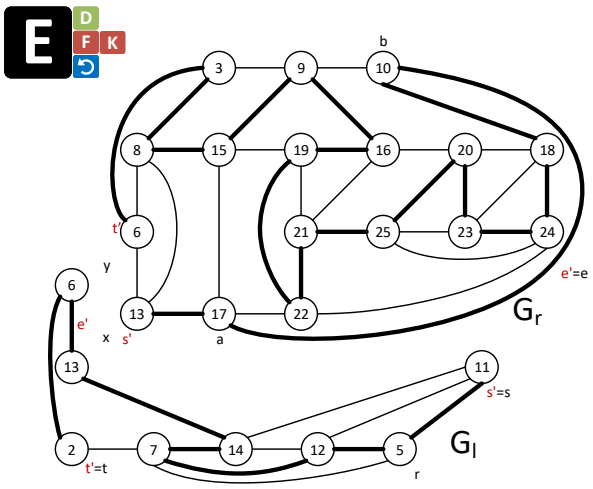
Split: no separation pair exists leading to another Type II reduction where the removed path P_{sa} only consists of node 1 this time. Here again, no other subgraphs are split from component C_b and the recursive call is made only over the induced subgraph G_b.

Merge: the combined Hamiltonian path is a merge of P_{sb} and Hamitonian path for G_b.



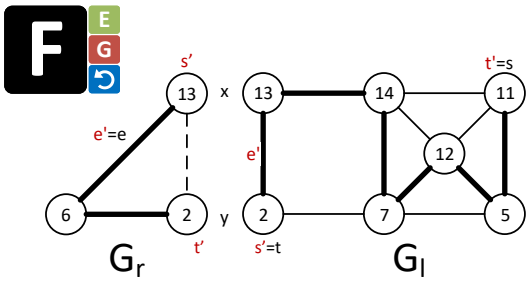
Split: for the third consecutive time, no vertical separation pair is found so we repeat a Type II reduction with P_{sa} only consisting of node 4. The recursive call is made only over the induced subgraph G_b after the s, t, and e are redefined.

Merge: the combined Hamiltonian path is a merge of P_{sb} consisting of node 1 and G_b's.



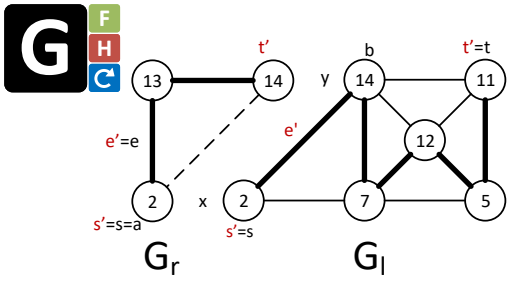
Split: a separation pair $(13, 6)$ is found leading to a Type I reduction case as the one described in Subsection 3.2.1. Graph splits in G_l and G_r and the s, t, e tuple get redefined for each of the subgraphs before recursive calls in Steps F and K, respectively.

Merge: by combining the Hamiltonian paths of G_l and G_r, and discarding $e \in E(G_l)$, we obtain the Hamiltonian path for this graph before split.



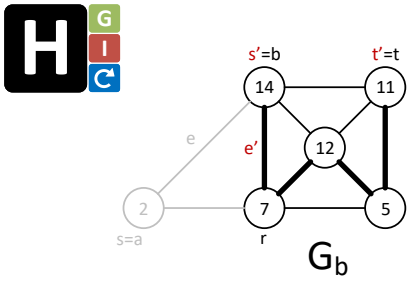
Split: separation pair $(13, 2)$ leads to Type I reduction where G_r results in a trivial case so there is only recursive call for G_l. Note the order inversion of s' and t' so $b \neq t$ in the next iteration.

Merge: combine the Hamiltonian paths computed for G_l and G_r; and discard arc $e \in E(G_l)$.



