

Ricardo Dutra da Silva

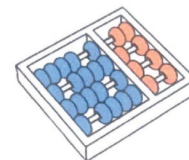
**“Discrete Morse Complex of Images:  
Algorithms, Modeling and Applications”**

***“Complexo Discreto de Morse para Imagens:  
Algoritmos, Modelagem e Aplicações”***

**CAMPINAS  
2013**







University of Campinas  
Institute of Computing

*Universidade Estadual de Campinas  
Instituto de Computação*

Ricardo Dutra da Silva

**“Discrete Morse Complex of Images:  
Algorithms, Modeling and Applications”**

Supervisor: Prof. Dr. Hélio Pedrini  
*Orientador(a):*

***“Complexo Discreto de Morse para Imagens:  
Algoritmos, Modelagem e Aplicações”***

PhD Thesis presented to the Post Graduate Program of the Institute of Computing of the University of Campinas to obtain a PhD degree in Computer Science.

*Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Doutor em Ciência da Computação.*

THIS VOLUME CORRESPONDS TO THE FINAL VERSION OF THE THESIS DEFENDED BY RICARDO DUTRA DA SILVA, UNDER THE SUPERVISION OF PROF. DR. HÉLIO PEDRINI.

*ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA TESE DEFENDIDA POR RICARDO DUTRA DA SILVA, SOB ORIENTAÇÃO DE PROF. DR. HÉLIO PEDRINI.*

Supervisor's signature / *Assinatura do Orientador(a)*

CAMPINAS  
2013

Ficha catalográfica  
Universidade Estadual de Campinas  
Biblioteca do Instituto de Matemática, Estatística e Computação Científica  
Ana Regina Machado - CRB 8/5467

Si38d Silva, Ricardo Dutra da, 1982-  
Discrete Morse complex of images : algorithms, modeling and applications /  
Ricardo Dutra da Silva. – Campinas, SP : [s.n.], 2013.

Orientador: Hélio Pedrini.  
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de  
Computação.

1. Morse, Teoria de. 2. Processamento de imagens. 3. Complexo celular. 4.  
Topologia computacional. 5. Homologia persistente. I. Pedrini, Hélio, 1963-. II.  
Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

**Título em outro idioma:** Complexo discreto de Morse para imagens : algoritmos, modelagem e aplicações

**Palavras-chave em inglês:**

Morse theory

Image processing

Cell complex

Computational topology

Persistent homology

**Área de concentração:** Ciência da Computação

**Titulação:** Doutor em Ciência da Computação

**Banca examinadora:**

Hélio Pedrini [Orientador]

João Paulo Papa

Helton Hideraldo Bísaro

Jorge Stolfi

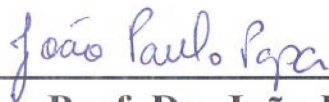
Neucimar Jerônimo Leite

**Data de defesa:** 05-11-2013

**Programa de Pós-Graduação:** Ciência da Computação

# TERMO DE APROVAÇÃO

Tese Defendida e Aprovada em 05 de novembro de 2013, pela Banca  
examinadora composta pelos Professores Doutores:



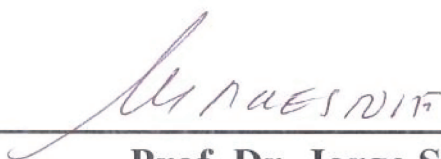
---

**Prof. Dr. João Paulo Papa**  
DCC / UNESP



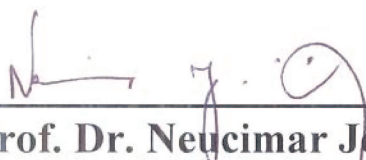
---

**Prof. Dr. Helton Hideraldo Bísaro**  
EACH / USP



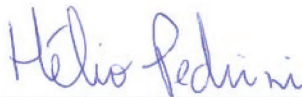
---

**Prof. Dr. Jorge Stolfi**  
IC / UNICAMP



---

**Prof. Dr. Neucimar Jerônimo Leite**  
IC / UNICAMP



---

**Prof. Dr. Hélio Pedrini**  
IC / UNICAMP



# Discrete Morse Complex of Images: Algorithms, Modeling and Applications

Ricardo Dutra da Silva<sup>1</sup>

November 05, 2013

## Examiner Board / *Banca Examinadora*:

- Prof. Dr. Hélio Pedrini (Supervisor / *Orientador*)
- Prof. Dr. João Paulo Papa  
Departamento de Computação - UNESP/Bauru
- Prof. Dr. Helton Hideraldo Bísaro  
Escola de Artes, Ciências e Humanidades - USP
- Prof. Dr. Jorge Stolfi  
Instituto de Computação - UNICAMP
- Prof. Dr. Neucimar Jerônimo Leite  
Instituto de Computação - UNICAMP

---

<sup>1</sup>This work was partially supported by São Paulo Research Foundation – FAPESP (Grants 2009/10627-5 and 2011/22749-8), National Counsel of Technological and Scientific Development – CNPq (Grant 307113/2012-4) and Coordination for Enhancement of Higher Education Personnel – CAPES (Grant 01-P-04388/2010).



# Abstract

The Morse theory is important for studying the topology of scalar functions such as elevation of terrains and data from physical simulations, which relates the topology of a function to critical points. The smooth theory has been adapted to discrete data through constructions such as the Morse-Smale complexes and the discrete Morse complex. Morse complexes have been applied to image processing, however, there are still challenges involving algorithms and practical considerations for computation and modeling of the complexes. Morse complexes can be used as means of defining the connectedness of interest points in images. Usually, interest points are considered as independent elements described by local information. Such an approach has its limitations since local information may not suffice for describing certain image regions. Minimum and maximum points are widely used as interest points in images, which can be obtained from Morse complexes, as well as their connectivity in the image space. This thesis presents an algorithmic and data structure driven approach to computing the discrete Morse complex of 2-dimensional images. The construction is optimal and allows easy manipulation of the complex. Theoretical and applied results are presented to show the effectiveness of the method. Applied experiments include the computation of persistent homology and hierarchies of complexes over elevation terrain data. Another contribution is the proposition of a topological operator, called Local Morse Context (LMC), computed over Morse complexes, for extracting neighborhoods of interest points to explore the structural information in images. The LMC is used in the development of a matching algorithm, which helps reducing the number of incorrect matches between images and obtaining a confidence measure of whether a correspondence is correct or incorrect. The approach is tested in synthetic and challenging underwater stereo pairs of images, for which available methods may obtain many incorrect correspondences.





# Resumo

A Teoria de Morse é importante para o estudo da topologia em funções escalares como elevação de terrenos e dados provenientes de simulações físicas, a qual relaciona a topologia de uma função com seus pontos críticos. A teoria contínua foi adaptada para dados discretos através de construções como os complexos de Morse-Smale e o complexo discreto de Morse. Complexos de Morse têm sido aplicados em processamento de imagens, no entanto, ainda existem desafios envolvendo algoritmos e considerações práticas para a computação e modelagem dos complexos para imagens. Complexos de Morse podem ser usados como um meio de definir a conexão entre pontos de interesse em imagens. Normalmente, pontos de interesse são considerados como elementos independentes descritos por informação local. Tal abordagem apresenta limitações uma vez que informação local pode não ser suficiente para descrever certas regiões da imagem. Pontos de mínimo e máximo são comumente utilizados como pontos de interesse em imagens, os quais podem ser obtidos a partir dos complexos de Morse, bem como sua conectividade no espaço de imagem. Esta tese apresenta uma abordagem dirigida por algoritmos e estruturas de dados para computar o complexo de Morse discreto em imagens bidimensionais. A construção é ótima e permite fácil manipulação do complexo. Resultados teóricos e experimentais são apresentados para mostrar que o método é eficaz. Experimentos realizados incluem a computação de homologia persistente e hierarquias de complexos sobre dados de elevação de terrenos. Outra contribuição é a proposição de um operador topológico, chamado Contexto Local de Morse, computado sobre complexos de Morse, para extrair vizinhanças de pontos de interesse para explorar a informação estrutural de imagens. O contexto local de Morse é usado no desenvolvimento de um algoritmo que auxilia a redução do número de casamentos incorretos entre pontos de interesse e na obtenção de uma medida de confiança para tais correspondências. A abordagem proposta é testada em pares de imagens sintéticas e de imagens subaquáticas, para as quais métodos existentes podem obter muitas correspondências incorretas.



# Acknowledgements

I would like to thank my advisor, H lio Pedrini, for his support through many years. H lio has kindly guided me and helped me build my research skills. The Institute of Computing of the University of Campinas (UNICAMP) for the great environment, the teachers and the efficient staff.

I would like to thank Bernd Hamann for receiving me at the University of California, Davis (UCDAVIS), for his collaboration and his always promptly attention. Also, from UCDAVIS, Jesus Pulido, with whom I closely worked, and the other colleagues from the Institute for Data Analysis and Visualization.

This work could not be performed without the support of Funda  o de Amparo   Pesquisa do Estado de S o Paulo (FAPESP) that offered me a scholarship while in Brazil (2009/10627-5) and in the United States of America (2011/22749-8). Also, Coordena  o de Aperfei amento de Pessoal de N vel Superior (CAPES) for the valuable support at the beginning of my endeavor.

Many thanks to my great colleagues and friends during all these years that made my way a mostly pleasant one: Manuela Swerts Batista Leite, Rodrigo Minetto, William Robson Schwartz, Juliana de Santi, Roberto Pereira, Elisa de C ssia Silva Rodrigues, Maria Antonieta Rocha Alves.

At last, but certainly not least, I would like to thank my parents, Edison and Sonia, the greatest supporters. My brave and caring grandmother, Eunice. All my close and beloved relatives: Edison Jr., Luiza, Eneci, Gustavo, Guilherme, Isadora and Carlos. Thank you!



# Contents

<b>Abstract</b>	<b>ix</b>
<b>Resumo</b>	<b>xi</b>
<b>Acknowledgements</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem . . . . .	1
1.2 Motivation . . . . .	2
1.3 Objectives . . . . .	4
1.4 Contributions . . . . .	5
1.5 Thesis Organization . . . . .	6
<b>2 Cell Complexes and Digital Images</b>	<b>7</b>
2.1 Basic Concepts on Cell Complexes . . . . .	7
2.2 The Quad-Edge Data Structure . . . . .	9
2.3 Images Treated as Cell Complexes . . . . .	10
2.3.1 Implementation . . . . .	11
2.4 Topological Operations . . . . .	15
<b>3 Discrete Morse Theory</b>	<b>17</b>
3.1 Computation of the Discrete Morse Vector Field . . . . .	19
<b>4 Discrete Morse Complex of Images</b>	<b>23</b>
4.1 Algorithms . . . . .	24
4.1.1 Some Considerations . . . . .	24
4.1.2 Extraction of $QV$ -Paths . . . . .	26
4.1.3 Disentangled Complex . . . . .	39
4.1.4 Simplified Morse Complex . . . . .	43
4.2 Experimental Results . . . . .	46



<b>5</b>	<b>Neighborhood-of-Interest Points Using the Morse Complex</b>	<b>55</b>
5.1	Local Morse Context . . . . .	58
5.1.1	Morse Complex and Critical Cells . . . . .	58
5.1.2	Local Morse Context . . . . .	59
5.2	Image Matching Using the LMC . . . . .	61
5.2.1	Matching of LMCs . . . . .	61
5.2.2	Image Match . . . . .	63
5.3	Experimental Results . . . . .	64
5.3.1	Datasets and Ground Truth . . . . .	65
5.3.2	Evaluation Metrics . . . . .	67
5.3.3	Method Setup . . . . .	67
5.3.4	Matching Results . . . . .	71
<b>6</b>	<b>Conclusions and Future Work</b>	<b>75</b>
	<b>Bibliography</b>	<b>78</b>





# List of Tables

- 4.1 Images used in the experiments. The second column shows the dimensions of the images and the columns three to five shows the number of cells of each dimension in the respective cell complexes. . . . . 47
- 4.2 Number of critical points of the images. . . . . 48
- 5.1 Best scores for each order of LMC. For each order of LMC, the best score was chosen, that is, a correspondence scored with this or a greater value is mostly probable to be correct (fifth column) and the probability of finding such a correspondence is also high (fourth column). Higher orders perform better in this case. We find out that an order of 3 and score 11 are good choices for measuring the confidence of correspondences. . . . . 70



# List of Figures

1.1	Set of points (a) forming a space and (b) with connectivity (neighborhood information) forming a topological space. . . . .	2
1.2	A growing space defined by level sets characterizing the topology of a function. . . . .	3
2.1	Examples of cells of dimension up to two. Figure adapted from [36]. . . . .	8
2.2	Examples of boundaries of the cells in Figure 2.1. . . . .	8
2.3	Sets of cell that satisfy (a) and that do not satisfy (b) the cell complex conditions. Figure adapted from [100]. Figure (b) has a cell without a face and two cells for which the intersection is not a shared face. . . . .	9
2.4	The quad-edge data structure is composed of four directed 1-cells: $e$ , $\text{InvRot}(e)$ , $\text{Sym}(e)$ and $\text{Rot}(e)$ . Figure adapted from [37]. . . . .	10
2.5	The output of the main functions applied to a 1-cell $e$ . Figure adapted from [37]. . . . .	11
2.6	Two different ways of modeling an image as a cell complex. The two complexes are duals such that in one (a) pixels are 2-cells and in another (b) the pixels are 0-cells. . . . .	11
2.7	Cells created and/or adjusted at each iteration of Algorithm 2.1. . . . .	13
2.8	2-cells created at each step of the algorithm. . . . .	14
2.9	Example of star, closure and link. The operations are applied to the 0-cells in the center of the highlighted areas. In the case of the closure, first a star is applied to the 0-cell and then the closure is applied to the result of the star. . . . .	15
3.1	Example of a function that is (a) a valid Morse function and (b) a function that is not a Morse function. Figure adapted from [51]. . . . .	18
3.2	Example of critical cells (saddle and minimum) of a discrete Morse function. Figure adapted from [51]. . . . .	18
3.3	Example of a discrete Morse vector field. All cells are paired except for critical cells. . . . .	18
3.4	Example of a $V$ -path. . . . .	19



3.5	Six level subcomplexes of a complex $K$ . . . . .	20
3.6	Results of the computation of the vector field of a complex at different iterations of the Algorithm 3.1. The resulting vector field contains all cells paired except for critical cells. . . . .	22
4.1	Complex $K$ with $M$ placed over it. The 0-cells of $M$ correspond to any type of cells $K$ while the 1-cells of $M$ explicitly model the boundary relations between cells. The 0-cells of $M$ inherit the properties of cells in $K$ . The index $p$ of a cell $v^p$ in $M$ is related to the dimension of the corresponding cell in $K$ . Also, the 0-cells can be regular or critical. . . . .	25
4.2	Given a critical 1-cell in $V$ (a), Algorithm 4.1 creates the $QV$ -paths out of it (b)-(e) and arranges them in the ring of edges out of the vertex related to the 1-cell (f). . . . .	28
4.3	Expanding triple of a $(0, 1)$ -path. . . . .	29
4.4	Expanding triple of a $(1, 2)$ -path. . . . .	30
4.5	One step expanded $QV$ -paths obtained with the expanding triples of the initial $(0, 1)$ - and $(1, 2)$ -paths of Figures 4.2b and 4.2d. . . . .	33
4.6	Example of a 0-cell shared by three $V$ -paths. . . . .	34
4.7	Example of a 2-cell shared by two $V$ -paths. . . . .	34
4.8	Example of $QV$ -paths that merge. . . . .	36
4.9	Example of $QV$ -paths that branch. . . . .	36
4.10	Examples of knots for $(0, 1)$ - and $(1, 2)$ -paths in a complex $M$ . . . . .	40
4.11	Example of untangling knots of a merged path. . . . .	42
4.12	Simplification kernel for a subpath of a $QV$ -path. . . . .	44
4.13	Simplification of kernel in Figure 4.12 and the acquisition of a new kernel(a). The simplified path is shown in (b) connecting critical cells. . . . .	45
4.14	The resulting discrete Morse complex. . . . .	47
4.15	Discrete Morse complex of the synthetic sinusoidal image. The 2-cells of the complex are formed by a minimum a maximum and two saddles. Minima and maxima are always adjacent to saddles. . . . .	48
4.16	Graphs showing for each 0- and 1-cycle their respective persistence in the sinusoidal image. The sorted persistence of the cycles grows exponentially, except for the last 0-cycle which has infinite persistence. . . . .	49



4.17	Persistence of cycles for real elevation terrain images. The most persistent cycles are present in the right portion of the graphics. It is possible to notice that the majority of cycles have a very small persistence value. That can be due to noise or non-relevant topological features. For instance, the number of cycles with persistence less than 10 in all graphs represents the greatest part of the features, which suggests that the persistence can be used for compression or noise removal. . . . .	50
4.18	Number of cycles for different level set of the image Death Valley. Each line shows the number of cycles with persistence greater than or equal to a specific value of persistence (1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024). .	51
4.19	Hierarchy of a discrete Morse complex for different persistence values. Only points with persistence larger than a persistence threshold are maintained in the complex. The complex is greatly simplified and topological noise is removed. As the hierarchy progresses, fine details and noise are removed. Close to a persistence threshold of 32, the valleys become almost free of noise whereas relevant features in the mountains are preserved in the complex.	52
4.20	Hierarchy for the Crater Lake image discrete Morse complex. The island in the middle of the lake, as well as other important features, stands out from noise in the complex as the hierarchy progresses. The persistence values range from 0 to 360 (except for one infinite persistent 0-cycle). . . . .	53
4.21	A hierarchy of discrete Morse complexes computed for a face image. When considering coarser levels of the hierarchy, features in the eyes, nose and mouth stand out from other features. The persistence ranges from 0 to 138.	54
5.1	Examples of incorrect matches computed due to regions of high similarity. The descriptor by itself is not able to discriminate between some regions. The zoomed regions show the pixel level texture similarities that produce close descriptors and consequently difficulties for the correspondence of interest points. . . . .	56
5.2	Examples of output matches obtained through SIFT method. Incorrect correspondences are filtered out, however, the number of correspondences is drastically reduced and many regions do not contain any paired points. .	57
5.3	Example of LoG and DMC of an image. . . . .	59
5.4	Example of computing the $LMC_2$ of a maximum critical cell $\alpha$ . Increasing the order augments the number of critical points in the set and expands the neighborhood. . . . .	60
5.5	Example of a pattern between two related critical cells. Missing or added critical cells are likely to occur. . . . .	61





5.6	Corresponding points between LMCs, the correspondences are represented with numbers. . . . .	63
5.7	Correspondences growing from initial seed set (a). The LMC locally guides the matches (b-c) until all interest points have correspondences established. . . . .	65
5.8	Examples of synthetic (a)-(c) and underwater images (d)-(f) used in our experiments. . . . .	66
5.9	LMC order influence on growing the number of matches. The lower orders have a higher recall value. Therefore, the LMC-based algorithms performs better if the neighborhood considered to grow matches is smaller, the search for new correspondences in related regions between images is more restrict. . . . .	68
5.10	Geometrical property for score where corresponding points in a pair of LMCs have similar distances to the horizontal lines. . . . .	69
5.11	Relation between the number of seeds and number of matches. The number of correctly corresponded points increases with the number of randomly chosen seeds, but converges near 20 seeds. . . . .	71
5.12	Average results for all the test images using HGV, HOG and SIFT descriptors. The recall of the LMC-based matching is higher, independent of the descriptor, when compared to the results obtained using the 1-NN matching. The conflicts of descriptors are avoided with the LMC neighborhood restriction, allowing to obtain more correct correspondences. . . . .	72
5.13	Examples of interest points correctly corresponded due to the use of the LMC (solid lines) but incorrectly corresponded by using the nearest neighbor approach (dashed lines). . . . .	73



# List of Algorithms

2.1	ImageToCellImage . . . . .	12
2.2	SetFirst1Cell . . . . .	13
2.3	SetSecond1Cell . . . . .	14
2.4	SetThird1Cell . . . . .	14
2.5	SetFourth1Cell . . . . .	14
2.6	Set2Cell . . . . .	15
3.1	ComputeVectorField . . . . .	21
4.1	InitVPaths . . . . .	27
4.2	Get01ExpandingTriple . . . . .	31
4.3	Get12ExpandingTriple . . . . .	32
4.4	Expand01Path . . . . .	32
4.5	Expand12Path . . . . .	33
4.6	Get01ExpandingTriple . . . . .	37
4.7	Get12ExpandingTriple . . . . .	37
4.8	ProcessQVPath . . . . .	38
4.9	RemoveQVPath . . . . .	39
4.10	ExtractQVPaths . . . . .	40
4.11	DisentangleKnot . . . . .	41
4.12	DisentangleComplex . . . . .	43
4.13	SimpKernel . . . . .	44
4.14	SimplifyKernel . . . . .	45
4.15	SimplifyComplex . . . . .	46
5.1	LMCImageMatching . . . . .	63



# Chapter 1

## Introduction

Computational topology has been studied for information visualization, image processing, fluid mechanics and structural biology. In this thesis, we focus on the computation and exploration of topological tools in image processing. In particular, we study the combinatorial representation of images by cell complexes and an important tool for acquiring and analyzing the topology of such spaces, the Morse theory. We introduce a new method for computing the complexes derived from the Morse theory and also present practical matters for coding the algorithms and the representations of such complexes. The features obtained by the Morse theory are then related to image features and we present a method for computing connections between these features.

### 1.1 Problem

Topology [68] is a branch of mathematics whose studies emphasize information such as connectivity and continuity of spaces. A space is usually formed by a set of points and, therefore, it has no structure at all.