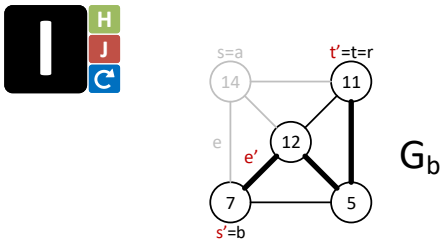
Split: separation pair $(2, 14)$ leads to Type I reduction with a trivial G_r and recursive call to G_l.

Merge: $G_l + G_r$ paths discarding arc $e \in E(G_l)$ results in the combined Hamiltonian path.



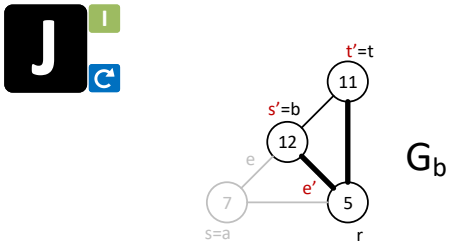
Split: no separation pair leads to a Type II reduction with P_{sa} consisting only of node 2 and no further splits in C_b so we only call recursively G_b.

Merge: P_{sb} added to path in G_b produce the combined Hamiltonian path of the graph.



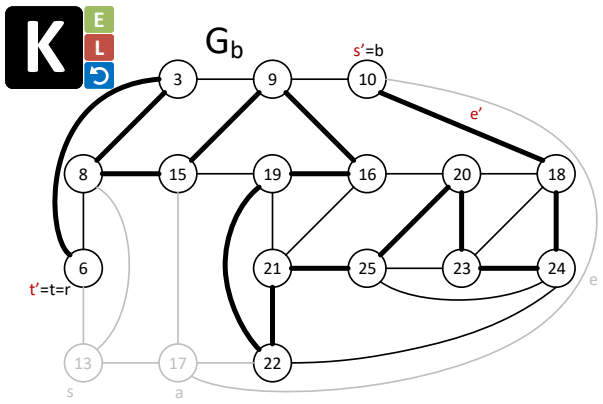
Split: another Type II reduction with P_{sa} consisting only of node 14 and no further splits of C_b so we only call recursively for G_b.

Merge: P_{sb} added to path in G_b produce the combined Hamiltonian path of the graph.



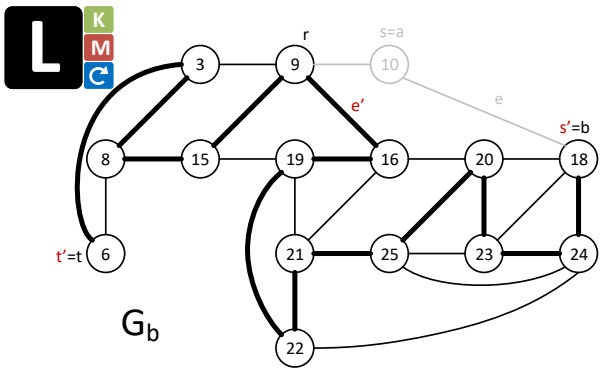
Split: a Type II reduction with $P_{sa} = \{7\}$ leads to a trivial case for the resulted subgraph G_b.

Merge: by combining path P_{sb} and trivial path found in G_b subgraph.



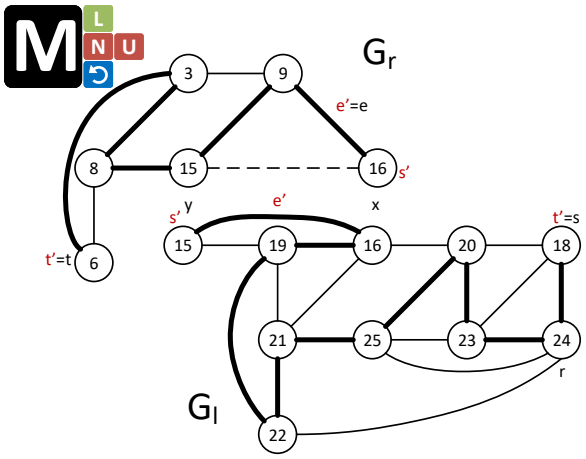
Split: no separation pair leads to a Type II reduction. After removing the path $P_{sa} = \{13, 17\}$ we have a single component with no further separation pairs in the induced subgraph G_b.

Merge: the combined Hamiltonian path adding back the path P_{sb} and the Hamitonian path of G_b.



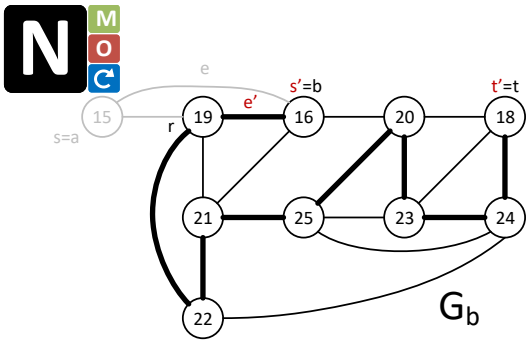
Split: no separation pair again leads to another Type II reduction with $P_{sa} = \{10\}$. Since no other subgraphs split from component C_b, there is only one recursive call for the induced subgraph G_b.

Merge: the combined Hamiltonian path is resulted from the merging of P_{sb} and the Hamitonian path of G_b.



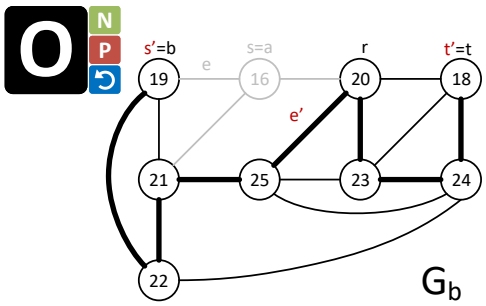
Split: a separation pair $(16, 15)$ leads to a Type I reduction with G_l and G_r being not trivial cases with recursive calls in Steps N and U, respectively.

Merge: $G_l + G_r$ paths minus the arc $e \in E(G_l)$ results in the combined Hamiltonian path.



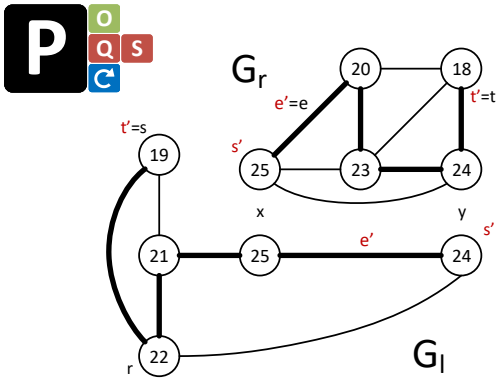
Split: Type II reduction with $P_{sa} = \{14\}$ with no further splits in C_b. Therefore, only one recursive call recursively for G_b.

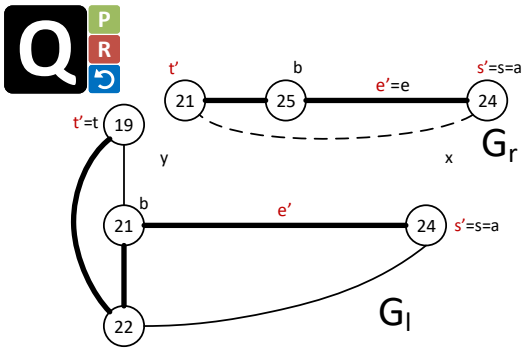
Merge: P_{sb} and the path in G_b produce the combined Hamiltonian path of the graph.



Split: Type II reduction with $P_{sa} = \{16\}$ with no further splits in C_b. Therefore, only one recursive call recursively for G_b.

Merge: P_{sb} and Hamiltonian path in G_b give us the combined path.