In image processing and computer vision, features are commonly considered as independent elements defined or described by some local information. This is a limitation since there is no knowledge of the relationship between elements and of how they relate to form image functions as a whole.

Topological tools can be used to introduce information on connectivity and on how a space is formed. However, in order to use these tools in image processing and analysis, some questions must be answered. How to retrieve the topology of an image? How to represent the topological features and their connections? How to describe the topology and explore its connectedness?

## 1.2 Motivation

Computational topology is a field gaining importance for analyzing images at qualitative, structural and abstract levels [5, 11, 19, 28, 50, 73, 92]. In particular, the Morse theory [64] is a tool for studying the topology of functions. The Morse theory relates the topology of a smooth function with its critical points (maxima, saddles and minima in a 2-dimensional function) and decomposes a differentiable function in partitions of uniform flow characterizing the vector field of the function [26].

Topology makes a space more interesting by adding a connectivity so that points start to have a knowledge of other points in their neighborhood. Figure 1.1a shows points in a space, whereas Figure 1.1b shows a topological space. Clearly, space with connectivity carries more information than space without topology.

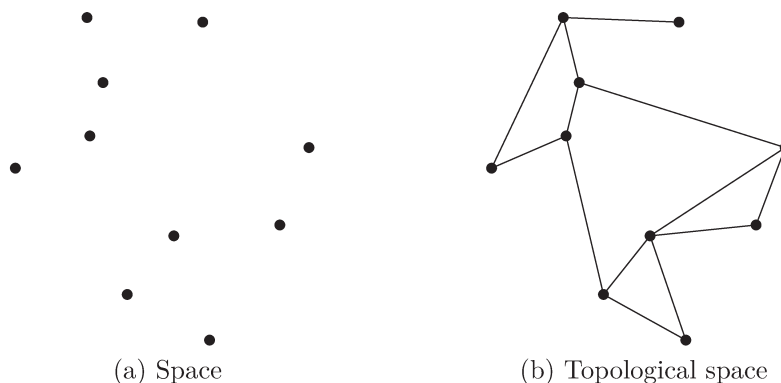


Figure 1.1: Set of points (a) forming a space and (b) with connectivity (neighborhood information) forming a topological space.

Figure 1.2 shows level sets of a function<sup>1</sup>  $f$ . The level set is a subset of points  $p$  with  $f(p) \leq v$ , such that, for increasing values of  $v$ , the level sets form a “growing” space. The contours generated by the boundaries of the components in each step have already been studied for partitions of terrains [12]. These are the origins of the Morse theory. The contours are determined by the geometry of the function and the relation between geometry and topology is the subject of the Morse theory.

A set of growing spaces, as in Figure 1.2, produces what is called a filtration. The filtration produces a history of topological changes and can be used to capture the structure of a function. This is the subject of study of the persistence homology [99], that computes a topological invariant of great importance in the latest advances related to computational topology [9, 10, 35, 40, 41, 93], including applications in image processing and computer vision [16, 27, 33].

---

<sup>1</sup>The function shown in the images was produced with MATLAB `peaks`.

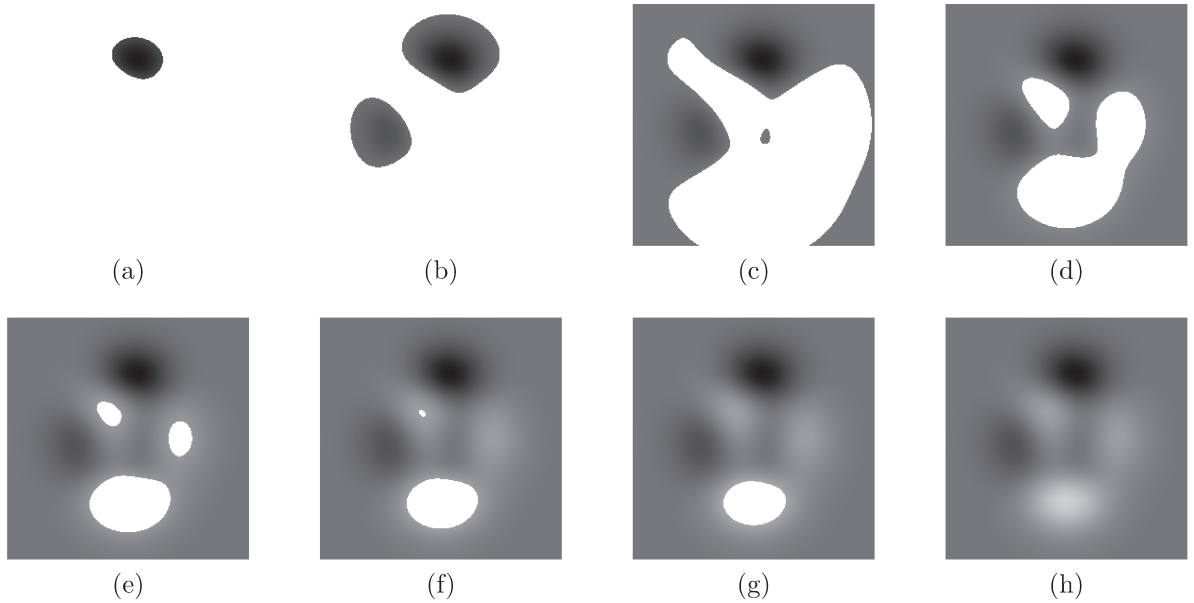


Figure 1.2: A growing space defined by level sets characterizing the topology of a function.

Several issues arise when transposing the topology tools to computational methods. The adaptation of the Morse theory does not encounter the same smooth notions for defining the basics on critical points and gradient flow. In many image processing problems, functions represent a space with scalar measures over it. This is, for instance, the case of elevation of terrains and functions derived from physical simulations. The domain of image functions is sampled and the continuous Morse theory can not be directly applied to it. The adaptation of the continuous theory to sampled data has been studied in works such as [6, 26, 99], which, based on the work of Banchoff [1], use the simulation of smooth notions to guide the computation and produce what is called Morse-Smale complex [99].

Such constructions present structural considerations that need to be dealt with and that occur due to the discrete nature of the data. An important case is that of saddle points. Saddles in the discrete adaptation can be found in more than one form. Another difficulty comes from the intersection of integral curves that introduce extra critical points.

The discrete Morse theory, formulated by Forman [30, 31], is an adaptation of the Morse theory to discrete structures. It has been explored in works such as King et al. [46] and Lewiner et al. [53, 54], which proposes algorithms for the computation of discrete Morse functions. Robins et al. [76] recently provided algorithms for computing discrete Morse complexes for image analysis purposes. However, there are also many practical issues that are not considered.

The problem of saddles does not occur in the discrete Morse complex since there is only one type. However, the  $V$ -paths (counterparts of integral curves in the smooth case)

still can be an issue. The data structures and their influences are not a primary concern in the prior studies. In Robins et al. [76], for example, discrete paths (integral curves) are only implicitly stored and, therefore, it is not possible to draw them correctly for visualizations.

In theoretical discussions, it is common to use adjacency matrices to model the complexes [36]. This kind of structure is clearly not efficient for storage and for the computation of topological operations, such as neighborhoods of cells. An adjacency list would be faster in such cases, but there are still some problems such as testing adjacent vertices or dynamically maintaining the structure when complexes are simplified. There are, however, efficient and well known data structures that can be used, such as the quad-edge [37].

Structures such as the quad-edge are ideal for subdivisions as the ones in 2-dimensional complexes. However, it may be difficult to adapt the existing algorithms for computing discrete Morse complexes directly. For example, the breadth-search based algorithm of Robins et al. [76] for computing discrete Morse complexes produces inconsistent subdivisions when using the quad-edge. In the quad-edge data structure, the traversal of edges adjacent to a vertex, in a counterclockwise fashion (or, alternatively, clockwise), needs to be consistently maintained, however, the breadth-search algorithm does not offer the necessary information to provide which edge should precede or succeed another edge in the traversal.

As we will show along this thesis, it is possible to compute the 2-dimensional discrete Morse complex and maintain the consistency of the quad-edge data structure.

### 1.3 Objectives

The topology theory may be difficult for computer scientists since its mathematical language is not studied in undergraduate courses and is a rare subject in graduate level. However, the problems related to topology arise in fields such as computational geometry, computer graphics, robotics, structural biology, and chemistry. We intend to introduce the basic mathematical concepts with examples so that it is possible to easily understand the ideas behind the concepts. Algorithms and data structures are also subject of our discussion so that the presented methods are more intuitive for computer scientists (Chapters 2 and 3).

We restrict our attention to the 2-dimensional case and tackle some important considerations for the computation and modeling of discrete Morse complexes of images (Chapter 4). We are particularly interested in algorithms and data structures for constructing the discrete Morse complex while allowing important topological computations such as persistence and simplification of topological features, as well as the visualization of the complexes.



As already mentioned,  $V$ -paths can be a problem in the computation of discrete Morse complexes. As we will show, paths can merge and branch, such that these cases need to be considered to produce visualizations and simplifications of topological features.

We base our approach on Forman's theory [30] and on the algorithm of Robins et al [76] to compute the discrete Morse vector field. We determine the paths between critical cells, in a discrete vector field, starting from saddles. It is shown that such an approach is optimal since it only processes cells that are in a path, opposed to the breadth-search that considers cells even if they are not in any path. It is also possible to deal with merging and branching cases properly and produce a consistent representation of the quad-edge model. We also present algorithms to dealing with such cases for visualization purposes. Finally, we model a complex so that topological computations such as homological persistence [4, 9, 18, 67, 93] and simplification of topological features [60, 75, 91] can be easily computed. Such computations are performed and evaluated over elevation terrain data.

A topological operator computed over Morse complexes, called Local Morse Context (LMC), is introduced as a way of computing neighborhoods of interest points to explore the structural information in images (Chapter 5). The LMC is used in the development of a matching algorithm that helps reducing the number of incorrect matches and obtaining a confidence measure of whether a correspondence is correct or incorrect.

The approach is tested for the correspondence of synthetic and specially challenging underwater stereo pairs of images, for which traditional methods present difficulties for finding correct correspondences.

## 1.4 Contributions

The specific contributions of this thesis are:

- an algorithmic approach to the construction and manipulation of discrete Morse complexes.
- an optimal algorithm for extracting a 2-dimensional discrete Morse complex from a discrete vector field.
- a model of the discrete Morse complex by means of a quad-edge data structure that allows:
  - visualization of the complexes.
  - computation of persistent Betti numbers.
  - simplification of the complex as used for topological noise removal and hierarchies of complexes.

- a general neighborhood relation which can be easily adapted to different applications.
- a method for improving correspondences of pairs of points between complex structured images.
- an approach to evaluating the confidence of such correspondences.

## 1.5 Thesis Organization

Since many concepts of the topology theory presented in the text comes from a mathematical field, computer scientists usually are not familiar with them. We review the main concepts used in our approach to provide a self contained text. In Chapter 2 and 3, we review topological and data structure concepts that are used in the remaining of the discussions. In Chapter 4, we introduce a new algorithm for computing the Morse complexes of images. Chapter 5 introduces a local method for extracting the neighborhood information from Morse complexes and its application to find correspondences between images. The conclusions of this thesis and directions for future work are presented in Chapter 6.

# Chapter 2

## Cell Complexes and Digital Images

We present in this chapter basic concepts on topology and cells complexes and how they can be employed to represent digital images. Cell complexes are a way of sampling functions that can be used in computers. The topological and cell complexes discussions are based on works such as [36, 61, 101], which may be referred for detailed theoretical explanations on the topics.

### 2.1 Basic Concepts on Cell Complexes

We start with some notions about spaces before introducing the type of space we are interested in, the cell complexes. A space is simply a set of points. In topology, a neighborhood notion is added to a space such that it is possible to determine the connectivity between points. A set of points  $X$  along with a definition for neighborhood informally defines a topological space. Usually, we are used to metric spaces, that is, a space with an associated metric that makes it possible to measure the distance between points and define neighborhoods. The Euclidean space is a metric space. For now, we are interested in topological spaces and some particular ones used in computations.

Important topological spaces used in computations are the manifolds and the cell complexes. We are particularly interested in 2-dimensional cell complexes and, therefore, we present their basics in the following.

A  $p$ -cell is the building block to define the discrete domain which is a cell complex. A  $p$ -cell has dimension  $p$ . The first three low-order  $p$ -cells are the 0-cells or nodes, 1-cells or edges and 2-cells or faces. Even though the secondary names are commonly used in computer science, we will avoid them. The concept of a face is a different one in topology, as we will state soon. Therefore, to be consistent with the topology and computational topology literature, we choose to use the primary names.

Figure 2.1 shows examples of  $p$ -cells up to dimension two. It can be noticed that a  $p$ -cell

has not a unique geometric shape, for instance, both the triangle and the quadrilateral of Figure 2.1c are 2-cells. We will drop the superscript of a  $p$ -cell  $\alpha^p$ , denoting it  $\alpha$ , whenever the dimension is clear from the context.

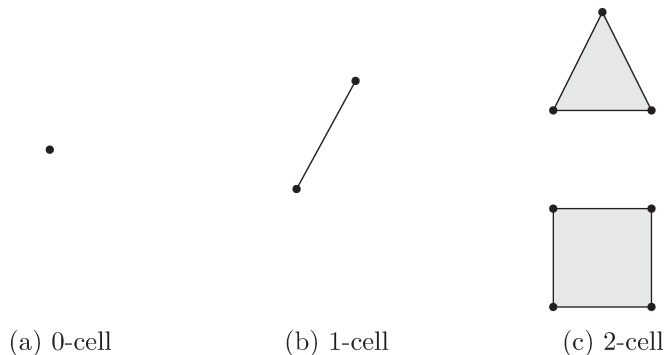


Figure 2.1: Examples of cells of dimension up to two. Figure adapted from [36].

The boundary of a  $p$ -cell consists of cells of dimension less than  $p$  that form the limit of the  $p$ -cell. Figure 2.2 presents the boundaries of the cells in Figure 2.1. Only the boundaries for the cells of dimension 1 and 2 are shown, since a 0-cell has no boundary.

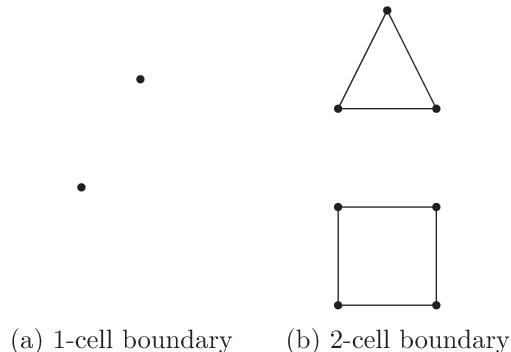


Figure 2.2: Examples of boundaries of the cells in Figure 2.1.

The face of a  $p$ -cell  $\sigma^p$  is a cell  $\tau^k$ , with  $k \leq p$ , which is part of the boundary of the  $p$ -cell. The  $p$ -cell  $\sigma^p$  is called a coface of  $\tau^k$ . As such, we can say that the face of a cell bounds it. The bounding relations of face and coface will be stated as  $\tau^p \preceq \sigma^k$  and  $\sigma^p \succeq \tau^k$ .

A cell complex  $K$  is a finite collection of cells that satisfies the following requirements:

- all the faces of a cell in the complex also belong to the complex, and
- the intersection of any two cell is either empty or a face of both cells.

A  $p$ -complex is a cell complex such that all its cells have dimension less than or equal to  $p$ . Figure 2.3 shows examples of sets of cells that satisfy and that do not satisfy the conditions of cell complexes. A subcomplex of a cell complex  $K$  is a subset of cells  $L \subseteq K$  such that  $L$  is also a cell complex.

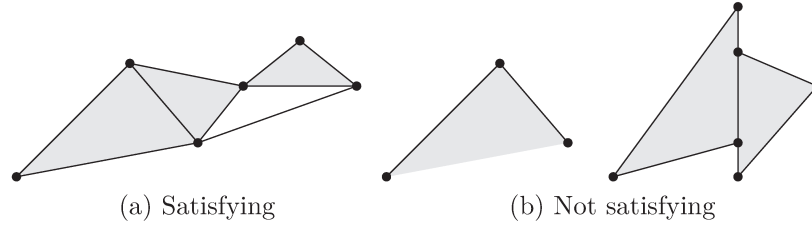


Figure 2.3: Sets of cell that satisfy (a) and that do not satisfy (b) the cell complex conditions. Figure adapted from [100]. Figure (b) has a cell without a face and two cells for which the intersection is not a shared face.

## 2.2 The Quad-Edge Data Structure

Many topological data structures have been proposed in the literature. Weiler [96] provides a detailed discussion on the subject of topological representations and operations. Similar to the edge-based structures reviewed and proposed by [96], the quad-edge, by Guibas and Stolfi [37], is a data structure that can be easily used to model 2-dimensional complexes. It provides simple manipulation of the data structure by a few operators and also is capable of representing the primal and dual representations of a planar subdivision.

We briefly introduce important concepts and operations of the quad-edge that will be used in the algorithms presented in this thesis. Some terminology of the original paper is adapted to reflect the names used in topology, namely, vertices, edges and faces are called 0-cells, 1-cells and 2-cells, respectively. For a complete understanding on quad-edges, one should refer to the original paper [37].

The basic element of the structure is a directed 1-cell  $e$  from its origin to its destination which are 0-cells returned by the functions  $\text{Org}(e)$  and  $\text{Dest}(e)$ , respectively. Other basic functions are  $\text{Left}(e)$  that returns the left 2-cell of  $e$  and  $\text{Right}(e)$  that returns the right 2-cell of  $e$ .

The quad-edge data structure is composed of four directed 1-cells, the 1-cell  $e$  itself and three more, as shown in Figure 2.4. The function  $\text{Sym}(e)$  returns the symmetric 1-cell which is directed from  $\text{Dest}(e)$  to  $\text{Org}(e)$ . This 1-cell and the 1-cell  $e$  are the primal 1-cells. The other two 1-cells are the dual 1-cells returned by functions  $\text{InvRot}(e)$  and  $\text{Rot}(e)$ . The first function returns the 1-cell directed from left to right and the second one

returns the 1-cell directed from right to left. All the functions consider a counterclockwise orientation.

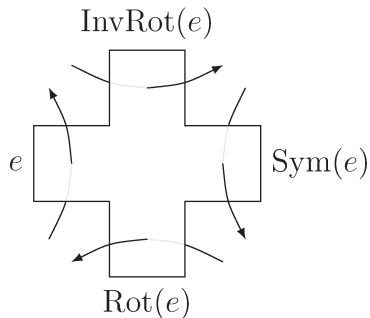


Figure 2.4: The quad-edge data structure is composed of four directed 1-cells:  $e$ ,  $\text{InvRot}(e)$ ,  $\text{Sym}(e)$  and  $\text{Rot}(e)$ . Figure adapted from [37].

By considering a sufficiently small disk  $D$  around a 0-cell, it is possible to establish a cyclic ordering of the 1-cells leaving a 0-cell  $v$ . The set of 1-cells ordered in such a manner is called the ring of 1-cells out of  $v$ . From this concept, another important 1-cell function arises. The function  $\text{Onext}(e)$  returns the next 1-cell with the same origin of  $e$ , which is the counterclockwise 1-cell immediately following  $e$  in the ring of 1-cells. The function  $\text{Edge}(v)$  returns a 1-cell in the ring of  $v$ . A set of functions is derived from the previously defined ones. The main functions and their results when applied to a 1-cell  $e$  are summarized in Figure 2.5.

Besides these functions, defined in the edge algebra of the quad-edge, a single topological operation, called splice, is used to connect, split and rearrange the quad-edge structure.

## 2.3 Images Treated as Cell Complexes

An image is ordinarily a function  $f: D \rightarrow \mathbb{R}$  defined on subset of the discrete lattice,  $D = \{(x, y) \in \mathbb{Z}^2 \mid 1 \leq x \leq M, 1 \leq y \leq N\}$ , such that a point  $p$  of  $D$  along with its value  $f(p)$  is called a pixel. An image can be modeled by a regular 2-dimensional cell complex  $K$ . Kovalevsky [47] states that cell complexes have an important role in image processing since they are the only way of consistently defining the topology of finite sets.

Kovalevsky associates pixels to 2-cells, since both are area related elements. Figure 2.6a shows an example of image represented as a cell complex with pixels associated to 2-cells. Alternatively, pixels could be related to 0-cells with 2-cells being polygons composed by pixels (Figure 2.6b). Both models are dual to each other and the use of one or another depends on preference or application needs.

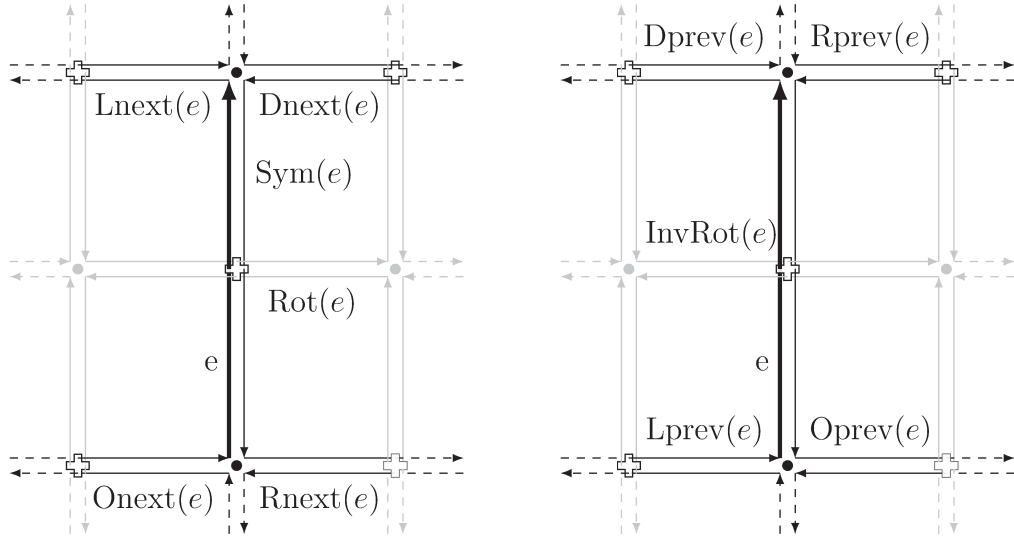


Figure 2.5: The output of the main functions applied to a 1-cell  $e$ . Figure adapted from [37].

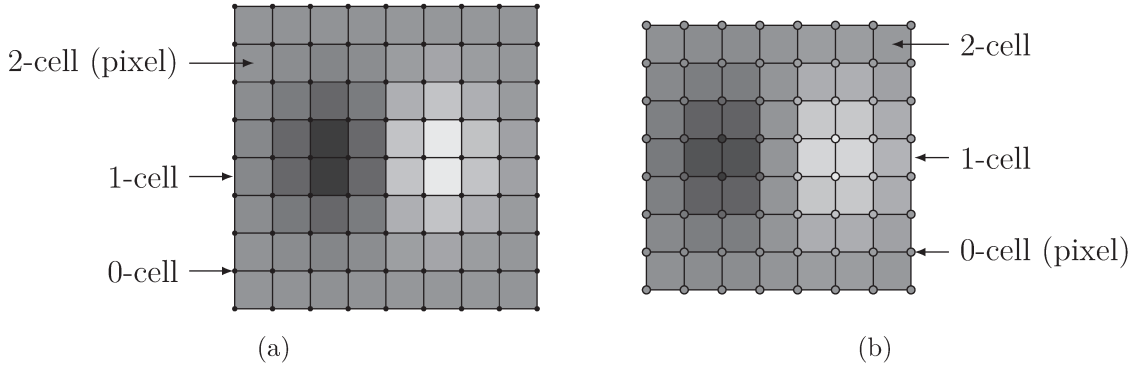


Figure 2.6: Two different ways of modeling an image as a cell complex. The two complexes are duals such that in one (a) pixels are 2-cells and in another (b) the pixels are 0-cells.

We adopt the latter model in most of our future discussions. Therefore, unless stated differently, an image is a cell complex  $K$  with 0-cells related to the pixels over  $D$ . The 2-cells are squares or triangles defined by a pixel and some of its closest pixels in  $D$ . The 1-cells are faces of the 2-cells.

### 2.3.1 Implementation

We present here an algorithm to converting an image  $f: D \rightarrow \mathbb{R}$ , to a cell complex modeled by using the quad-edge data structure. The method creates both quadrangular

2-cell complexes discussed. The two complexes can be easily obtained since we use a quad-edge data structure that also allows changing from one to another in constant time. The conversion is performed by Algorithm 2.1, which computes the primal complex as in Figure 2.6a and its dual as in Figure 2.6b. It is worth recalling that we will mainly use the latter complex, that is, the dual complex constructed by the algorithm.

---

**Algorithm 2.1:** ImageToCellImage

---

**Input:** Image  $f: D \rightarrow \mathbb{R}$

**Output:**  $K$  image cell complex

```

1 create primal 0-cells set  $VP$ 
2 create dual 0-cells set  $VD$ 
3 forall the pixel  $I(x, y)$  do
4     retrieve primal 0-cells  $v_{nw}, v_{sw}, v_{se}, v_{ne} \in VP$ 
5     retrieve dual 0-cell  $v_c \in VD$ 
6     SetFirst1Cell()
7     SetSecond1Cell()
8     SetThird1Cell()
9     SetFourth1Cell()
10    create primal 2-cell  $f$  and Set2Cell( $f, a$ )
11    create dual 2-cell  $f$  and Set2Cell( $g, \text{Rot}(a)$ ) if it exists
12 end
```

---

Line 1 of Algorithm 2.1 creates the sets  $VP$  and  $VD$  of primal and dual 0-cells. The four primal 0-cells of a pixel  $f(x, y)$  are created at coordinates  $(x - 0.5, y - 0.5)$ ,  $(x - 0.5, y + 0.5)$ ,  $(x + 0.5, y + 0.5)$  and  $(x + 0.5, y - 0.5)$ , corresponding to the corners of a unit square. The dual 0-cells are set to coordinates  $(x, y)$ .

The main loop (Lines 3-14) goes through all the pixels of the input image building 1- and 2-cells of the cell complex. Figure 2.7a describes the names of the variables of the algorithm for the primal cells created at each iteration. Figure 2.7b shows the variables and cells of the dual counterparts. The dual 2-cell  $g$  is created whenever a pixel is not at the boundary of the image, that is, for positions  $(x, y)$  such that  $1 < x \leq M$  and  $1 < y \leq N$ . Some of the 1-cells at boundaries must also have an invalid or dummy 0-cell.

The first steps of the loop retrieves the primal 0-cells  $v_{nw}, v_{sw}, v_{se}, v_{ne}$  and the dual 0-cell  $v_d$  related to a pixel  $f(x, y)$ . Given the 0-cells, the algorithm builds the 1-cells  $a, b, c, d$  following a counterclockwise orientation, consequently, the dual 1-cells ( $\text{Rot}(a)$ ,  $\text{Rot}(b)$ ,  $\text{Rot}(c)$ ,  $\text{Rot}(d)$ ) have  $v_d$  as destination 0-cell. Setting all 1-cells is a similar task but with some particularities that will be further described.

Consider edge  $a$ , this edge can enter in one of three cases as the algorithm iterates. In one condition, the vertex  $v_{nw}$  does not have a defined outgoing edge, that is,  $\text{Edge}(v_{nw}) =$



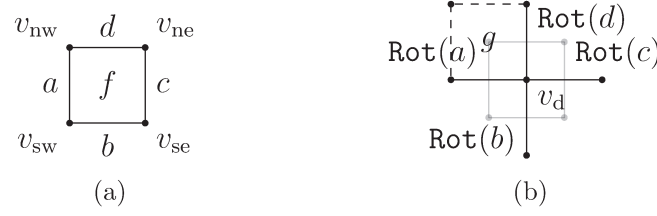


Figure 2.7: Cells created and/or adjusted at each iteration of Algorithm 2.1.

NIL. Lines 2 and 4 of Algorithm 2.2 then create the new edge  $a$  and make it the outgoing edge of  $v_{nw}$ . Lines 5 and 6 set the dual 1-cell destination and makes its symmetric the outgoing 1-cell of  $v_d$ . The internal conditional (Line 3) calls the splice operator to rearrange the orbit of  $v_{nw}$ , this is the second case  $a$  may enter. In the third case (Lines 8 and 9), an already existing edge  $a$  in the orbit of  $v_{nw}$  is found to subsequently be used in the computation of the other 1-cells.

---

**Algorithm 2.2:** SetFirst1Cell

---

```

1 if Edge( $v_{nw}$ ) = NIL or Dest(Edge( $v_{nw}$ )) =  $v_{ne}$  then
2   create edge  $a$  with endpoints  $v_{nw}$  and  $v_{sw}$ 
3   if Dest(Edge( $v_{nw}$ )) =  $v_{ne}$  then Splice(Onext(Edge( $v_{nw}$ )),  $a$ )
4   Edge( $v_{nw}$ )  $\leftarrow a$ 
5   Dest(Rot( $a$ ))  $\leftarrow v_d$ 
6   Edge( $v_d$ )  $\leftarrow$  InvRot( $a$ )
7 else
8    $a \leftarrow$  Edge( $v_{nw}$ )
9   repeat  $a \leftarrow$  Onext( $a$ ) until Dest( $a$ )  $\neq v_{sw}$ 
10 end

```

---

The 1-cell  $b$  is created according to Algorithm 2.3. The subtlety of this algorithm is the correct setting of the orbit of  $v_{sw}$ , carried out by Lines 2 and 3. The 1-cell  $c$  is the simplest case as shown in Algorithm 2.4. The fourth 1-cell has its particularities since it may be necessary to treat both the orbit of  $v_{ne}$  as well as the orbit of  $v_{nw}$ , as shown in Algorithm 2.5.

At each iteration, primal and dual 2-cells are also created. Figure 2.8a shows a primal 2-cell. The primal 2-cell is related to pixels and therefore created at each iteration. The dual 2-cell is not always created as already mentioned. If that is not the case, the dual 2-cell closed in pixel  $f(x, y)$ , corresponding to the dual 0-cell  $v_d$  (Figure 2.7b), is created as shown in Figure 2.8b.

Setting the 2-cell to all its 1-cells can be made as shown in Algorithm 2.6. The loop is executed four times since the 2-cells are squares.

**Algorithm 2.3:** SetSecond1Cell

---

```

1 create edge  $b$  with endpoints  $v_{sw}$  and  $v_{se}$ 
2 if  $\text{Edge}(v_{sw}) = \text{null}$  then  $\text{Splice}(\text{Sym}(a), b)$ 
3 else  $\text{Splice}(\text{Onext}(\text{Sym}(a), b)$ 
4  $\text{Edge}(v_{sw}) \leftarrow b$ 
5  $\text{Dest}(\text{Rot}(b)) \leftarrow v_d$ 

```

---

**Algorithm 2.4:** SetThird1Cell

---

```

1 create edge  $c$  with endpoints  $v_{se}$  and  $v_{ne}$ 
2  $\text{Splice}(\text{Sym}(b), c)$ 
3  $\text{Edge}(v_{se}) \leftarrow c$ 
4  $\text{Dest}(\text{Rot}(c)) \leftarrow v_d$ 

```

---

**Algorithm 2.5:** SetFourth1Cell

---

```

1 if  $\text{Edge}(v_{ne}) = \text{null}$  then
2   | create edge  $d$  with endpoints  $v_{ne}$  and  $v_{nw}$ 
3   |  $\text{Splice}(\text{Sym}(c), d)$ 
4   |  $\text{Splice}(\text{Sym}(d), a)$ 
5   |  $\text{Edge}(v_{ne}) \leftarrow d$ 
6 else
7   |  $d \leftarrow \text{Onext}(\text{Edge}(v_{ne}))$ 
8   |  $\text{Splice}(\text{Sym}(c), d)$ 
9 end
10  $\text{Dest}(\text{Rot}(d)) \leftarrow v_d$ 

```

---

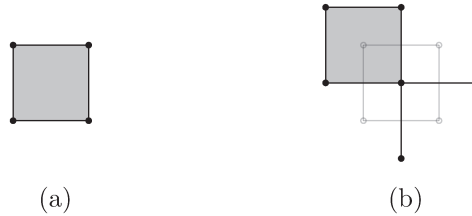


Figure 2.8: 2-cells created at each step of the algorithm.

Algorithm 2.1 returns the resulting complex  $K$ . It is worth mentioning that it suffices to return one of the primal edges created. All cells can be retrieved from this single one by traversing a complex using quad-edge functions. The Locate procedure used by Guibas and Stolfi [37] to construct a Delaunay triangulation is an example of how to traverse the data-structure.

**Algorithm 2.6:** Set2Cell

---

**Input:** 2-cell  $\tau$  and one of its 1-cells  $\alpha$   
**Input:**  $\alpha$  edge (2-cell)

```

1  $\beta \leftarrow \alpha$ 
2 repeat
3    $\text{Left}(\beta) \leftarrow \tau$ 
4    $\beta \leftarrow \text{Onext}(\beta)$ 
5 until  $\beta = \alpha$ 

```

---

## 2.4 Topological Operations

We introduce here some basic topological operations on cell complexes. The star of a cell is particularly interesting since it defines a neighborhood notion. Actually, it is the same as the smallest neighborhood of a cell  $\sigma$  in a complex  $K$ .

**Definition 2.1.** *The star of a subcomplex  $L \subseteq K$  contains all cofaces of  $L$ ,  $\text{St}(L) = \{\alpha \in K \mid \alpha \succeq \tau \in L\}$ . The closure of  $L$  is the smallest subset of  $K$  containing  $L$ ,  $\text{Cl}(L) = \{\tau \in K \mid \tau \preceq \alpha \in L\}$ . The link of  $L$  is the boundary of its star,  $\text{Lk}(L) = \text{Cl}(\text{St}(L)) - \text{St}(\text{Cl}(L) - \{\emptyset\})$ .*

We are particularly interested in the application of the star and the link to 0-cells (pixels) of image cell complexes such as the one in Figure 2.6b. Figure 2.9 shows examples of each operation. The star and the link are applied to 0-cells and the closure is applied to the star of a 0-cell since the closure of the pixel is the pixel itself.

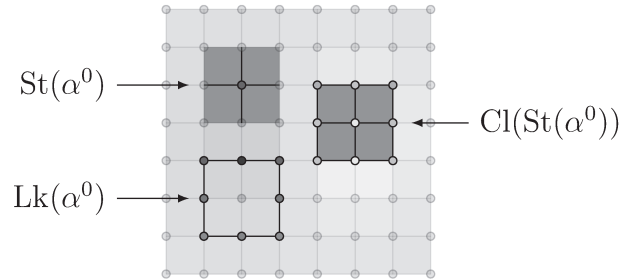


Figure 2.9: Example of star, closure and link. The operations are applied to the 0-cells in the center of the highlighted areas. In the case of the closure, first a star is applied to the 0-cell and then the closure is applied to the result of the star.

These functions can be easily computed by using the quad-edge data structure. The star requires to traverse the ring or 1-cells of a 0-cell storing the 1-cells and the 2-cells returned by the function **Left** at each 1-cell. For the closure, it also suffices to store all the faces of cells in the star. The link can be obtained by listing the 1-cells which are faces

of all 2-cells in the star and removing those that are shared by two 2-cells in the star. The remaining edges as well as their 0-cells belong to the link. Since quad-edge functions take constant time and a constant number of them is computed for each cell in the topological operations (star, closure and link), then the previous computations can be performed in linear time on the cardinality of the sets  $\text{St}(L)$ ,  $\text{Cl}(L)$  and  $\text{Lk}(L)$ .

# Chapter 3

## Discrete Morse Theory

The discrete Morse theory, formulated by Forman [30, 31, 32], is an adaptation to discrete structures of the Morse theory [64]. We will present here the basics on the theory and the concepts will be used in the next chapter for the construction of the discrete Morse complex. The topological space for the discrete Morse theory is a cell complex [52].

**Definition 3.1.** *A discrete Morse function on a cell complex  $K$  is a function  $f: K \rightarrow \mathbb{R}$  satisfying, for all  $\sigma^p \in K$ :*

$$\sharp\{\tau^{p+1} \succ \sigma^p \mid f(\tau^{p+1}) \leq f(\sigma^p)\} \leq 1 \quad (3.1)$$

$$\sharp\{\lambda^{p-1} \prec \sigma^p \mid f(\lambda^{p-1}) \geq f(\sigma^p)\} \leq 1 \quad (3.2)$$

such that  $\sharp$  is the set cardinality.

Definition 3.1 states that, for every  $\sigma^p \in K$ ,  $f$  takes a value less than or equal to  $f(\sigma^p)$  in at most one coface of  $\sigma^p$  and takes a value greater than or equal to  $f(\sigma^p)$  in at most one face of  $\sigma^p$ . Figure 3.1a shows an example of a discrete Morse function. The function of Figure 3.1b is not a discrete Morse function since the 2-cell with value 4 and the 1-cell with value 0 are not valid according to the definition.

**Definition 3.2.** *Given a discrete Morse function  $f$ , a cell  $\sigma^p \in K$  is critical of index  $p$  if*

$$\sharp\{\tau^{p+1} \succ \sigma^p \mid f(\tau^{p+1}) \leq f(\sigma^p)\} = 0 \quad (3.3)$$

$$\sharp\{\lambda^{p-1} \prec \sigma^p \mid f(\lambda^{p-1}) \geq f(\sigma^p)\} = 0. \quad (3.4)$$

Definition 3.2 states that a cell  $\sigma^p \in K$  is critical if all cofaces take strictly greater values in  $f$  and all faces are strictly lower in  $f$  [76]. In a two dimensional cell complex, minima are 0-cells, saddles are 1-cells and maxima are 2-cells. Figure 3.2 shows examples of critical cell of a discrete Morse function. The 0-cell of value 0 is a minimum and the 1-cell of value 5 is a saddle.

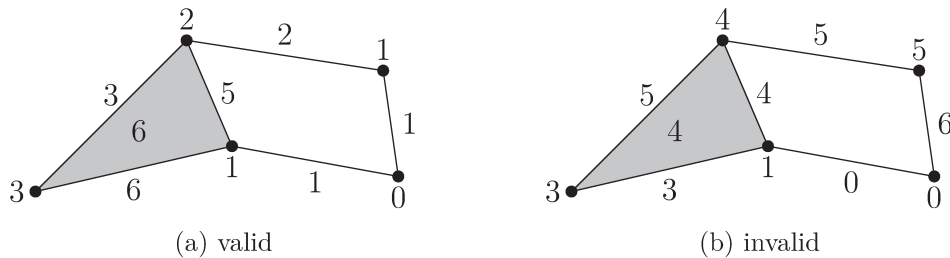


Figure 3.1: Example of a function that is (a) a valid Morse function and (b) a function that is not a Morse function. Figure adapted from [51].

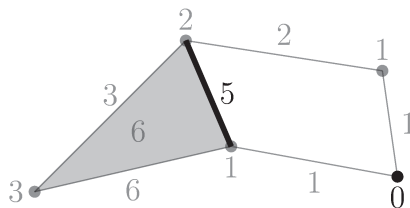


Figure 3.2: Example of critical cells (saddle and minimum) of a discrete Morse function. Figure adapted from [51].

**Definition 3.3.** A discrete vector field,  $V$ , is a collection of pairs  $\{\alpha^p \prec \beta^{p+1}\}$  of cells in  $K$  such that each cell is in at most one pair in  $V$ . A discrete Morse function defines a discrete vector field by pairing  $\alpha^p \prec \beta^{p+1}$  whenever  $f(\beta^{p+1}) \leq f(\alpha^p)$ .

A pair  $\alpha^p \prec \beta^{p+1}$  can be thought of as a discrete tangent vector leaving  $\alpha^p$  and oriented by  $\beta^{p+1}$ . Pictorially, the vector is represented by an arrow from  $\alpha^p$  to  $\beta^{p+1}$ . Figure 3.3 shows an example of discrete vector field of the Morse function of Figure 3.1a. Every cell belongs to a pair in the vector field, except for critical cells (Figure 3.2).

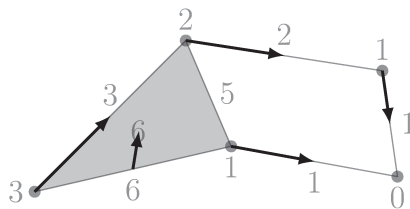


Figure 3.3: Example of a discrete Morse vector field. All cells are paired except for critical cells.

An important concept related to vector fields is that of flow, which in the discrete Morse theory is called a  $V$ -path.

**Definition 3.4.** A *V-path* is a sequence of cells:

$$\alpha_1^p, \alpha_1^{p+1}, \alpha_2^p, \alpha_2^{p+1}, \alpha_3^p, \dots, \alpha_{r-1}^{p+1}, \alpha_r^p \quad (3.5)$$

where  $\alpha_i^p, \alpha_i^{p+1} \in V$ ,  $\alpha_i^{p+1} \succ \alpha_{i+1}^p$ , and  $\alpha_i^p \neq \alpha_{i+1}^p$ , for all  $i = 1, \dots, r-1$ .

A *V-path* is a non-trivial closed *V-path* if  $\alpha_r^p = \alpha_1^p$  for  $r \geq 2$ . Forman [30] presents  $V$  without non-trivial closed paths as the discrete analogue of the continuous gradient vector field. Figure 3.4 shows a *V-path* in a discrete vector field.

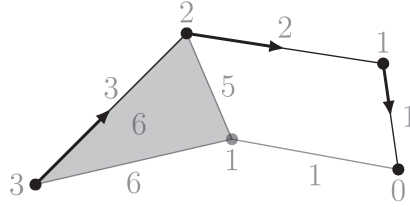


Figure 3.4: Example of a *V-path*.

The counterpart of the integral curves limited by critical points are *V-paths* such that  $\alpha_1^p$  and the coface  $\alpha^{p+1} \succ \alpha_r^p$ ,  $\alpha_{r-1}^{p+1} \neq \alpha^{p+1}$ , are critical cells.

### 3.1 Computation of the Discrete Morse Vector Field

We present here some additional topological concepts and notions on simple homotopy theory [17, 97]. The theory presented by Forman makes use of the simple homotopy and the related concepts are used by Robins [76] to produce an algorithm to computing discrete vector fields. The following discussion is based on these works.