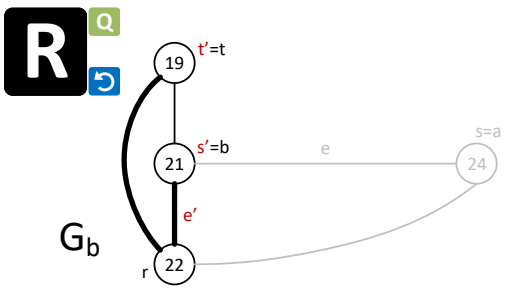


Split: vertical separation pair $(25, 24)$ leads to the non-trivial subgraphs G_l and G_r.

Merge: paths obtained from the recursive calls for each subgraph minus the virtual edge in G_l produce the combined Hamiltonian path.

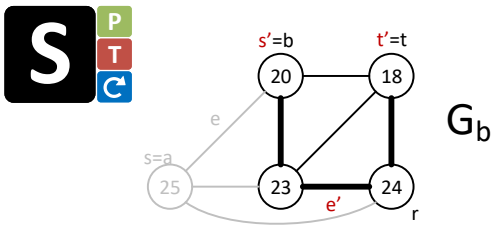
Split: vertical separation pair $(24, 21)$ leads to G_l and a trivial subgraph G_r.

Merge: paths obtained from the recursive calls for each subgraph minus the virtual edge $e = (24, 21)$ outlines the Hamiltonian path.



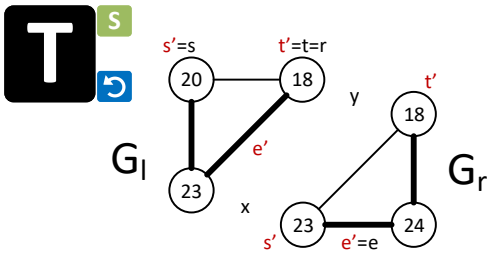
Split: no separation pair causes a Type II reduction with $P_{sa} = \{24\}$ and G_b with no further splits.

Merge: P_{sb} and the trivial path in G_b produce the combined Hamiltonian path of the graph.



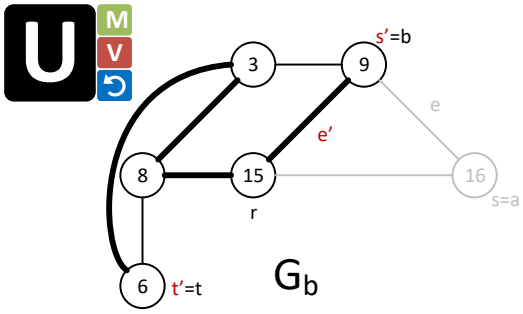
Split: a Type II reduction with $P_{sa} = \{25\}$. The induced subgraph G_b has no further splits.

Merge: P_{sb} and the trivial path in G_b produce the combined Hamiltonian path of the graph.



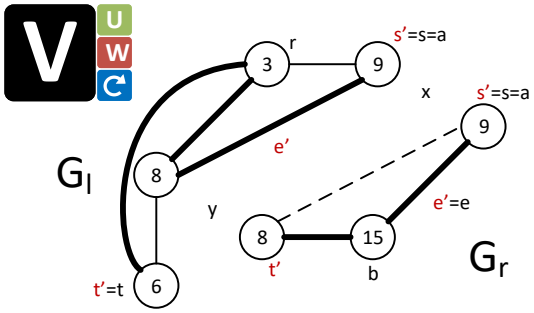
Split: vertical separation pair $(23, 18)$ leads to trivial subgraphs G_l and G_r after a Type I reduction.

Merge: paths from the trivial subgraphs outlines the Hamiltonian path after removing $e = (23, 18)$.



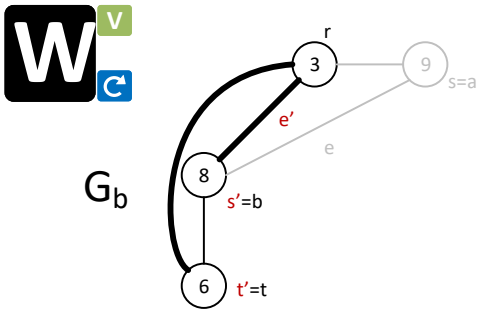
Split: a Type II reduction with $P_{sa} = \{16\}$. Induced subgraph G_b presents no further splits.