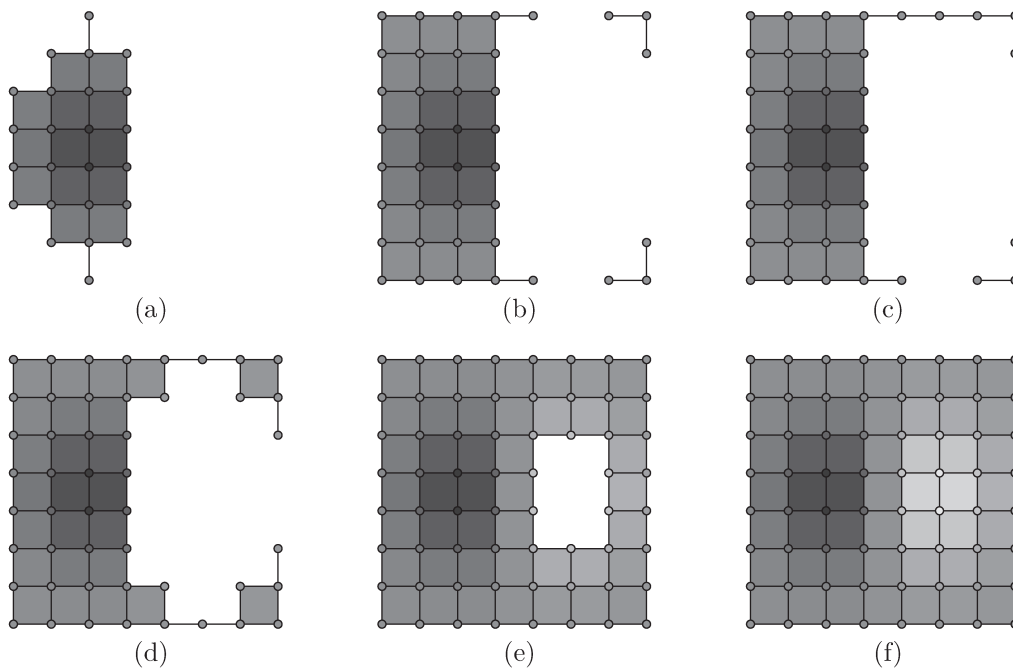
**Definition 3.5.** Given a discrete Morse function  $f$  on a complex  $K$  and any  $t \in \mathbb{R}$ , a level subcomplex  $K(t)$  is defined as

$$K(t) = \cup_{f(\sigma) \leq t} \cup_{\alpha \leq \tau} \sigma. \quad (3.6)$$

The subcomplex  $K(t)$  is composed by all cells  $\tau$  with  $f(\tau) \leq t$  as well as all of their faces. A sequence of subcomplexes  $K(t_i)$  for increasing values  $t_i$  defines a filtration of the complex  $K$ .

**Definition 3.6.** A filtration of a complex  $K$  is a nested sequence of subcomplexes,  $\emptyset = K(0) \subseteq K(1) \subseteq K(2) \subseteq \dots \subseteq K(m) = K$ .

Figure 3.5 shows a subset of level sets from a filtration of an example complex.

Figure 3.5: Six level subcomplexes of a complex  $K$ .

**Definition 3.7.** Let  $\alpha^{p-1} \prec \alpha^p$  be cells of a complex  $K$ . If  $\alpha^{p-1}$  has no other cofaces, then  $K \setminus \{\alpha^{p-1}, \alpha^p\}$  is called an elementary collapse. The inverse of a collapse is an expansion.

**Lemma 3.1.** If there are no critical cells  $\sigma$  with  $f(\sigma) \in [a, b[$ , then  $K(a)$  collapses to  $K(b)$ .

Lemma 3.1 (Theorem 3.3 of Forman's [30]) relates the simple homotopy theory with the level subcomplexes. It says that a complex  $K(a)$  collapses to a subcomplex  $K(b)$  by a finite sequence of elementary collapses. Forman shows that a complex  $K$  with a discrete Morse function is homotopy equivalent<sup>1</sup> to a complex with exactly one cell of dimension  $p$  for each critical cells of  $K$  of dimension  $p$ .

These results were translated by Robins et al. [76] into an algorithm that computes the vector field  $V$  of a complex  $K$  by simple homotopy expansions and pairings. Algorithm 3.1 shows how the computations are done by using the quad-edge to model the vector field. The method requires distinct values for the function  $f: K \rightarrow \mathbb{R}$  for each pixel so that the values can be ordered in increasing order and produce a filtration of  $K$ . To ensure unique values, the function can be perturbed as explained in [76]. The sorting to produce a filtration of a complex with  $n$  cells can be computed in  $O(n \log n)$  or, alternatively, if the range of the values is properly limited, a linear sorting algorithm can be used. The

<sup>1</sup>Spaces that can be deformed continuously into one another.



**Algorithm 3.1:** ComputeVectorField

---

**Input:** Cell complex  $K$ .  
**Output:** Cell complex  $K$  with paired cells and set of critical cells  $C$ .

```

1 forall the  $\sigma^0 \in K$  ordered by  $f(\sigma^0)$  do
2   if  $\text{LowerStar}(\sigma^0) = \{\sigma^0\}$  then
3     Insert  $\sigma^0$  into  $C$ 
4   else
5      $\beta \leftarrow \alpha^1 \in \text{LowerStar}(\sigma^0)$  such that  $G(\alpha^1)$  is minimal
6      $\text{Pair}(\sigma^0) \leftarrow \beta$ 
7     add all other 1-cells of  $\text{LowerStar}(\sigma^0)$  to  $PQ_0$ 
8     add  $\gamma \in \text{LowerStar}(\sigma^0)$ ,  $\gamma \succ \beta$  and  $\text{Unpaired}(\gamma) = 1$ , to  $PQ_1$ 
9     while  $PQ_0 \neq \emptyset$  or  $PQ_1 \neq \emptyset$  do
10      while  $PQ_1 \neq \emptyset$  do
11        remove  $\gamma$  from the front of  $PQ_1$ 
12        if  $\text{Unpaired}(\gamma) = 0$  then
13          add  $\gamma$  to  $PQ_0$ 
14        else
15           $\text{Pair}(\text{UnpairedFace}(\gamma)) \leftarrow \gamma$ 
16          Remove  $\text{UnpairedFace}(\gamma)$  from  $PQ_0$ 
17          add  $\delta \in \text{LowerStar}(\alpha)$ ,  $(\delta \succ \gamma$  or  $\delta \succ \text{UnpairedFace}(\gamma))$  and
             $\text{Unpaired}(\delta) = 1$ , to  $PQ_1$ 
18        end
19      end
20      if  $PQ_0 \neq \emptyset$  then
21        remove  $\gamma$  from the front of  $PQ_0$ 
22        insert  $\gamma^0$  into  $C$ 
23        add  $\delta \in \text{LowerStar}(\alpha)$ ,  $\delta \succ \gamma$  and  $\text{Unpaired}(\delta) = 1$ , to  $PQ_1$ 
24      end
25    end
26  end
27 end

```

---

complexity of Algorithm 3.1 is  $O(N)$  [76], where  $N$  is the number of disjoint lower stars of a complex  $K$ .

Algorithm 3.1 processes the levels sets produced by the ordering of the values  $f(\sigma^0)$  for each pixel  $\sigma^0 \in K$ . Each iteration considers a pixel  $\sigma^0$  in the ordering and its neighborhood, which is given by its lower star ( $\text{LowerStar}(\sigma^0)$ ). The lower star is formed by the cells  $\tau$  in the star (Definition 2.1) for which  $f(\tau) \leq f(\sigma^0)$ . The algorithm uses two priority queues,  $PQ_0$  and  $PQ_1$ , to control the cells that are paired in the vector field  $V$ .

We introduce a function **Pair** to set and retrieve the pairs in  $V$ . Given a cell  $\beta$  of

$K$ , if the cell is in a pair of  $\beta \prec \gamma$  or  $\gamma \prec \beta$  in  $V$ , the function  $\text{Pair}(\beta)$  returns the cell  $\gamma$ ; otherwise it returns NIL, which means the cell is critical. This reflects the idea of a vector field presented in Definition 3.3. The function  $\text{Unpaired}(\beta)$  returns the number of faces of a cell  $\beta$  that are neither paired in  $V$  nor inserted into the set of critical cells  $C$ . When a single face is unpaired,  $\text{UnpairedFace}(\beta)$  returns the face. The algorithm requires an ordering of the faces in each lower star. Robins et al. implemented a lexicographic ordering by listing the values of  $f$  for the 0-cells of cell in decreasing order.

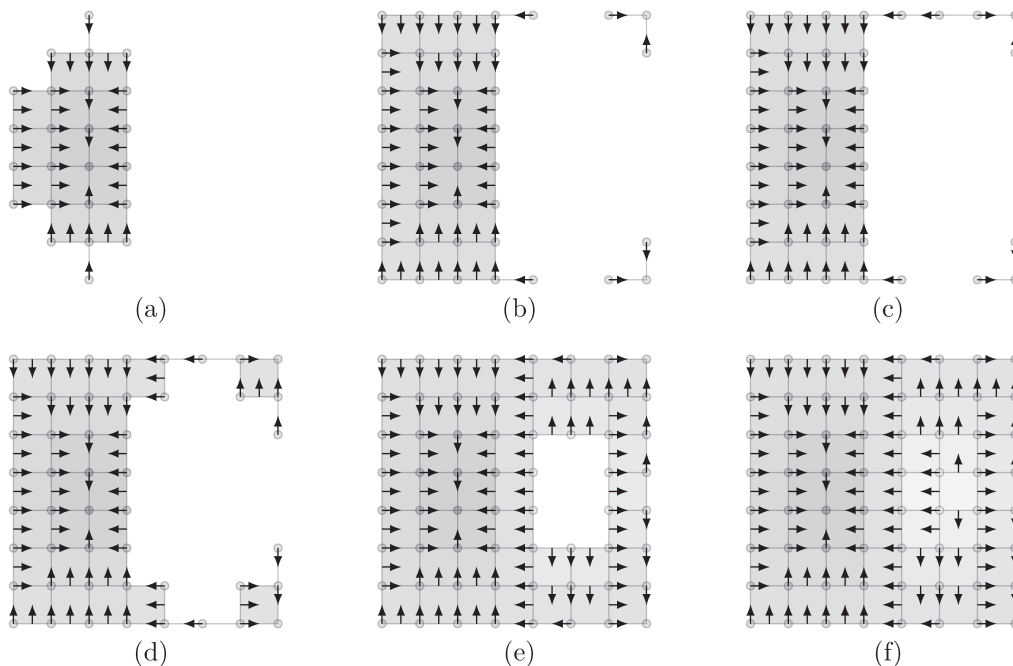


Figure 3.6: Results of the computation of the vector field of a complex at different iterations of the Algorithm 3.1. The resulting vector field contains all cells paired except for critical cells.

For a complete discussion on the algorithm, one should refer to the original paper [76]. Our interest is in the resulting vector field  $V$  of  $K$ . Figure 3.6 shows the result of the algorithm for each level set of Figure 3.5. At each level set, it is possible to notice the critical cells as unpaired cells while the other cells are paired. The level set of Figure 3.6a shows that only one 0-cell is not paired, therefore, it is a minimum of the function. Figure 3.6a depicts two new critical 0-cells. At the level sets shown in Figures 3.6c, 3.6d and 3.6e, three critical 1-cells (saddles) are unpaired. The complete vector field of  $K$  is shown in Figure 3.6f and a critical 2-cell (maximum) is identified.

## Chapter 4

# Discrete Morse Complex of Images

In this chapter we present a method to build the discrete Morse complex of images. We have already mentioned that approaches for Morse complexes have been proposed in works such as [6, 26, 99]. These are approaches adapt smooth notion of the Morse theory to discrete data and produce the so called Morse-Smale complexes. However there are difficulties because of saddle points of multiple types and spurious critical points created because of intersections of integral curves.

The discrete Morse theory [30] is another construction studied, for example, in [46, 54]. The problem of saddles does not occur in the discrete Morse complex since there is only one type. However, the  $V$ -paths (counterparts of integral curves in the smooth case) still can be an issue. Algorithms for images were presented in [76], but their approach is not suitable for visualizations or the computation of hierarchies of Morse complexes. As in the previous works, the data structures and their influences are not a primary concern and it is difficult to adapt their algorithms to topological data structures (see Section 2.2). The breadth-search based algorithm does not maintain the necessary information to deal with merge and branch cases that we will describe in this chapter.

We will show there are ways of computing a 2-dimensional discrete Morse complex and maintain the consistency of a quad-edge data structure. We base our approach on Forman's theory [30] and on the algorithm of Robins et. al [76] to compute the discrete Morse vector field (Algorithm 3.1). Our approach determines the paths between critical cells, in a discrete vector field, starting from saddle critical cells. It is shown that such an approach is optimal since it only processes cells that are in a path, opposed to the breadth-search that considers cells even if they are not in any path. It is also possible to deal with merging and branching cases properly and produce a consistent representation of the quad-edge model. We also present algorithms to deal with such cases for visualization purposes. Finally, we model a complex so that topological computations such as persistent homology [4, 9, 18, 67, 93] and simplification of topological features [60, 75, 91] can be

easily computed.

In summary, the main contributions of this chapter are: an algorithmic approach to the construction and manipulation of discrete Morse complexes; an optimal algorithm for extracting a 2-dimensional discrete Morse complex from a discrete vector field; and a model of the discrete Morse complex by means of a quad-edge data structure (that allows visualization of the complexes; computation of persistent Betti numbers and simplification of the complex as used for topological noise removal and hierarchies of complexes).

The discrete Morse complex construction described in Section 4.1 is applied to experimental images in Section 4.2 to show a real application of the method and its usefulness for important topological computation, namely the persistent homology and simplification of Morse complexes to produce hierarchical representations.

## 4.1 Algorithms

We present here a detailed discussion on each step of our method to acquire the  $V$ -paths that result in the discrete Morse complex. The initial sections provide discussions on how to extract the paths. After the extraction of all paths in a discrete Morse complex, we show how to deal with particular cases of merging and branching paths, producing a complex that is suitable for visualization tasks. At last, we show how to compute a simplified discrete Morse complex that can also be used for efficient topological computations.

### 4.1.1 Some Considerations

In the method, we will basically work with two complexes. The first one is the complex  $K$  for the input image along with its vector field  $V$ . The second complex is the discrete Morse complex  $M$  in construction. Both of them are modeled using the quad-edge data structure.

The two complexes will be referred to in the discussion stating the correspondence between their cells. In order to avoid confusion when we are talking about cells in  $K$  or about cells in  $M$ , from now on, we will use Greek letters for cells in  $K$  and Latin letters for cells in  $M$ . Particularly, we will use  $v^p$  or  $u^p$  to denote a 0-cell in  $M$  and  $e$  to denote a 1-cell in  $M$ . A cell  $v$  will correspond to a  $p$ -cell in  $K$  and  $e$  will denote a boundary relation between cells in  $K$ .

Since a 0-cell  $v$  in  $M$  can correspond to any  $p$ -cell in  $K$  we will introduce a function  $\text{Index}(v)$  that returns the dimension of the cell in  $K$  related to  $v$ . A superscript will also be used whenever we need to state the dimension of the cell in  $K$  corresponding to  $v$ . Therefore,  $v^p$  is a 0-cell in  $M$  related to a  $p$ -cell in  $K$ . Figure 4.1 shows two complexes  $K$  and  $M$  and the corresponding cells. The 0-cells in  $M$  can be regular or critical according

to the cells they correspond in  $K$ . Also the index  $p$  of a 0-cell  $v^p$  is related to the dimension of the cell in  $K$ . The 1-cells explicitly model the boundary relation between cells. The symbols  $\ominus$ ,  $\oplus$  and  $\odot$  will henceforth denote critical 0-cells in  $M$  related to 0-, 1- and 2-cells which are also critical in  $K$ . If not critical, the 0-cell will be drawn with the symbol  $\circ$ .

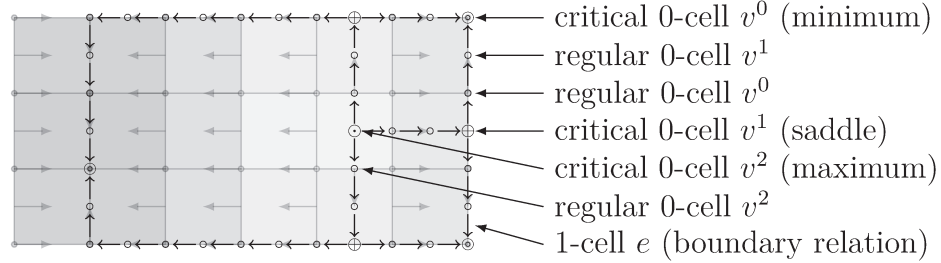


Figure 4.1: Complex  $K$  with  $M$  placed over it. The 0-cells of  $M$  correspond to any type of cells  $K$  while the 1-cells of  $M$  explicitly model the boundary relations between cells. The 0-cells of  $M$  inherit the properties of cells in  $K$ . The index  $p$  of a cell  $v^p$  in  $M$  is related to the dimension of the corresponding cell in  $K$ . Also, the 0-cells can be regular or critical.

Before presenting the algorithms, we introduce alternative definitions of  $V$ -paths to reflect the explicitly modeled paths used in the algorithms. In two dimensions, a discrete Morse complex can have two types of  $V$ -paths:  $(0, 1)$ -paths and  $(1, 2)$ -paths. We denote such paths as  $QV$ -paths of the complex  $M$ . Definitions for both cases are presented next.

**Definition 4.1.** *Given a  $(0, 1)$ -path in  $V$  (see Definition 3.4),  $\alpha_1^0, \alpha_1^1, \alpha_2^0, \dots, \alpha_{r-1}^1, \alpha_r^0$ , a  $(0, 1)$ -path in  $M$ , leaving  $v_0^1$ , is a sequence of 0-cells (denoted by  $v_i^{\{0,1\}}$ ) and 1-cells (denoted by arrows):*

$$v_0^1 \rightarrow v_1^0 \rightarrow v_1^1 \rightarrow v_2^0 \rightarrow \dots \rightarrow v_{r-1}^1 \rightarrow v_r^0$$

where  $v_i^0 \rightarrow v_i^1$  are the cells related to a pair  $\alpha_i^0, \alpha_i^1 \in V$ ;  $v_{i+1}^0$  is related to  $\alpha_{i+1}^0$  such that  $\alpha_i^1 \succ \alpha_{i+1}^0$  and  $\alpha_i^0 \neq \alpha_{i+1}^0$ , for all  $i = 1, \dots, r-1$ ; and  $v_0^1$  is related to a critical cell  $\alpha^1 \succ \alpha_1^0$ ,  $\alpha^1 \neq \alpha_1^1$ .

Notice from this definition that the  $(0, 1)$ -path leaves a 1-cell and arrives to a 0-cell, in  $K$ , making explicit the connections of endpoints and following the intuitive notion of a flow going downwards from a saddle to a minimum cell. In a similar way, a  $(1, 2)$ -path is presented in Definition 4.2, but now the path leaves a 2-cell towards a 1-cell of  $K$ , that is, follows from a maximum to a saddle.

**Definition 4.2.** *Given a  $(1, 2)$ -path in  $V$  (see Definition 3.4),  $\alpha_1^1, \alpha_1^2, \alpha_2^1, \dots, \alpha_{r-1}^2, \alpha_r^1$ , a  $(1, 2)$ -path in  $M$ , arriving to  $v_r^1$ , is a sequence of 0-cells (denoted by  $v_i^{\{1,2\}}$ ) and 1-cells*

(denoted by arrows):

$$v_0^2 \rightarrow v_1^1 \rightarrow v_1^2 \rightarrow v_2^1 \rightarrow \dots \rightarrow v_{r-1}^2 \rightarrow v_r^1$$

where  $v_i^1 \rightarrow v_i^2$  are the cells related to a pair  $\alpha_i^1, \alpha_i^2 \in V$ ;  $v_{i+1}^1$  is related to  $\alpha_{i+1}^1$  such that  $\alpha_i^2 \succ \alpha_{i+1}^1$  and  $\alpha_i^1 \neq \alpha_{i+1}^1$ , for all  $i = 1, \dots, r-1$ ; and  $v_0^2$  is related to the cell  $\alpha^2 \succ \alpha_1^1$ ,  $\alpha^2 \neq \alpha_1^2$ .

From the definitions of paths, we acquire the precedent and subsequent 0-cells of another 0-cell. We add these relations to the 0-cells in  $M$  so that the functions  $\text{Prev}(v)$  and  $\text{Next}(v)$  return the previous and next 0-cells of  $v$  in a path.

### 4.1.2 Extraction of $QV$ -Paths

The algorithm for extracting the  $QV$ -paths connecting critical cells is based on searching the paths out of 1-cells. Such an approach allows optimal computation of the paths and also maintaining the quad-edge data structure consistent. The latter is possible since the approach make it straight forward to deal with mergings of  $(0, 1)$ -paths and branchings of  $(1, 2)$ -paths to maintain a consistent ordering in the rings of 1-cells of the quad-edge. The former happens because the algorithms consider only paths that may connect critical cells.

#### Initialize $V$ -paths of a 1-cell

The first step of the method initializes  $QV$ -paths out of a saddle in  $K$ , identifying all possible  $V$ -paths and including them in the resulting complex  $M$ . At most four  $V$ -paths can be expected to go out of a 1-cell, as soon will be stated.

**Lemma 4.1.** *A 1-cell has exactly two 0-cell faces.*

**Lemma 4.2.** *A 1-cell has at most two 2-cell cofaces.*

**Lemma 4.3.** *A 1-cell has at most four  $V$ -paths leaving or arriving.*

Lemmas 4.1 and 4.2 are intuitive from the definitions of a cell and a cell complex. For mathematical approaches on the results, one should refer to combinatorial texts on cell complexes such as [2]. An 1-cell may not have two 2-cell cofaces if it is in the boundary of the cell complex. Alternatively, one can think as it has two faces, one in the complex and one is the complement of the complex. Lemma 4.3 follows from the two previous lemmas.

From these results it is now possible to initialize the  $QV$ -paths related to a 1-cell. Algorithm 4.1 receives a critical 1-cell  $\alpha^1$  of  $K$  and creates all  $QV$ -paths arriving and leaving a 0-cell  $v_\alpha^1 \in M$  related to  $\alpha^1$ . Figure 4.2 shows an example of how the algorithm

**Algorithm 4.1:** InitVPaths

---

**Input:** Critical 1-cell  $\alpha^1$   
**Output:**  $\{(\alpha^1, v_\alpha), (\beta^0, v_\beta), (\gamma^0, v_\gamma), (\sigma^2, v_\sigma), (\tau^2, v_\tau)\}$

- 1 create 0-cell  $v_\alpha \in M$  for 1-cell  $\alpha^1$
- 2  $\beta^0 \leftarrow \alpha^0 \prec \alpha^1$
- 3  $\gamma^0 \leftarrow \alpha^0 \prec \alpha^1$  such that  $\alpha^0 \neq \beta^0$
- 4  $\sigma^2 \leftarrow \alpha^2 \succ \alpha^1$
- 5  $\tau^2 \leftarrow \alpha^2 \succ \alpha^1$  such that  $\alpha^2 \neq \sigma^2$
- 6 create 0-cell  $v_\beta \in M$  for 0-cell  $\beta^0$
- 7 create 0-cell  $v_\gamma \in M$  for 0-cell  $\gamma^0$
- 8 create 0-cell  $v_\sigma \in M$  for 2-cell  $\sigma^2$
- 9 create 0-cell  $v_\tau \in M$  for 2-cell  $\tau^2$
- 10 create 1-cell  $e_1$  from  $v_\alpha$  to  $v_\beta$
- 11 create 1-cell  $e_2$  from  $v_\alpha$  to  $v_\gamma$
- 12 create 1-cell  $e_3$  from  $v_\alpha$  to  $v_\sigma$
- 13 create 1-cell  $e_4$  from  $v_\alpha$  to  $v_\tau$
- 14 Splice( $e_1, e_2$ )
- 15 Splice( $e_2, e_3$ )
- 16 Splice( $e_3, e_4$ )
- 17 Splice( $e_4, e_1$ )

---

works. The input 1-cell  $\alpha$ , has two faces,  $\beta^0$  and  $\gamma^0$ ; and also two cofaces,  $\sigma^2$  and  $\tau^2$  (Figure 4.2a).

The algorithm creates the 0-cells in  $M$  related to each of in  $K$  as well as the four 1-cells that model the boundary relations, this can be observed in Figures 4.2b, 4.2c, 4.2d and 4.2e. At the end, the ring of 1-cells (see Section 2.2) out of  $v_\alpha$  is adjusted (Figure 4.2f). The algorithm returns pairs with cells created in  $M$  and the cells related to them in the complex  $K$ .

As stated before, the symbols  $\odot$ ,  $\oplus$  and  $\ominus$  denote critical 0-cells in  $M$  and the symbol  $\circ$  depicts not critical cells. In Figure 4.2 only  $v_\alpha$  is depicted differently since this is the only critical cell, a saddle.

Recall Definitions 4.1 and 4.2 of  $QV$ -paths. It is useful to perceive how the resulting paths agree with the definitions given. Let  $v_\alpha$  be  $v_0^1$  in Definition 4.1, two  $QV$ -paths of the form  $v_0^1 \rightarrow v_1^0$  are now in the complex  $M$ , with  $v_1^0$  being  $v_\beta$  in one path and  $v_\gamma$  in another. Both paths are related to a trivial  $V$ -path of the form  $\alpha_1^0$  in  $V$  (Definition 3.4).

A similar reasoning can be done for the  $(1, 2)$ -paths, leading to two paths of the form  $v_0^2 \rightarrow v_1^1$ . However, as the extraction of the paths is further discussed and algorithms are presented, it will become clear it is more convenient to think of  $(1, 2)$ -paths as being computed in a backwards fashion. In such a manner, one could think of  $v_\alpha$  to be  $v_r^1$  and

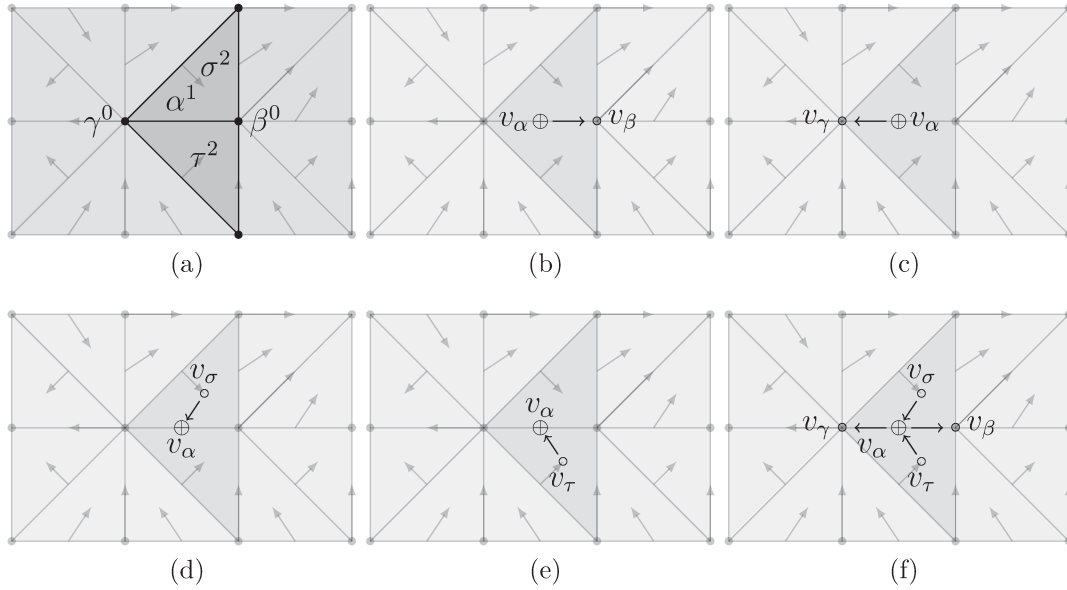


Figure 4.2: Given a critical 1-cell in  $V$  (a), Algorithm 4.1 creates the  $QV$ -paths out of it (b)-(e) and arranges them in the ring of edges out of the vertex related to the 1-cell (f).

$v_\sigma$  and  $v_\tau$  to be  $v_{r-1}^2$  in Definition 4.2. Therefore, two paths of the form  $v_{r-1}^2 \rightarrow v_r^1$  are obtained. Both paths are related to a trivial  $V$ -path of the form  $\alpha_r^2$  in  $V$  (Definition 3.4).

Algorithm 4.1 takes constant time. All operations to create cells take constant time. The retrieval of faces and cofaces also take  $O(1)$  since it suffices using  $O(1)$  quad-edge operations: **Orig**, **Dest**, **Left**, **Right** (see Section 2.2). The **Splice** operation also has  $O(1)$  time [37]. We consider that with the creation of 0-cells their attributes are properly set (index  $p$  of  $v^p$ , if it is critical and the next and previous 0-cells).

### Expanding $QV$ -paths

The next step after initializing the  $QV$ -paths is to expand the paths until a termination condition is found. Now it will be shown that there is only one way of growing a particular  $QV$ -path. Algorithms will be presented to show how to compute such expansions of  $(0, 1)$ - and  $(1, 2)$ -paths.

From the previous discussion it can be noticed that  $(0, 1)$ -paths are initialized so to grow in a forward manner while  $(1, 2)$ -paths are to be grow backwards. Therefore, after the initialization step, the following types of paths were present in the complex  $M$ :  $(0, 1)$ -paths of the form  $v_0^1 \rightarrow v_1^0$  and  $(1, 2)$ -paths of the form  $v_{r-1}^2 \rightarrow v_r^1$ .

Expanding forward a  $(0, 1)$ -path means to grow the initial  $QV$ -path to  $v_0^1 \rightarrow v_1^0 \rightarrow v_2^1$  (related to a path  $\alpha_1^0, \alpha_1^1, \alpha_2^0$  in  $V$ ), maintaining a path according to Definition 4.1. Likewise, expanding backwards a  $(1, 2)$ -path means to grow the initial  $QV$ -path



to  $v_{r-2}^2 \rightarrow v_{r-1}^1 \rightarrow v_{r-1}^2 \rightarrow v_r^1$  (related to a path  $\alpha_{r-1}^2, \alpha_{r-1}^1, \alpha_r^2$  in  $V$ ), also maintaining a path according to Definition 4.2. Therefore, the expansion of the  $QV$ -paths is basically obtained by considering a pair  $\alpha^0 \prec \alpha^1$  or a pair  $\alpha^1 \prec \alpha^2$ , in the input vector field  $V$ , as well as the another face or coface of  $\alpha^1$  that is not paired with it.

**Definition 4.3.** *Given a  $(0,1)$ -path  $v_0^1 \rightarrow v_1^0 \rightarrow \dots \rightarrow v_{k-2}^1 \rightarrow v_{k-1}^0$  and its related  $V$ -path  $\alpha_1^0, \alpha_1^1, \dots, \alpha_{k-2}^1, \alpha_{k-1}^0$ , the expanding triple of the path is the set of cells  $\{\alpha_{k-1}^0, \alpha_{k-1}^1, \alpha_k^0\}$  such that  $\alpha_{k-1}^0, \alpha_{k-1}^1 \in V$  and  $\alpha_{k-1}^1 \succ \alpha_k^0$ ,  $\alpha_{k-1}^0 \neq \alpha_k^0$ , that allows growing the  $QV$ -path to  $v_0^1 \rightarrow v_1^0 \rightarrow \dots \rightarrow v_{k-2}^1 \rightarrow v_{k-1}^0 \rightarrow v_{k-1}^1 \rightarrow v_k^0$  with related  $V$ -path  $\alpha_1^0, \alpha_1^1, \dots, \alpha_{k-2}^1, \alpha_{k-1}^0, \alpha_{k-1}^1, \alpha_k^0$ .*

Take, for example, the  $(0,1)$ -path  $v_\alpha \rightarrow v_\beta$  in Figure 4.2b. The three cells in the expanding triple are the cells shown in Figure 4.3 following the  $V$ -path out of  $\beta_o^0$ . These cells are all needed to expand the  $QV$ -path out of  $v_\beta$ . The endpoints of a expanding triple will be indexed with an “o” or a “d” to make it explicit where the flow enters the path and where it goes to. Therefore, in the example of Figure 4.3, the expanding triple comprises the cells  $\beta_o^0$ ,  $\beta^1$  and  $\beta_d^0$ .

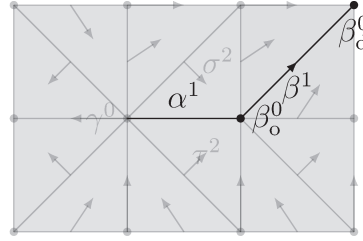


Figure 4.3: Expanding triple of a  $(0,1)$ -path.

**Definition 4.4.** *Given a  $(1,2)$ -path  $v_{k+1}^2 \rightarrow v_{k+2}^1 \rightarrow \dots \rightarrow v_{r-1}^2 \rightarrow v_r^1$ , its related  $V$ -path  $\alpha_{k+2}^1, \alpha_{k+2}^2, \dots, \alpha_{r-1}^2, \alpha_r^1$  and  $\alpha_{k+1}^2$  related to  $v_{k+1}^2$ , the expanding triple of the path is the set of cells  $\{\alpha_k^2, \alpha_{k+1}^1, \alpha_{k+1}^2\}$  such that  $\alpha_{k+1}^1, \alpha_{k+1}^2 \in V$  and  $\alpha_k^2 \succ \alpha_{k+1}^1$ ,  $\alpha_k^2 \neq \alpha_{k+1}^2$ , that allows growing the  $QV$ -path to  $(0,1)$ -path  $v_k^2 \rightarrow v_{k+1}^1 \rightarrow v_{k+1}^2 \rightarrow v_{k+2}^1 \rightarrow \dots \rightarrow v_{r-1}^2 \rightarrow v_r^1$  related to a  $V$ -path  $\alpha_{k+1}^1, \alpha_{k+1}^2, \alpha_{k+2}^1, \alpha_{k+2}^2, \dots, \alpha_{r-1}^2, \alpha_r^1$ .*

Consider the example of the  $(1,2)$ -path  $v_\sigma \rightarrow v_\alpha$  in Figure 4.2d. The cell  $\sigma^2$  in  $K$  is related to  $v_\sigma$  in  $M$  (see Figure 4.2a). The expanding triple is composed by the cells  $\sigma_o^2$ ,  $\sigma^1$  and  $\sigma_d^2$ , as shown in Figure 4.4.

**Lemma 4.4.** *A  $(0,1)$ -path  $v_0^1 \rightarrow v_1^0 \rightarrow v_1^1 \rightarrow v_2^0 \rightarrow \dots \rightarrow v_{k-1}^1 \rightarrow v_k^0$  has no expanding triple if the 0-cell,  $\alpha^0 \in K$ , related to  $v_k^0$  is critical.*

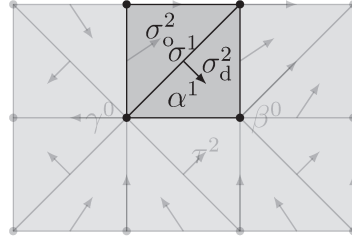


Figure 4.4: Expanding triple of a (1, 2)-path.

*Proof.* If  $\alpha^0$  is not critical then by definition it is paired in  $V$ . A pair  $\alpha_i^0 \prec \beta_i^1$  in a  $V$ -path is always followed by a 0-cell  $\alpha_{i+1}^0$  which is a face of  $\beta_i^1$  other than  $\alpha_i^0$  (Definition 3.4). This face must also exist since by Lemma 4.1 every 1-cell has two faces. If  $\alpha^0$  is critical then it is not paired in the vector field and, therefore, no expanding triple can be found.  $\square$

**Lemma 4.5.** *A (1, 2)-path  $v_k^2 \rightarrow v_{k+1}^1 \rightarrow v_{k+1}^2 \rightarrow v_{k+2}^1 \rightarrow \dots \rightarrow v_{r-1}^2 \rightarrow v_r^1$  has no expanding triple if one of two conditions is true:*

1. *the cell  $\alpha_k^2$ , related to  $v_k^2$ , is critical;*
2. *the cell  $\alpha_k^2$  is paired with a 1-cell,  $\alpha^1 \in V$ , which has no coface in  $K$  other than  $\alpha^2$ .*

*Proof.* Condition 1. If  $\alpha^2$  is critical, then it is not paired in  $V$  and therefore no expanding triple out  $\alpha^2$  can be found. Condition 2. If  $\alpha^2$  is not critical, then it is paired in  $V$ , with a 1-cell,  $\alpha^1$ , according to Definition 3.3. The 1-cell is in  $K$  since all the faces of a cell must be in  $K$ . Given the pair cell  $\alpha^1$ , if its two cofaces are in  $K$  (see Lemma 4.2), then a expanding triple for the path is found since there is coface  $\alpha_{k-1}^2 \neq \alpha_k^2$ . Otherwise, if there is no other coface other than  $\alpha_k^2$ , then no expanding triple can be found.  $\square$

**Lemma 4.6.** *The expanding triple of a (0, 1)-path is unique.*

*Proof.* If  $v_0^1 \rightarrow v_1^0 \rightarrow \dots \rightarrow v_{k-2}^1 \rightarrow v_{k-1}^0$  with related  $V$ -path  $\alpha_1^0, \alpha_1^1, \dots, \alpha_{k-2}^1, \alpha_{k-1}^0$  is the (0, 1)-path that have a expansion, suppose the expanding triple is not unique. Let's call two of these expansions

$$\alpha_1^0, \alpha_1^1, \dots, \alpha_{k-2}^1, \alpha_{k-1}^0, \beta_{k-1}^1, \beta_k^0 \quad (4.1)$$

and

$$\alpha_1^0, \alpha_1^1, \dots, \alpha_{k-2}^1, \alpha_{k-1}^0, \gamma_{k-1}^1, \gamma_k^0 \quad (4.2)$$

If  $\beta_{k-1}^1 \neq \gamma_{k-1}^1$  it means there are pairs  $\alpha_{k-1}^0 \prec \beta_{k-1}^1$  and  $\alpha_{k-1}^0 \prec \gamma_{k-1}^1$  in  $V$ . However, by definition, any cell is paired once in  $V$ . Consequently  $\alpha_{k-1}^1 = \beta_{k-1}^1 = \gamma_{k-1}^1$ . The paths can still be different if  $\beta_k^0 \neq \gamma_k^0$ . However, by Definition 3.3, and knowing that  $\alpha_{k-1}^1 = \beta_{k-1}^1 = \gamma_{k-1}^1$ , the 1-cell  $\alpha_{k-1}^1$  should have three faces:  $\alpha_{k-1}^0$ ,  $\beta_k^0$  and  $\gamma_k^0$ . That's not possible, what means  $\beta_k^0 = \gamma_k^0$ . Therefore, the expanding triple of a (0, 1)-path is unique.  $\square$

**Lemma 4.7.** *The expanding triple of a  $(1, 2)$ -path is unique.*

*Proof.* If  $v_{k+1}^2 \rightarrow v_{k+2}^1 \rightarrow \dots \rightarrow v_{r-1}^2 \rightarrow v_r^1$  with related  $V$ -path  $\alpha_{k+2}^1, \alpha_{k+2}^2, \dots, \alpha_{r-1}^2, \alpha_r^1$  is a  $(1, 2)$ -path that have a expansion, suppose the expanding triple is not unique. Let's call two of these expansions

$$\beta_{k+1}^1, \beta_{k+1}^2, \alpha_{k+2}^1, \alpha_{k+2}^2, \dots, \alpha_{r-1}^2, \alpha_r^1 \quad (4.3)$$

and

$$\gamma_{k+1}^1, \gamma_{k+1}^2, \alpha_{k+2}^1, \alpha_{k+2}^2, \dots, \alpha_{r-1}^2, \alpha_r^1 \quad (4.4)$$

If  $\beta_{k+1}^2 \neq \gamma_{k+1}^2$  then the 1-cell  $\alpha_{k+2}^1$  should have three cofaces:  $\alpha_{k+2}^2$ ,  $\beta_{k+1}^2$  and  $\gamma_{k+1}^2$ . That's not possible, what means  $\beta_{k+1}^2 = \gamma_{k+1}^2$ . Since  $\alpha_{k+1}^2 = \beta_{k+1}^2 = \gamma_{k+1}^2$ , the paths can differ if the pairs  $\beta_{k+1}^1, \beta_{k+1}^2$  and  $\gamma_{k+1}^1, \gamma_{k+1}^2$  are in  $V$  and  $\beta_{k+1}^1 \neq \gamma_{k+1}^1$ . By definition of  $V$  any cell is paired once and since  $\beta_{k+1}^2 = \gamma_{k+1}^2$  then  $\beta_{k+1}^1 = \gamma_{k+1}^1$ . Therefore, the expanding triple of a  $(1, 2)$ -path is unique.  $\square$

Algorithm 4.2 shows how to retrieve a expanding triple given a cell  $\alpha^0$  in the input vector field  $V$ . The algorithm sets the input cell as the origin cell and then retrieves the 1-cell paired with  $\alpha^0$ , in  $V$ , and the another face of this 1-cell, which will be the destination 0-cell. The function **Pair** (see Section 3.1) used in the algorithm returns the pair of a cell in the vector field. Lines 1 and 2 of both algorithms are clearly  $O(1)$ . Lines 3 also can be computed in constant time once a 1-cell has exactly two faces and therefore at most one comparison suffices to get the correct face.

---

**Algorithm 4.2:** Get01ExpandingTriple

---

**Input:**  $\alpha^0 \in V$

**Output:**  $\alpha_o^0$ ,  $\alpha^1$  and  $\alpha_d^0$

1  $\alpha_o^0 \leftarrow \alpha$

2  $\alpha^1 \leftarrow \text{Pair}(\alpha_o^0)$

3  $\alpha_d^0 \leftarrow \beta^0 \prec \alpha^1$  such that  $\beta^0 \neq \alpha_o^0$

---

Retrieving the expanding triple in a  $(1, 2)$  is a similar computation except that the notion of origin and destination may seem inverted. But actually it will be agreeing with the flow and the backward nature already pointed out. Algorithm 4.3 show how to get the triple of cells to expand a  $(1, 2)$ -path. The same reasoning as in the previous algorithm can be used here to argue that all lines take constant time to be computed.

A expanding triple of a cell  $\alpha$  in  $K$  together with its related cell  $v \in M$  are all that is needed to expand a path. Algorithms 4.4 and 4.5 compute, respectively, the extensions of a  $(0, 1)$ - and of a  $(1, 2)$ -path. Both cases are similar, however we maintain the algorithms

---

**Algorithm 4.3:** Get12ExpandingTriple

---

**Input:**  $\alpha^2 \in V$ **Output:**  $\alpha_o^2$ ,  $\alpha^1$  and  $\alpha_d^2$ 

- 1  $\alpha_d^2 \leftarrow \alpha$
  - 2  $\alpha^1 \leftarrow \text{Pair}(\alpha_d^2)$
  - 3  $\alpha_o^2 \leftarrow \beta^2 \succ \alpha^1$  such that  $\beta^2 \neq \alpha_d^2$
- 

---

**Algorithm 4.4:** Expand01Path

---

**Input:** 0-cell  $v \in M$ ; expanding triple  $\alpha_o$ ,  $\alpha^1$  and  $\alpha_d \in K$ **Output:**  $t$  and  $\alpha_d$ 

- 1 create 0-cell  $u \in M$  for 1-cell  $\alpha^1$
  - 2 create 0-cell  $t \in M$  for 0-cell  $\alpha_d$
  - 3 create 1-cell  $e_1$  from  $v$  to  $u$
  - 4 create 1-cell  $e_2$  from  $u$  to  $t$
  - 5  $\text{Splice}(\text{Sym}(e_1), e_2)$
  - 6  $e \leftarrow \text{PrevInRing}(e_1, v)$
  - 7  $\text{Splice}(e, e_1)$
  - 8  $e \leftarrow \text{PrevInRing}(\text{Sym}(e_2), t)$
  - 9  $\text{Splice}(e, \text{Sym}(e_2))$
- 

separated to be consistent with the choices of directions of growth for the two types of paths and to make it easy to comprehend the differences.