Merge: P_{sb} and the path in G_b produce the Hamiltonian path of the entire graph.



Split: vertical separation pair $(9, 8)$ leads to subgraph G_l and trivial G_r after a Type I reduction.

Merge: $G_l + G_r$ paths after removing $e = (9, 8)$ produce the Hamiltonian path of the entire graph.



Split: Type II reduction with $P_{sa} = \{9\}$. The induced subgraph G_b implies a trivial case.

Merge: P_{sb} and trivial path in G_b produce the Hamiltonian path of the graph.

Bibliography

[1] Bribiesca, E., Guzmán, A., and Martínez, L. A., "Enclosing trees," *Pattern Analysis and Applications* **15**, 1–17 (September 2012).

[2] Ansaldi, S., Floriani, L. D., and Falcidieno, B., "Geometric modeling of solid objects by using a face adjacency graph representation," *ACM SIGGRAPH Computer Graphics* **19**(3), 131–139 (1985).

[3] Diestel, R., [*Graph Theory*], vol. 173 of *Graduate Texts in Mathematics*, Springer, New York, 3rd ed. (August 2005).

[4] Euler, L., "Elementa doctrinae solidorum. – Demonstratio nonnullarum insignium proprietatum, quibus solida hedris planis inclusa sunt praedita," *Novi comment acad. sc. imp. Petropol.* **NA**(4), 109–140–160 (1752-3).

[5] "Twenty Proofs of Euler's Formula." http://www.ics.uci.edu/~eppstein/junkyard/euler/ (2017). Accessed: 2017-08-06.

[6] Bermond, J., [*Hamiltonian graphs*], Selected Topics in Graph Theory, Academic Press, London (1978).

[7] "Sir William Hamilton's Icosian Game and Traveller's Dodecahedron puzzle." http://puzzlemuseum.com/month/picm02/200207icosian.htm (2017). Accessed: 2017-07-23.

[8] "Icosian Game – from Wolfram MathWorld." http://mathworld.wolfram.com/IcosianGame.html (2017). Accessed: 2017-07-23.

[9] Gould, R. J., "Updating the Hamiltonian problem – A survey," *Journal of Graph Theory* **15**, 121–157 (June 1991).

[10] Gould, R. J., "Advances on the Hamiltonian problem – A survey," (2002).

[11] Gould, R. J., "Recent advances on the Hamiltonian problem: Survey III," *Graphs and Combinatorics* **30**, 1–46 (January 2014).

[12] Dirac, G., "Some theorems on abstract graphs," *Proceedings of the London Mathematical Society* **s3-2**(1), 69–81 (1952).

[13] Ore, O., "A note on hamiltonian circuits," *The American Mathematical Monthly* **67**, 55 (January 1960).

[14] Bondy, J. A. and Chvátal, V., "A method in graph theory," *Discrete Math* **15**, 111–135 (January 1976).

[15] Chvátal, V. and Erdös, P., "A note on hamiltonian circuits," *Discrete Mathematics* **2**(2), 111–113 (1972).

[16] Goodman, S. and Hedetniemi, S., "Sufficient conditions for a graph to be Hamiltonian," *Journal of Combinatorial Theory, Series B* **16**, 175–180 (June 1974).

[17] Tutte, W. T., "A theorem on planar graphs," *Transactions of the American Mathematical Society* **82**, 99–116 (May 1956).

[18] Tutte, W. T., "Bridges and Hamiltonian circuits in planar graphs," *Aequationes Mathematicae* **15**(1), 1–33 (1977).

[19] Whitney, H., "A theorem on graphs," *Ann. Math.* **32**, 378–390 (1931).