The connection of the two new edges  $e_1$  and  $e_2$  is accomplished computing the splice operator (lines numbered 5). A new function is introduced in these algorithms to help to adjust the rings of 1-cells out of origin and destination 0-cells. The function, called **PrevInRing** receives as input a 0-cells  $v$  and a 1-cells  $e_i$  which will be inserted into the ring of  $v$ . The function returns the edge  $e$ , already in the ring of  $v$ , such that it is the one to precede the new edge  $e_i$  in the ring of  $v$ . The **Splice** function in lines 6 and 8 make the adjustments.

Almost all the lines of Algorithms 4.4 and 4.5 clearly take constant time and the new function **PrevInRing** can also be expected to run in constant time. The 1-skeleton of a 2-complex is a planar graph. From graph theory the average degree of such a graph is expected to be less than 6 [38]. Likewise, it is possible to count the average degree of a 2-cell by considering the dual of a cell complex. Again, the 1-skeleton is a planar graph and the same results apply. The number of 1-cells in the rings of 0-cells of  $M$  is less than 6 in average. Therefore, the algorithms are expected to be  $O(1)$ .

As state at the beginning of this section, expanding a  $(0, 1)$ -path  $v_0^1 \rightarrow v_1^0 \rightarrow v_1^1 \rightarrow \dots \rightarrow v_{k-2}^1 \rightarrow v_{k-1}^0$  and a  $(1, 2)$ -path  $v_{k+1}^2 \rightarrow v_{k+2}^1 \rightarrow v_{k+2}^2 \rightarrow \dots \rightarrow v_{r-1}^2 \rightarrow v_r^1$  means

**Algorithm 4.5:** Expand12Path**Input:** 0-cell  $v \in M$ ; expanding triple  $\alpha_o, \alpha^1$  and  $\alpha_d \in K$ **Output:**  $t$  and  $\alpha_o$ 

- 1 create 0-cell  $u \in M$  for 1-cell  $\alpha^1$
- 2 create 0-cell  $t \in M$  for 2-cell  $\alpha_o$
- 3 create 1-cell  $e_1$  from  $u$  to  $v$
- 4 create 1-cell  $e_2$  from  $t$  to  $u$
- 5  $\text{Splice}(e_1, \text{Sym}(e_2))$
- 6  $e \leftarrow \text{PrevInRing}(\text{Sym}(e_1), v)$
- 7  $\text{Splice}(e, \text{Sym}(e_1))$
- 8  $e \leftarrow \text{PrevInRing}(e_2, t)$
- 9  $\text{Splice}(e, e_2)$

obtaining the longer paths  $v_0^1 \rightarrow v_1^0 \rightarrow v_1^1 \rightarrow \dots \rightarrow v_{k-2}^1 \rightarrow v_{k-1}^0 \rightarrow v_{k-1}^1 \rightarrow v_k^0$  and  $v_k^2 \rightarrow v_{k+1}^1 \rightarrow v_{k+1}^2 \rightarrow v_{k+2}^1 \rightarrow v_{k+2}^2 \rightarrow \dots \rightarrow v_{r-1}^2 \rightarrow v_r^1$ . That's exactly what the previous algorithms do. Consider again the paths obtained after the initialization steps, those were of the form  $v_0^1 \rightarrow v_1^0$  and  $v_{r-1}^2 \rightarrow v_r^1$ . Now, given the paths shown in Figures 4.2b and 4.2d, the vertices  $v_\beta$  and  $v_\sigma$  and the expanding triples of Figures 4.3 and 4.4, the application of the algorithms produce paths of the form  $v_0^1 \rightarrow v_1^0 \rightarrow v_1^1 \rightarrow v_2^0$  and  $v_{r-2}^2 \rightarrow v_{r-1}^1 \rightarrow v_{r-1}^2 \rightarrow v_r^1$  as shown in Figures 4.5a and 4.5b.

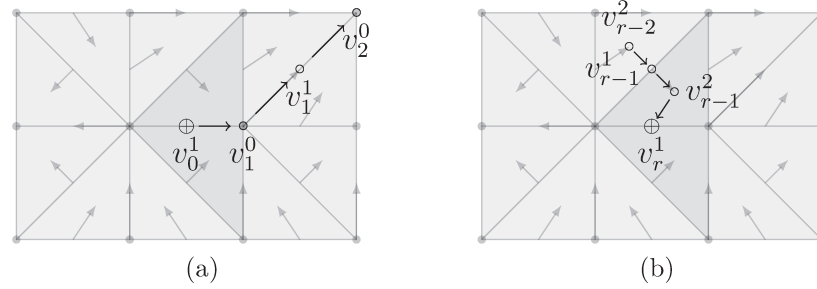


Figure 4.5: One step expanded  $QV$ -paths obtained with the expanding triples of the initial  $(0, 1)$ - and  $(1, 2)$ -paths of Figures 4.2b and 4.2d.

**Stop Conditions**

The paths must be expanded until a stop condition is met. From Lemmas 4.3 and 4.4, a path can not be expanding any further when:

- the last cell  $\alpha^0$  of a  $(0, 1)$ -path is a critical cell;

- the last cell  $\alpha^2$  of a  $(1, 2)$ -path is a critical cell;
- the last cell  $\alpha^2$  of a  $(1, 2)$ -path has only one coface in  $K$ .

From these conditions it is possible to expand a whole  $QV$ -path. However some additional stop conditions will be introduced. These are based on the fact that  $V$ -paths can merge or branch. Considering these cases is important to maintain the consistence of the quad-edge data structure.

Consider a particular 0-cell  $\alpha^0$  in a complex  $K$  and suppose it is paired with a 1-cell in the vector field  $V$  of the complex. Let also  $\alpha^0$  be a face of  $k$  1-cells in a complex  $K$ . Except for the 1 cell which is paired with  $\alpha^0$ , each one of the other 1-cells either is paired (with a 0-cell or a 2-cell) or is a critical 1-cell. If  $l \leq k$  of these 1-cells are paired with a 0-cell, then there are  $l$   $V$ -paths containing the cell  $\alpha^0$  and therefore at least  $l$  paths merge into  $\alpha^0$ . Figure 4.6 shows an example of cell at which  $V$ -paths merge. The 0-cell  $\alpha^0$  in Figure 4.6 is paired with one 1-cell and has five other cofaces of which three are paired with a 0-cell.

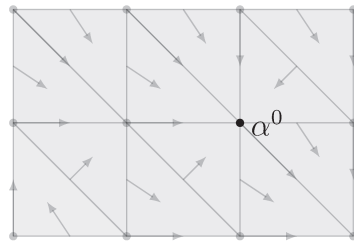


Figure 4.6: Example of a 0-cell shared by three  $V$ -paths.

Now let  $\alpha^2$  be a 2-cell in  $K$  paired in  $V$  with one of its 1-cell faces. If the  $\alpha^2$  has  $k$  1-cell faces then these cells may be paired with a 0-cell, paired with a 2-cell or be a critical cell. If  $l \leq k$  of them are paired with 2-cells, the  $V$ -path containing  $\alpha^2$  will have  $l$   $V$ -paths out of it, in other words, the path branches into  $l$   $V$ -paths. Figure 4.7 shows a  $V$ -path which branches into two  $V$ -paths.

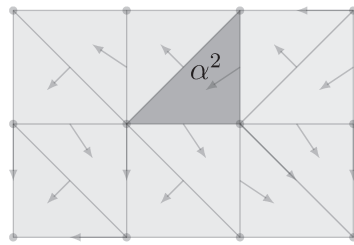


Figure 4.7: Example of a 2-cell shared by two  $V$ -paths.

These examples show that that  $V$ -paths can merge or branch according to their type. In the case of the two dimensional complexes it is possible to say that:  $(0, 1)$ -paths can merge but can't branch and  $(1, 2)$ -paths can branch but can't merge. From the examples it has been shown that  $(0, 1)$ -paths merge in a 0-cell and  $(1, 2)$ -paths branch in a 2-cell. Now the following complete the statements.

**Lemma 4.8.**  *$(0, 1)$ -paths in  $V$  do not merge in a 1-cell.*

*Proof.* Given two  $V$ -paths, suppose these paths merge. Suppose the paths merge in a 1-cell, then there exists a pair  $\alpha_i^0 \prec \alpha_i^1$  in the first path and a pair  $\beta_j^0 \prec \beta_j^1$  in the second path such that  $\alpha_i^1 = \beta_j^1$  and  $\alpha_i^0 \neq \beta_j^0$ . But that means the intersecting 1-cell is paired twice in  $V$ , which is not possible by definition.  $\square$

**Lemma 4.9.**  *$(0, 1)$ -paths in  $V$  do not branch.*

*Proof.* Suppose a  $(0, 1)$ -path branches, that is, the path either becomes two (or more) in a 0-cell or in a 1-cell. If the path becomes two in a 0-cell then this 0-cell must have been paired twice in the vector field  $V$ . However, that's not possible by definition. If the branching occurs in a 1-cell of the path, this cell, by definition, belongs to a pair  $\alpha^0 \prec \alpha^1$  in the path. The face  $\alpha^0$  is common to both paths, therefore, the 1-cell should have two other faces that are not common to the paths. That's absurd since a 1-cell has exactly 2 faces. Therefore, a  $(0, 1)$ -paths in  $V$  do not branch.  $\square$

**Lemma 4.10.**  *$(1, 2)$ -paths in  $V$  do not merge.*

*Proof.* Suppose a  $(1, 2)$ -path merges. A merging means that along the path either two (or more) paths become one in a 1-cell or in a 2-cell. If the paths become one in a 1 cell  $\alpha^1$ , then besides the coface it is paired in  $V$ , it must have two other cofaces belonging to the different paths. That is not possible since 1-cells have at most 2 cofaces. If the paths merge in a 2-cell, then this cell should be paired twice in  $V$ , but that is not possible by definition.  $\square$

**Lemma 4.11.**  *$(1, 2)$ -paths in  $V$  do not branch in a 1-cell.*

*Proof.* If a path branches at a 1-cell then this cell is paired twice in  $V$ , but that is not possible by definition.  $\square$

Both the merge and branch cases give rise to new stop cases in which a  $QV$ -path stops being expanded if it arrives at a cell in a path already processed. Figure 4.8 depicts what happens when two  $(0, 1)$ -paths merge. Consider the two paths shown Figure 4.8a. Suppose the path ending at the 0-cell  $v$  is being currently processed and the other path is a  $QV$ -path already processed. When the current path is expanded from  $v$ , the two

paths merge as in Figure 4.8b. From that point on, the expansion of the current path is exactly the same as the one previously expanded, which is a consequence from the fact that  $(0, 1)$ -paths do not branch (see Lemma 4.9). In such a case, the expansion can be stopped.

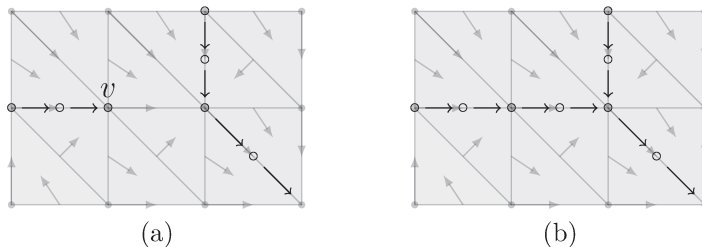


Figure 4.8: Example of  $QV$ -paths that merge.

Likewise, it is simple to deal with branch cases. In Figure 4.9a, two  $(1, 2)$ -paths are shown, one previously processed and one expanded up to the 0-cell  $v$ . Notice that since  $(1, 2)$ -paths are traversed in a backwards fashion, algorithmically the branch case of  $(1, 2)$ -paths is similar to the merge case of  $(0, 1)$ -paths. Actually, with the backwards expanding process, a branching case of a  $(1, 2)$ -path becomes a merge (or pseudo-merge) case and makes it possible to deal with both cases the same way. With the expanding triple from  $v$  the path reaches a vertex already processed (Figure 4.9b) and the path from this point on is the same (consequence of Lemma 4.10). Therefore, no more expansion is needed to be computed.

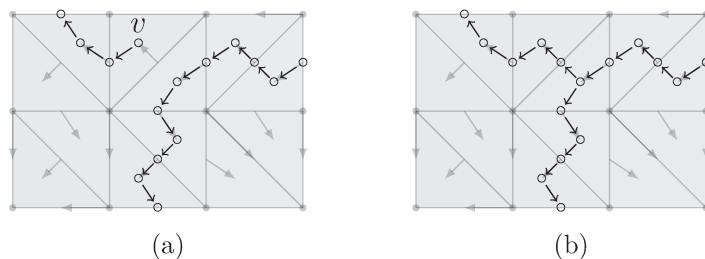


Figure 4.9: Example of  $QV$ -paths that branch.

The expanding triple algorithms are modified to incorporate the stopping conditions. Algorithms 4.6 and 4.7 show the modified versions with stopping conditions. In Algorithm 4.6, line 1 tests if the input cell is critical or is in a path already processed, that is, the current path will merge with a previous path. If the condition is true the expanding triple is returned NIL.

Line 1 of Algorithm 4.7 tests the same conditions of critical and already processed path, that is, the current path will branch (merge algorithmically as already argued) with



---

**Algorithm 4.6:** Get01ExpandingTriple

---

**Input:**  $\alpha^0 \in V$   
**Output:**  $\alpha_o^0$ ,  $\alpha^1$  and  $\alpha_d^0$

```

1 if  $\alpha^0$  is critical or already processed then
2   |  $\alpha_o^0 \leftarrow \alpha^1 \leftarrow \alpha_d^0 \leftarrow NIL$ 
3 else
4   |  $\alpha_o^0 \leftarrow \alpha$ 
5   |  $\alpha^1 \leftarrow \text{Pair}(\alpha_o^0)$ 
6   |  $\alpha_d^0 \leftarrow \beta^0 \prec \alpha^1$  such that  $\beta^0 \neq \alpha_o^0$ 
7 end
```

---

a previous path. Line 7 tests the stopping condition when a 1-cell has only one coface in the complex. All the new tests and operations clearly also take constant time, therefore, the  $O(1)$  complexity for obtaining a expanding triple is maintained.

---

**Algorithm 4.7:** Get12ExpandingTriple

---

**Input:**  $\alpha^2 \in V$   
**Output:**  $\alpha_o^2$ ,  $\alpha^1$  and  $\alpha_d^2$

```

1 if  $\alpha^2$  is critical or already processed then
2   |  $\alpha_o^2 \leftarrow \alpha^1 \leftarrow \alpha_d^2 \leftarrow NIL$ 
3 else
4   |  $\alpha_d^2 \leftarrow \alpha$ 
5   |  $\alpha^1 \leftarrow \text{Pair}(\alpha_d^2)$ 
6   |  $\alpha_o^2 \leftarrow \beta^2 \succ \alpha^1$  such that  $\beta^2 \neq \alpha_d^2$ 
7   | if  $\alpha_o^2 = nil$  then
8     |  $\alpha_o^2 \leftarrow \alpha^1 \leftarrow \alpha_d^2 \leftarrow NIL$ 
9   | end
10 end
```

---

Given the new algorithms, a path can be fully extracted as show in Algorithm 4.8. The algorithm has as input a cell of the complex  $K$  and the corresponding cell of the complex  $M$ . These are from one pair of cells of a  $QV$ -path initialized in Algorithm 4.1. The algorithm will expand the path from these cells until a stop condition is met. The stop conditions are tested in the while loop by checking if the expanding triple is valid, which means no cell has value NIL. While a expanding triple is found and processed, the last cells of the path are updated and a new expanding triple is searched for. The dimension of a  $p$ -cell  $\sigma$  is given by the function  $\text{dim}$ ,  $\text{dim } \sigma = p$ . This functions is tested in the algorithm to guide what type of path is being processed.

---

**Algorithm 4.8:** ProcessQVPath

---

**Input:**  $\lambda \in K$  and  $v_\lambda \in M$ **Output:** last 0-cell  $v$  created in  $M$ 

```

1  $v \leftarrow v_\lambda$ 
2  $\alpha \leftarrow \lambda$ 
3 if  $\dim \alpha = 0$  then
4    $\{\alpha_o, \alpha^1, \alpha_d\} \leftarrow \text{Get01ExpandingTriple}(\alpha)$ 
5 else
6    $\{\alpha_o, \alpha^1, \alpha_d\} \leftarrow \text{Get12ExpandingTriple}(\alpha)$ 
7 end
8 while expanding triple  $\{\alpha_o, \alpha^1, \alpha_d\}$  is valid do
9   if  $\dim \alpha = 0$  then
10     $\{(\lambda, v_\lambda)\} \leftarrow \text{Expand01Path}(v, \alpha_o, \alpha^1, \alpha_d)$ 
11  else
12     $\{(\lambda, v_\lambda)\} \leftarrow \text{Expand12Path}(v, \alpha_o, \alpha^1, \alpha_d)$ 
13  end
14   $v \leftarrow v_\lambda$ 
15   $\alpha \leftarrow \lambda$ 
16  if  $\dim \alpha = 0$  then
17     $\{\alpha_o, \alpha^1, \alpha_d\} \leftarrow \text{Get01ExpandingTriple}(\alpha)$ 
18  else
19     $\{\alpha_o, \alpha^1, \alpha_d\} \leftarrow \text{Get12ExpandingTriple}(\alpha)$ 
20  end
21 end

```

---

From the previous discussion, most lines of the algorithm take constant time to be computed. The loop of line 8 is executed  $O(L)$  times which is the length of a  $V$ -path. Therefore the algorithm is linear on the size of the  $V$ -path.

The paths of a discrete Morse complex are exactly the  $QV$ -paths with critical cells as endpoints. From previous discussions we know that there are  $(1, 2)$ -paths that do not end at a critical 2-cell. That follows from the second condition of Lemma 4.5 which happens when a path arrives at the boundary of the complex  $K$  and a 1-cell has only one coface.

Therefore, to obtain only the paths of the Morse complex it suffices removing a  $(1, 2)$ -path if one of its endpoints is not a critical cell. Algorithm 4.9 performs the removal of the path. The input of the algorithm is the last 0-cell expanded of the  $QV$ -path. The  $QV$ -path is traversed from the input vertex towards the another endpoint (loop of line 2). Along the traversal, the final cells of path are deleted. When the initial 0-cell vertex is reached the process stops since the 0-cell is a saddle point. All cells of the  $QV$ -path where removed. Remember from Section 2.2 that  $\text{Edge}(v)$  returns a 1-cell  $e$  adjacent to

$v$ . The algorithm performs  $O(1)$  operations and the loop is executed once for each vertex in the path, therefore the algorithm is linear on the size of the path.

---

**Algorithm 4.9:** RemoveQVPath

---

**Input:**  $v_{\alpha^2} \in M$   
**Output:** Complex  $M$  without path ending in  $v_{\alpha^2}$

```

1  $v_o \leftarrow v_{\alpha^2}$ 
2 repeat
3    $e \leftarrow \text{Edge}(v_o)$ 
4    $v_d \leftarrow \text{Orig}(e)$ 
5   if  $v_o = v_d$  then  $v_d \leftarrow \text{Dest}(e)$ 
6   delete  $v_o$ 
7   delete  $e$ 
8    $v_o \leftarrow v_d$ 
9 until  $v_o$  is not critical

```

---

It is now possible to extract all  $QV$ -paths in a discrete Morse complex. The loop of line 1 goes through all saddles of the input complex. Line 2 and the conditionals take constant time while the other functions called are linear on the size of the paths. If the complex has  $N$  saddles, the algorithm takes  $O(NL)$ .

Since the paths extracted can merge or branch, we call the complex an “entangled complex”. In the following section, we will consider these cases to produce a disentangled complex, that is, a complex without united paths.

### 4.1.3 Disentangled Complex

In applications such as visualization of the Morse complex it is often important that cases like merging and branching do not occur. In the following it will be shown how to deal with such situations, separating merged and branched paths to keep them independent of each other.

It has been previously seen that  $QV$ -paths can merge or branch at a given 0-cell along their extent. We will call such 0-cell a knot.

**Definition 4.5.** *A knot is a 0-cell  $v \in M$  such that it is not critical and its degree (number of paths arriving and leaving) is greater than two.*

Figure 4.10a shows a knot  $v$  where paths merge, it has degree three ( $\deg(v) = 3$ ) and is not critical. A knot of branching paths is shown in Figure 4.10b.