[20] Plummer, M. D., "Problem in Infinite and finite sets, a. hajnal, r. rado, and v.t. sós," *Colloquia Mathematica Societatis J. Bolyai 10* **III**, 1549–1550 (1975).

[21] Thomassen, C., "A theorem on paths in planar graphs," *Journal of Graph Theory* **7**(2), 169–176 (1983).

[22] Gouyeau-Beauchamps, D., "The Hamiltonian circuit problem is polynomial for 4-connected planar graphs," *SIAM Journal on Computing* **11**(3), 529–539 (1982).

[23] Chiba, N. and Nishizeki, T., "The hamiltonian cycle problem is linear-time solvable for 4-connected planar graphs," *Journal of Algorithms* **10**, 187–211 (June 1989).

[24] Graham, R. L. and Hell, P., "On the History of the Minimum Spanning Tree Problem," *Annals of the History of Computing* **7**, 43–57 (January 1985).

[25] Kruskal, J. B., "Digital representation schemes for 3d curves," *Proceedings of the American Mathematical Society* **7**(1), 48–50 (1956).

[26] Prim, R. C., "Shortest connection networks and some generalizations," *The Bell System Technical Journal* **36**, 1389–1401 (November 1957).

[27] Hart, W. E., Goldbaum, M., Côté, B., Kube, P., and Nelson, M. R., "Measurement and classification of retinal vascular tortuosity," *International Journal of Medical Informatics* **53**, 239–252 (February 1999).

[28] Zhang, T. and Nagy, G., "Surface tortuosity and its application to analyzing cracks in concrete," in [*Proceedings of the 17th International Conference on Pattern Recognition (ICPR 2004)*], **2**, 851–854 (August 2004).

[29] Grisan, E., Foracchia, M., and Ruggeri, A., "A novel method for the automatic grading of retinal vessel tortuosity," *IEEE Transactions on Medical Imaging* **27**, 310–319 (March 2008).

[30] Le, L. H., Zhang, C., Ta, D., and Lou, E., "Measurement of tortuosity in aluminum foams using airborne ultrasound," *Ultrasonics* **50**, 1–5 (January 2010).

[31] Bribiesca, E., "A measure of tortuosity based on chain coding," *Pattern Recognition* **46**, 716–724 (March 2013).

[32] Pearson, R. M., "Optometric grading scales for use in everyday practice," *Optometry Today* **43**(20) (2003).

[33] Mächler, M., "Very smooth nonparametric curve estimation by penalizing change of curvature," Tech. Rep. Technical Report 71, ETH Zurich, ETH Zurich (May 1993).

[34] Patasius, M., Marozas, V., Lukosevicius, A., and Jegelevicius, D., "Evaluation of tortuosity of eye blood vessels using the integral of square of derivative of curvature, prague," *EMBEC'05: proceedings of the 3rd IFMBE European Medical and Biological Engineering Conference* **11**(20), 1–4 (2005).

[35] ISO 468:1982, "Surface roughness – Parameters, their values and general rules for specifying requirements," standard, International Organization for Standardization (Aug 1982).

[36] Freeman, H., "On the encoding of arbitrary geometric configurations," *IRE Transactions on Electronic Computers* **EC-10**(2), 260–268 (1961).

[37] Freeman, H., "Computer processing of line-drawing images," *ACM Computing Surveys* **6**, 57–97 (March 1974).

[38] Guzman-Arenas, A., "Canonical shape description for 3-D stick bodies," Tech. Rep. ACA-254-87, Microelectronics and Computer Technology Corporation, Austin TX (July 1987).

[39] Jonas, A. and Kiryati, N., "Digital representation schemes for 3d curves," *Pattern Recognition* **30**(11), 1803–1816 (1997).

[40] Bribiesca, E., "A chain code for representing 3D curves," *Pattern Recognition* **33**, 755–765 (May 2000).

[41] Bribiesca, E., "3D-curve representation by means of a binary chain code," *Mathematical and Computer Modelling* **40**, 285–295 (August 2004).