If a knot is present it means that  $k$  paths have a common subpath that occurs after a merging case or before a branching case. The objective of this section is to disentangle all

**Algorithm 4.10:** ExtractQVPaths**Input:**  $K$  image cell complex;  $V$  vector field**Output:** Discrete Morse complex  $M$ 

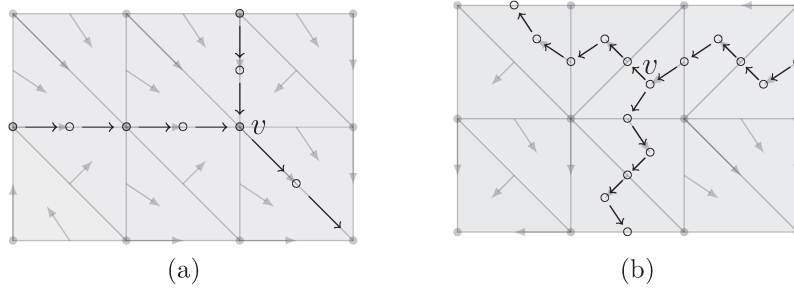

---

```

1 forall the  $\alpha^1 \in K$  such that  $\alpha^1$  is critical do
2    $\{(\beta^0, v_\beta), (\gamma^0, v_\gamma), (\sigma^2, v_\sigma), (\tau^2, v_\tau)\} \leftarrow \text{InitVPaths}(\alpha^1)$ 
3    $v \leftarrow \text{ProcessQVPath}(\beta^0, v_\beta)$ 
4    $v \leftarrow \text{ProcessQVPath}(\gamma^0, v_\gamma)$ 
5    $v \leftarrow \text{ProcessQVPath}(\sigma^2, v_\sigma)$ 
6   if  $v$  is not critical then
7     |  $\text{RemoveQVPath}(v)$ ;
8   end
9    $v \leftarrow \text{ProcessQVPath}(\tau^2, v_\tau)$ 
10  if  $v$  is not critical then
11    |  $\text{RemoveQVPath}(v)$ ;
12  end
13 end

```

---

Figure 4.10: Examples of knots for  $(0,1)$ - and  $(1,2)$ -paths in a complex  $M$ .

knots so that the  $k$  paths are separated along their common subpath and are maintained independent of each other. In other words, we want to produce a discrete Morse complex for which all paths do not have knots along their extent.

The algorithm developed in this section can be thought of as a method that pushes forward (backward) a knot along the common subpath of merging (branching) paths. The knots are pushed forward (backward) until reaching a critical endpoint. Algorithm 4.11 performs the basic operation. It works locally at a knot separating the paths one step forward (backward). The algorithm receives as input a knot  $v_1$ , the 0-cell to which  $v_1$  is pushed towards, and the type of path ( $(0,1)$ -path or  $(0,2)$ -path) it is in. The type of path is used as a flag to know if the knot should be pushed forward or backward.

The edge  $e_1$  between  $v_1$  and  $v_2$  is the common subpath to be separated into the  $k$  paths entangled in  $v_1$ . In the case of Figure 4.11a three paths merge in  $v_1$ , the knot will

---

**Algorithm 4.11:** DisentangleKnot

---

**Input:** Knot  $v_1, v_2$  which is the 0-cell  $v_1$  is pushed towards,  $t$  path type ( $t = 0$  if  $(0, 1)$ -path or  $t = 1$  if  $(1, 2)$ -path).

**Output:** Complex  $M$  with a knot removed

```

1  $e_1 \leftarrow$  1-cell  $e$  such that  $\text{Orig}(e) = v_1$  and  $\text{Dest}(e) = v_2$ 
2  $e_2 \leftarrow \text{Oprev}(\text{Sym}(e_1))$ 
3  $e \leftarrow \text{Onext}(e_1)$ 
4  $e_f \leftarrow e_1$ 
5 repeat
6    $v_3 \leftarrow \text{Dest}(e)$ 
7    $\text{Splice}(e, e_1)$ 
8   create 0-cell  $v$  in  $M$  close to  $v_1$ 
9    $\text{Orig}(e) \leftarrow v$ 
10  if  $t = 0$  then
11    | create 1-cell  $e_n$  from  $v$  to  $v_2$ 
12  else
13    | create 1-cell  $e_n$  from  $v_2$  to  $v$ 
14  end
15  create edge  $e_n$  from  $v$  to  $v_2$ 
16   $\text{Splice}(e, e_n)$ 
17   $\text{Splice}(\text{Sym}(e_n), e_2)$ 
18   $e \leftarrow \text{Onext}(e_1)$ 
19   $e_2 \leftarrow \text{Onext}(e_2)$ 
20 until  $e = e_f$ 
21 delete  $v_1$ 
22 delete  $e_1$ 
```

---

be pushed forward to  $v_2$  along the edge  $e_1$ . The loop of line 5 separates, at each iteration, one of the  $k$  paths. Initially, the path is disconnected from the ring of 1-cells of  $v_1$  (line 7). A new 0-cell  $v$  is then created to substitute  $v_1$  in the path and to be the 0-cell through which the path passes. The 1-cell  $e_n$  is created to reconnect the path to  $v_2$ . At the end of the loop all paths are disentangled by one step. Figure 4.11b shows the result for the example path. A new application of Algorithm 4.11 will disentangle the paths one step further, getting the paths of Figure 4.11c.

The interesting points for analyzing the complexity in this algorithm is the loop of line 5 and the operation of line 1. In both case, the execution involves traversing the rings of 1-cells. As already discussed in Section 17, this rings in a planar graph are expected to have average degree of 6. Therefore, we can argue that the algorithm is expected to be  $O(1)$ .

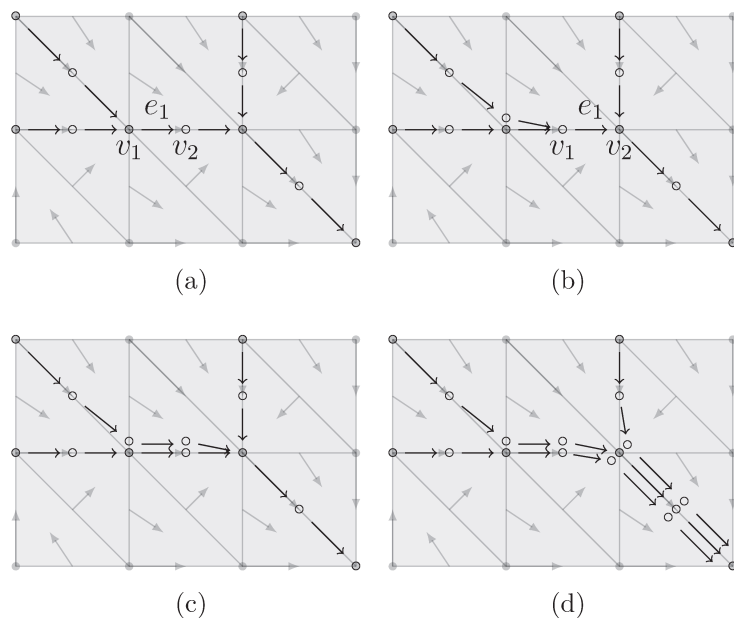


Figure 4.11: Example of untangling knots of a merged path.

It suffices to continue pushing forward (backward) the knot vertices to fully separate paths. The resulting paths are then disjoint. For visualization purposes and to maintain the planarity of the complexes, the geometrical position of the paths can be perturbed infinitesimally so that they are positioned parallel to each other. Figure 4.11d shows the result of disentangling the knots.

Similar to the approach to extract the  $QV$ -paths, we proceed from saddles in  $M$  and follow all of its paths disentangling the knots until another critical cell is found. The computation is performed by Algorithm 4.12. The outer loop of line 1 goes through all critical saddles and the loop of line 4 processes all paths leaving or arriving at the saddles. Finally, the loop of line 11 traverses a path searching for knots and disentangling them. The functions **Next** and **Prev** respectively return the next and the previous 0-cell following a given 0-cell in a  $QV$ -path (see Section 4.1.1). The first one is used for  $(0, 1)$ -paths and the second one for  $(1, 2)$ -paths.

The loop of line 11 traverses the whole path and takes linear time on the length of the path since each operation inside the loop take constant time. The loops of line 1 and 4, together, process all the paths and therefore are linear on the number of paths. If the number of saddles is  $N$  and each saddle can have up to 4 paths, the algorithm takes time  $O(NL)$ .

---

**Algorithm 4.12:** DisentangleComplex

---

**Input:** complex  $M$ .  
**Output:** Complex  $M$  with all knots removed

```

1 for all saddles  $v^1 \in M$  do
2    $e \leftarrow \text{Edge}(v^1)$ 
3    $e_f \leftarrow e$ 
4   repeat
5      $v_1 \leftarrow \text{Dest}(e)$ 
6     if  $\dim(v_1) = 0$  then
7        $t \leftarrow 0$ 
8     else
9        $t \leftarrow 1$ 
10    end
11    while  $v_1$  is not critical do
12      if  $t = 0$  then
13         $v_2 \leftarrow \text{Next}(v_1)$ 
14      else
15         $v_2 \leftarrow \text{Prev}(v_1)$ 
16      end
17      if  $\deg(v_1) > 2$  then
18         $\text{DisentangleKnot}(v_1, v_2, t)$ 
19      end
20       $v_1 \leftarrow v_2$ 
21    end
22     $e \leftarrow \text{Onext}(e)$ 
23  until  $e = e_f$ 
24 end
```

---

#### 4.1.4 Simplified Morse Complex

Even though the complex obtained in the previous section can be used for simple visualizations tasks, it is not efficient for many practical tasks such as computing numbers of Betti, persistent homology or hierarchies of the complex. For such tasks one would desire to obtain the critical cells connected by  $QV$ -paths in constant time, without having to traverse all the path. We present in this section a simplified version of our discrete Morse complex such that the 1-cells connect only critical cells and the in between cells of the  $QV$ -paths are stored in a list attributed to the 1-cells. With such a complex we obtain the objective of a model that is easy to manipulate with local operations but that is also appropriate for visualizations.

The algorithm works on subpaths as the one depicted in Figure 4.12. The subpath is

formed by the set of cells  $\{e, v_1, v_2, v_3, e_1, e_2\}$ . The 1-cell  $e$  has as origin a saddle in  $M$ .

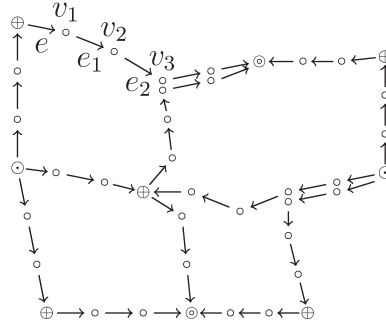


Figure 4.12: Simplification kernel for a subpath of a  $QV$ -path.

We call such set of cells a simplification kernel. Given a 1-cell  $e$ , Algorithm 4.13 retrieves the kernel according to the type of the path which is reflected in the index of the 0-cell  $v_1$  retrieved by the function `Index` in lines 2 and 5. The algorithm clearly takes constant time to retrieve the kernel.

---

**Algorithm 4.13:** SimpKernel

---

**Input:** 1-cell  $e$

**Output:**  $\{v_1, v_2, v_3, e_1, e_2\}$

```

1  $v_1 \leftarrow \text{Dest}(e)$ 
2 if  $\text{Index}(v_1) = 0$  then
3    $v_2 \leftarrow \text{Next}(v_1)$ 
4    $v_3 \leftarrow \text{Next}(v_2)$ 
5 else if  $\text{Index}(v_1) = 2$  then
6    $v_2 \leftarrow \text{Prev}(v_1)$ 
7    $v_3 \leftarrow \text{Prev}(v_2)$ 
8 end
9  $e_1 \leftarrow$  1-cell  $e_o$  such that  $\text{Orig}(e_o) = v_2$  and that  $\text{Dest}(e_o) = v_1$ 
10  $e_2 \leftarrow$  1-cell  $e_o$  such that  $\text{Orig}(e_o) = v_2$  and that  $\text{Dest}(e_o) = v_3$ 

```

---

Given the kernel, Algorithm 4.14 simplifies the subpath. The cells  $v_1$  and  $v_2$  as well as  $e_1$  and  $e_2$  are removed from the complex and the 1-cell  $e$  is connected to  $v_3$ . The result of the simplification on the complex of Figure 4.12 is shown in Figure 4.13a. The loop of line 9 is the costly part of the algorithm. It traverses the ring of 1-cells of  $v_3$  which we already argued is expected to be computed in constant time in average.

The process of finding kernels and simplifying them must be repeated until the 1-cell  $e$  finally connects critical 0-cells. Since the origin of  $e$  is already a critical cell (a saddle) it suffices to check if its destination is also a critical cell. If so, there is no need to search



**Algorithm 4.14:** SimplifyKernel

---

**Input:**  $v_1, v_2, v_3, e_1$  and  $e_2$ .  
**Output:** Complex  $M$  with simplified kernel

- 1  $e_3 \leftarrow \text{Oprev}(\text{Sym}(e_1))$
- 2  $e_4 \leftarrow \text{Oprev}(\text{Sym}(e_2))$
- 3 delete  $e_1$
- 4 delete  $e_2$
- 5 delete  $v_2$
- 6  $\text{Splice}(e_3, e_4)$
- 7  $e \leftarrow \text{Edge}(v_3)$
- 8  $e_f \leftarrow e$
- 9 **repeat**
- 10   |    $\text{Orig}(e) \leftarrow v_3$
- 11 **until**  $e = e_f$
- 12 delete  $v_1$

---

for more kernels of the path. The endpoints of  $e$  in Figure 4.13a still are not both critical, therefore a new kernel is found and simplified. The result is the edge  $e$  of Figure 4.13b that connects critical cells and the process stops.

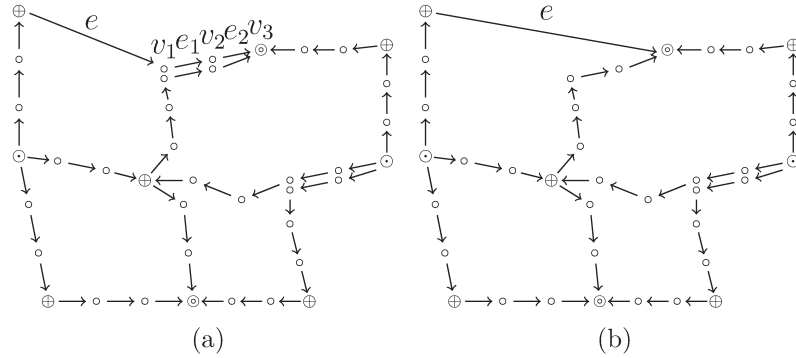


Figure 4.13: Simplification of kernel in Figure 4.12 and the acquisition of a new kernel(a). The simplified path is shown in (b) connecting critical cells.

If the destination of  $e$  is not a critical cell, then we can argue that a kernel can be found. Consider a  $(0, 1)$ -path  $u_0^1 \rightarrow u_1^0 \rightarrow u_1^1 \rightarrow u_2^0 \rightarrow \dots \rightarrow u_{r-1}^1 \rightarrow u_r^0$  as in Definition 4.1. The saddle in the origin of  $e$  is  $u_0^1$  and the destination is  $u_1^0$  in the path. By the definition, the subpath  $u_1^0 \rightarrow u_1^1 \rightarrow u_2^0$  should be present in the path. These are exactly the cells needed to form the kernel,  $v_1$  and  $v_2$  come from a pair in the vector field and  $v_3$  come from a face that must also be in the path. After the simplification performed by the algorithm, a new kernel is searched from  $u_2^0$ . It must also be in a pair and have a face in the path,

consequently forming a new kernel. We can keep going in this process until  $u_r^0$  is reached. This must be a critical cell in  $M$  and also our stop condition. The same thought can be applied to a  $(1, 2)$ -path.

Algorithm 4.15 goes through all paths simplifying the complex. The outer loop (line 1) and the loop of line 4, together, are used to visit all paths. The inner loop (line 5) traverses a path doing the simplifications. Line 7 stores the points of the  $QV$ -path in a list of cells which is an attribute of the 1-cell  $e$ . Therefore, the path can still be retrieved when needed.

---

**Algorithm 4.15:** SimplifyComplex

---

**Input:** Complex  $M$ .

**Output:** Complex  $M$  simplified

```

1 for all saddles  $v^1 \in M$  do
2    $e \leftarrow \text{Edge}(v^1)$ 
3    $e_f \leftarrow e$ 
4   repeat
5     while  $\text{Dest}(e)$  is not critical do
6        $\{v_1, v_2, v_3, e_1, e_2\} \leftarrow \text{SimpKernel}(e)$ 
7       Insert  $v_1$  and  $v_2$  into list of cells of  $e$ ;  $\text{SimplifyKernel}(v_1, v_2, v_3, e_1, e_2)$ 
8     end
9      $e \leftarrow \text{Onext}(e)$ 
10  until  $e = e_f$ 
11 end

```

---

The inner loop takes linear time on the size of the path. Therefore, similar to what happens in the algorithm of the previous section, for disentangling a path, the algorithm also takes  $O(NL)$  time. Therefore, the adding all the steps of the method until getting the final complex results in a  $O(NL)$  algorithm. The resulting complex for our example is shown in Figure 4.14. In this final complex, retrieving the neighborhood information of a critical cell, such as which other critical cells are connected to it, can be done in a simple way. For example, the critical cells connected by a path are retrieved in constant time, there is no need to follow the whole extent of the path. This is important to compute operations such as the star of a cell (informally, the set of cells formed by the cell itself and its cofaces), often used in topological computations.

## 4.2 Experimental Results

This section presents experimental results showing that the constructed discrete Morse complex is useful for analysis of image functions and for important topological computa-

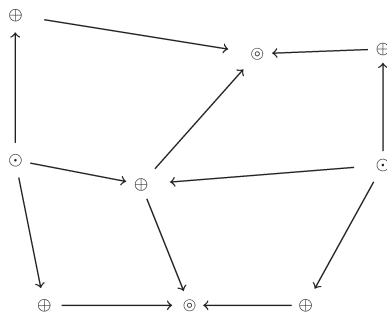


Figure 4.14: The resulting discrete Morse complex.

tions, such as the persistence of critical points [16, 33, 65, 66] and simplification (not to be confused with the simplification of Section 4.1) of the topological structure to reduce noise or produce a hierarchical representation [60, 81, 91].

The following results are similar in power when compared against Morse-Smale [6, 26, 99] and discrete Morse methods for these computations. However, as it is also based on the discrete theory, particular cases such as multiple saddles and also spurious critical points due to intersections of integral curves are not a difficulty. It also presents an improvement in the 2-dimensional computation of the  $V$ -paths, when compared with the breadth-search approach developed by Robins et al. [76] and, as the former techniques, the proposed method is also suitable for producing proper visualizations of topological structures.

We use a set of elevation functions comprising one synthetic image and four real terrain elevation data [90]. The elevation values are represented by 2-byte unsigned integers. The information on the image set is given in Table 4.1. Initially, the images are converted to a 2-dimensional quadrangular complex with pixels as the 0-cells, as described in Section 2.3.

Table 4.1: Images used in the experiments. The second column shows the dimensions of the images and the columns three to five shows the number of cells of each dimension in the respective cell complexes.

Image	Dimensions	Complex (number of cells)		
		0-cells	1-cells	2-cells
Sine	$256 \times 256$	65536	130560	65025
Crater Lake	$336 \times 459$	154224	307653	153430
Cumberland	$1201 \times 1201$	1442401	2882400	1440000
Death Valley	$1201 \times 1201$	1442401	2882400	1440000
Mars	$936 \times 949$	888264	1774643	886380

The synthetic elevation function is a sample of  $h(x, y) = \sin x + \sin y$  multiplied by an

Table 4.2: Number of critical points of the images.

Image	Morse Complex		
	Minima	Saddle	Maxima
Sine	49	84	36
Crater Lake	355	713	359
Cumberland	8058	23138	15081
Death Valley	27964	46319	18356
Mars	568	4293	3726

exponential  $g(x, y) = \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$ , with  $x$  and  $y$  in the interval  $[-40, 40]$  and  $\sigma = 6$ . The image is shown in Figure 4.15a and its resulting discrete Morse complex in Figure 4.15b. The result shows that our algorithm produces quadrangular 2-cells formed by critical points of minima and maxima always adjacent to saddles, as it is expected for a Morse complex for such a function.

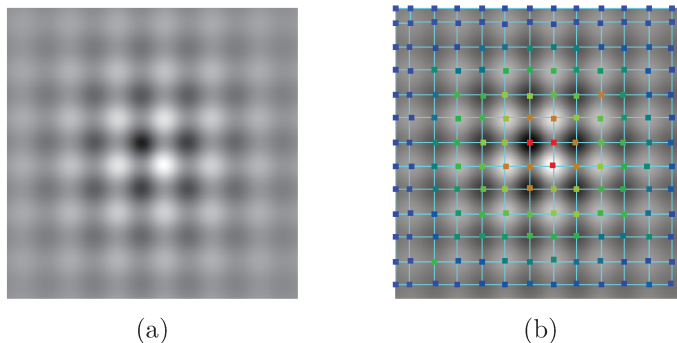


Figure 4.15: Discrete Morse complex of the synthetic sinusoidal image. The 2-cells of the complex are formed by a minimum a maximum and two saddles. Minima and maxima are always adjacent to saddles.

All other image complexes had their discrete Morse complexes computed using the algorithms in Section 4.1. Table 4.2 presents the number of critical points of each type in the resulting discrete Morse complexes.

The complexes are employed in two important computations over Morse complexes: the persistence of critical points and hierarchical decompositions of the complexes through cancellation of low persistent topological features. These topological features are 0-cycles and 1-cycles which, in a 2-dimensional space, related to connected components and holes in the level sets of the images. The number of cycles in the level sets are along with their persistence give the called persistent Betti numbers. The persistent Betti numbers were computed by using the algorithm for pairing cells described by Zomorodian [100]. The

algorithm requires a total ordering of the critical points in the Morse complex which was obtained by assigning the grayscale value of each critical cell to the corresponding point in the Morse complex. Cells of lower dimension precede cells of higher dimension if they have a same grayvalue. The persistence is defined as the difference of grayvalues between the paired cells (creator and destroyer).