[42] Martínez, L., Bribiesca, E., and Guzman-Arenas, A., "Chain coding representation of voxel-based objects with enclosing, edging and intersecting trees," *Pattern Analysis and Applications* **20**, 825–844 (Mar 2016).

[43] Artzy, E. and Herman, G. T., "Boundary detection in 3-dimensions with a medical application," *ACM SIGGRAPH Computer Graphics* **15**(2), 92–123 (1981).

[44] Lorensen, W. E. and Cline, H. E., "Marching cubes: A high resolution 3D surface construction algorithm," *ACM Siggraph Computer Graphics* **21**, 163–169 (July 1987).

[45] Nielson, G. M. and Hamann, B., "The asymptotic decider: resolving the ambiguity in marching cubes," in [*Proceedings of the IEEE Conference on Visualization (Visualization '91)*], 83–91 (October 1991).

[46] Thomas, R. and Yu, X. X., "4-connected projective-planar graphs are Hamiltonian," *Journal of Combinatorial Theory, Series B* **62**, 114–132 (September 1994).

[47] Grunbaum, B., "Polytopes, graphs, and complexes," *Bulletin of the American Mathematics Society* **76**(6), 1131–1201 (1970).

[48] Nash-Williams, N. A., [*Unexplored and semi-explored territories in graph theory*], Press, Academic (1973).

[49] Sagols, F., "Hamiltonian representation of vox-solids," *Computación y Sistemas* **2**(4), 213–217 (1999).

[50] Nishizeki, T. and Chiba, N., [*Planar graphs: Theory and algorithms*], vol. 32 of *Annals of Discrete Mathematics*, Elsevier Science Ltd (1988).

[51] Hopcroft, J. and Tarjan, R., "Dividing a graph into triconnected components," *SIAM Journal on Computing* **2**(3), 135–158 (1973).

[52] Cayley, A., "A theorem on trees," *Quarterly Journal of Pure and Applied Mathematics* **23**(23), 376–378 (1889).

[53] Galvez, J. M. and Canton, M., "Normalization and shape recognition of three-dimensional objects by 3D moments," *Pattern Recognition* **26**, 667–681 (May 1993).

[54] Bribiesca, E. and Guzmán, A., "How to describe pure forms and how to measure differences in shapes using shape numbers," *Pattern Recognition* **12**(2), 101–112 (1980).

[55] Meagher, D. J. R., "Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-D objects by computer," Tech. Rep. IPL-TR-80-111, Image Processing Laboratory, Electrical and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, New York (1980).

[56] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., [*Introduction to algorithms*], MIT Press, Cambridge, MA, 2nd ed. (2001).

[57] Bribiesca, E., "A method for representing 3D tree objects using chain coding," *Journal of Visual Communication and Image Representation* **19**, 184–198 (April 2008).

[58] Flin, F., Brzoska, J.-B., Coeurjolly, D., Pieritz, R. A., Lesaffre, B., Coléou, C., Lamboley, P., Teytaud, O., Vignoles, G. L., and Delesse, J.-F., "Adaptive estimation of normals and surface area for discrete 3-d objects: application to snow binary data from x-ray tomography," *IEEE Transactions on Image Processing* **14**(5), 585–596 (2005).

[59] Bribiesca, E., "An easy measure of compactness for 2d and 3d shapes," *Pattern Recognition* **41**(2), 543–554 (2008).

[60] Röttger, S., "The volume library." http://lgdv.cs.fau.de/External/vollib/ (2006). Accessed: 2013-05-09.

[61] Gasca Soto, M. d. l. L., *Hamiltonicidad en familias de gráficas geométricas*, PhD thesis, Posgrado en Ciencias Matemáticas, Facultad de Ciencias, Universidad Nacional Autónoma de México (2009).

[62] Schettini, N., Jaroszeski, M. J., West, L., and Saddow, S. E., [*Silicon Carbide Biotechnology, Chap.5- Hemocompatibility Assessment of 3C-SiC for Cardiovascular Applications*], Elsevier, Oxford (2012).