The image in Figure 4.15b depicts in different colors the persistence of each critical point in the discrete Morse complex of the sinusoidal image (Figure 4.15a). The color scale varies from blue to red, representing low persistence and high persistence. Intermediate values are shown in yellow and green intensities. The critical points of higher persistence are in the center of the image and the persistence circularly decays from these points to points in the boundaries of the image. This effect is due to the exponential function multiplied to the sinusoidal function. The 0-cycles and 1-cycles of the image are summarized in the graphs of Figure 4.16. The cycles are sorted by persistence and it can be noticed that the persistence increases following an exponential behavior, as expected.

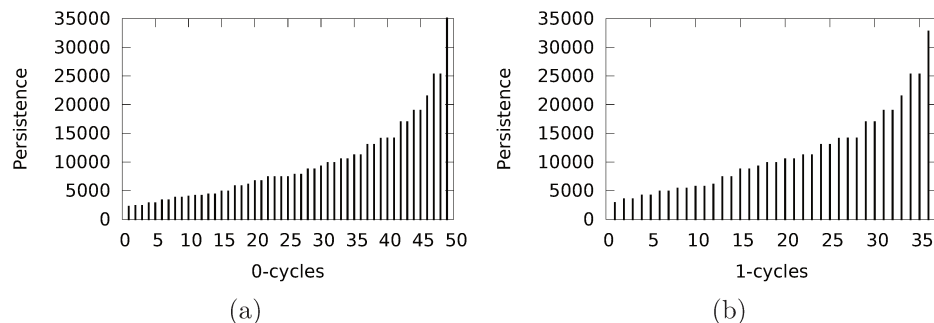


Figure 4.16: Graphs showing for each 0- and 1-cycle their respective persistence in the sinusoidal image. The sorted persistence of the cycles grows exponentially, except for the last 0-cycle which has infinite persistence.

The graphs of cycles versus persistence for the real images are shown in Figure 4.17. The persistence axis is shown in logarithmic scale so that the values can be better analyzed. These graphs show a common characteristic: a small amount of cycles with high persistence and a great amount of cycles with very low persistence. These very low persistent cycles may be due to topological noise of the data. A important task is then the removal of such topological noise, as explored in [6] and [26]. Cleaning the topological noise may be interesting for better understanding the function or phenomena being studied. The data-structure of our complex allows easy and fast manipulation of the complex for simplifications. We implemented the operation as described in [26].

The graphs of Figure 4.18 present an alternative view for the distribution of cycles according to their persistence and level sets of the Death Valley image. The horizontal

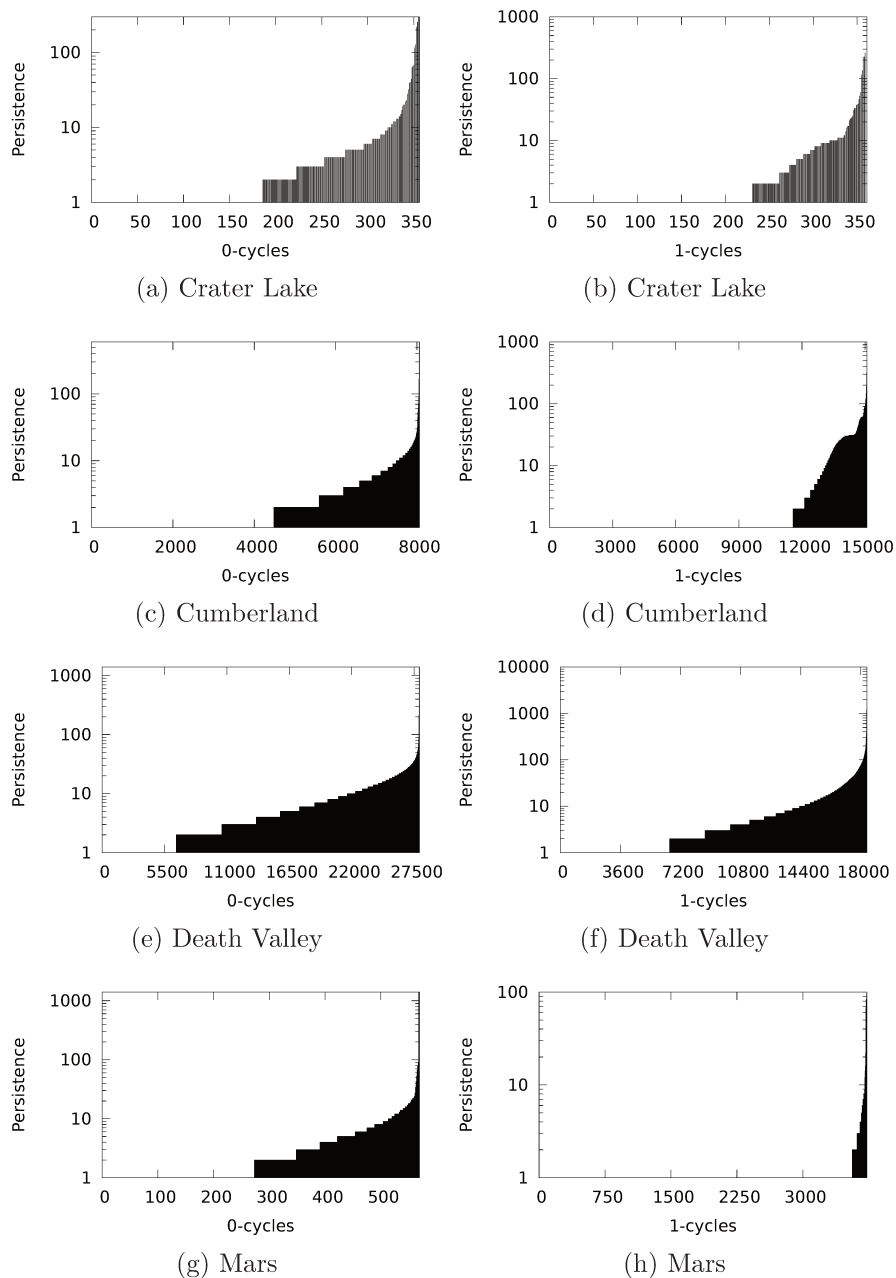


Figure 4.17: Persistence of cycles for real elevation terrain images. The most persistent cycles are present in the right portion of the graphics. It is possible to notice that the majority of cycles have a very small persistence value. That can be due to noise or non-relevant topological features. For instance, the number of cycles with persistence less than 10 in all graphs represents the greatest part of the features, which suggests that the persistence can be used for compression or noise removal.

axis are thresholds of the elevation values and consequently represent the level sets of the image function. The lines show the number of cycles with persistence greater than or equal to some specific value for all level sets.

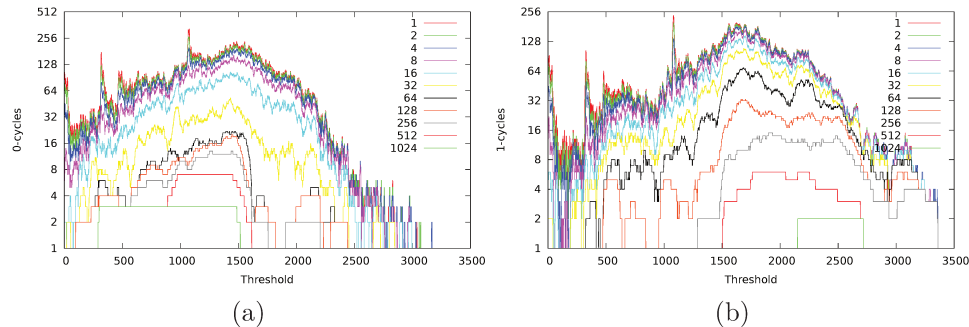


Figure 4.18: Number of cycles for different level set of the image Death Valley. Each line shows the number of cycles with persistence greater than or equal to a specific value of persistence (1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024).

It can be noticed that noisy topological features are mostly concentrated in low persistent points. By suppressing low persistent features the discrete Morse complex becomes cleaner and a lot less denser. This effect is reflected in the hierarchy of simplified discrete Morse complexes shown in Figure 4.19.

The persistence values range from 0 to 1225 (except for one infinite persistent 0-cycle). The Death Valley is a mountain region with a few valleys. It has several fine details, as can be seen from the amount of critical cells in the digital elevation model (DEM) of Figure 4.19a. The most important details are present in the mountains, even though there are also some details in the valleys. Some of those details might have been produced due to noise in the acquisition process. For example, the valley in the left portion of the image has a dense concentration of critical cells, as can be noticed in Figure 4.19b, which may probably be due to noise or non-relevant features. As the hierarchy progresses, these points will be removed. The hierarchies of Figures 4.19i and 4.19j show complexes where such features have been removed, the valleys become smoother, whereas details of the mountains have been preserved. Further steps in the hierarchy could have been considered, however, in detriment of topological feature preservation.

A hierarchy for the Crater Lake image is shown in Figure 4.20. As the simplification progresses, topological features of low prominence are removed from the image function. In the initial complexes, the island in the middle of the lake does not stand out since there are many low persistence features around it. These features are completely removed by thresholding critical cells with persistence less than 32. The number of cells necessary to capture the topology of the image is again massively reduced.



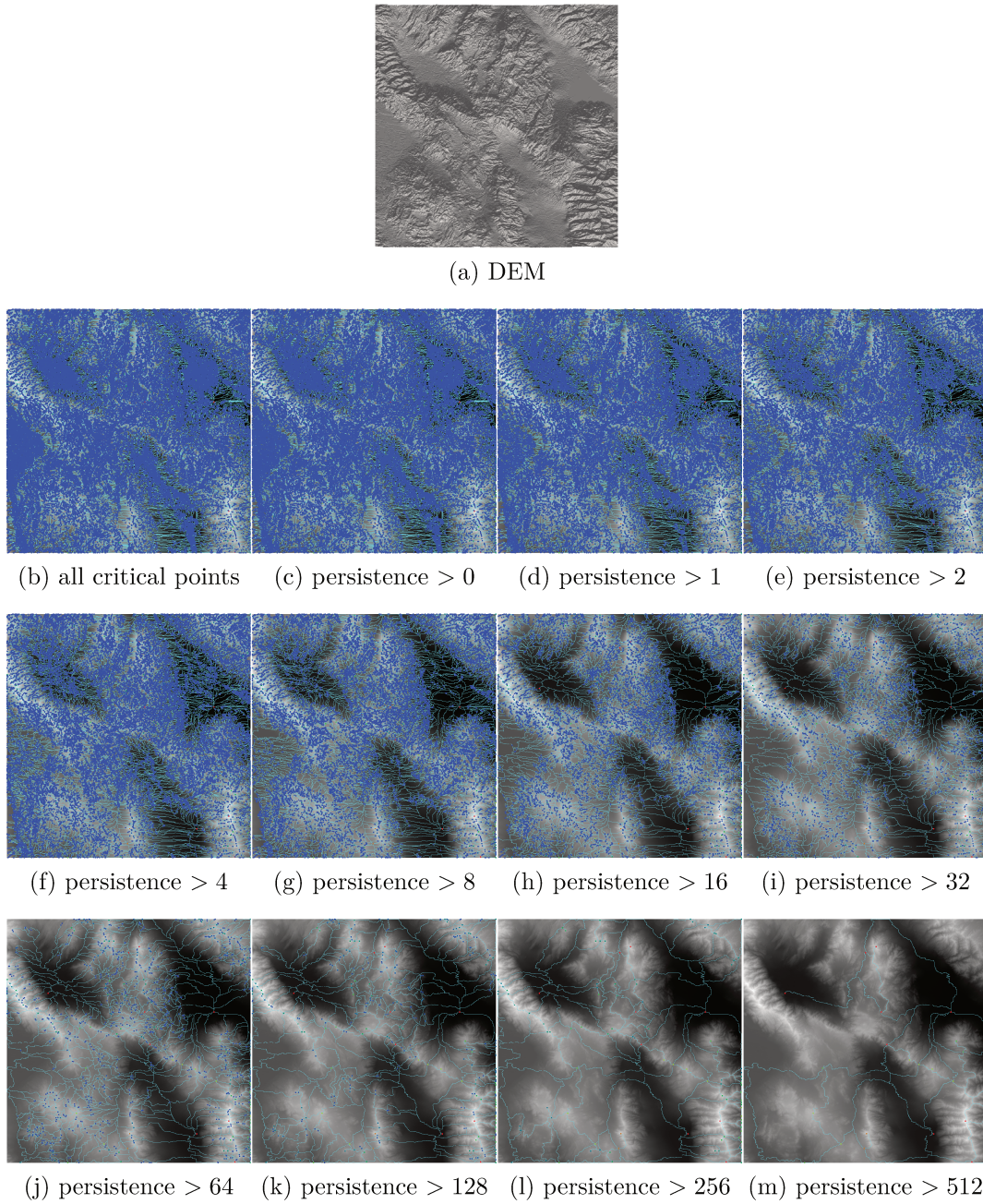


Figure 4.19: Hierarchy of a discrete Morse complex for different persistence values. Only points with persistence larger than a persistence threshold are maintained in the complex. The complex is greatly simplified and topological noise is removed. As the hierarchy progresses, fine details and noise are removed. Close to a persistence threshold of 32, the valleys become almost free of noise whereas relevant features in the mountains are preserved in the complex.



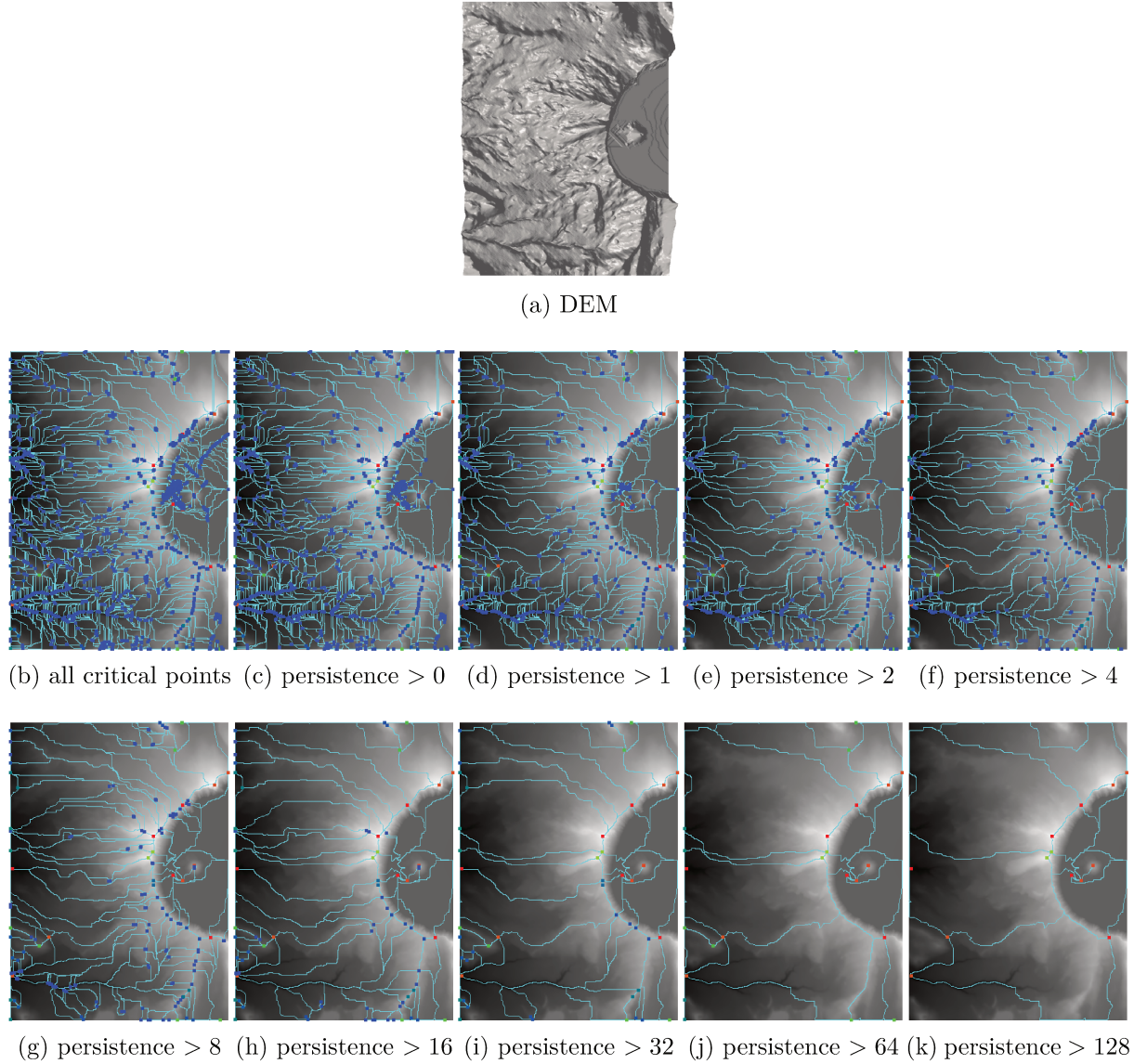


Figure 4.20: Hierarchy for the Crater Lake image discrete Morse complex. The island in the middle of the lake, as well as other important features, stands out from noise in the complex as the hierarchy progresses. The persistence values range from 0 to 360 (except for one infinite persistent 0-cycle).

Furthermore, we have computed the hierarchy for a face image, as shown in Figure 4.21. Initially, there is a high concentration of feature areas with non-relevant details. The hierarchy shows that such details are removed as the levels in hierarchy are traversed, whereas features located in the eyes, nose and mouth stand out. This example shows how the simplification of the complexes can be used to capture significant topological information of images and suggests that it can be useful for image classification. The

image is taken from the AT&T Laboratories Cambridge [72].

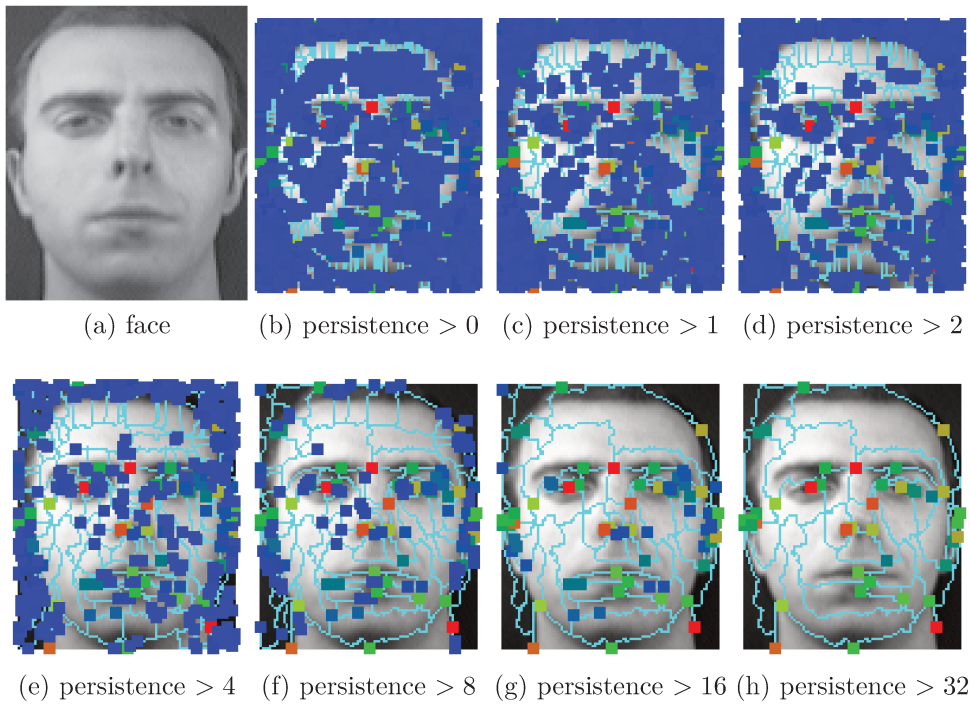


Figure 4.21: A hierarchy of discrete Morse complexes computed for a face image. When considering coarser levels of the hierarchy, features in the eyes, nose and mouth stand out from other features. The persistence ranges from 0 to 138.

## Chapter 5

# Nighborhood-of-Interest Points Using the Morse Complex

The correspondence or matching of image interest points is a basic step in computer vision that is used to find corresponding locations in different images for tasks such as image stitching, image registration, scene reconstruction, object detection and recognition [7, 48, 58, 62, 70, 85, 98]. A well explored advantage of the correspondence of interest points is that it allows matching in the presence of occlusions and changes in scale and orientation [84]. Furthermore, more reliable matching of images can be computed [21] and might be used to compute denser correspondences [55]. However, the approach may fail [71] depending on the image nature, acquisition method, noise corruption, and transformations between images.

We investigate in this chapter the correspondence of interest points, which is part of a project in collaboration with the Computer Science Department at the University of California, Davis [83]. The method is intended to be used for establishing a denser set of correspondences to further construct 3D models and merge point clouds produced from images acquired at different time steps (video frames).

The considered images are challenging and the application has some requirements to be satisfied, whenever possible:

1. the matches should be spread all over the images;
2. it should be possible to measure the confidence of a match being correct.

The basic stages for corresponding interest points between images include interest point detection, description, and matching. Interest points are usually considered as independent elements described by some limited local information, basically, the appearance of patches of pixels surrounding the point location [84]. The limited local information clearly is not able to discriminate between interest points in some cases. The difficulty

arises from patches related to two interest points not having enough dissimilarity between them. Examples of such cases occur in smooth regions, repeated patterns and symmetries. Therefore, images with different regions of high similarity often incur in problems of discrimination due to the limited information to compute descriptors.

Besides such difficulties, many other may occur. For example, for the tested underwater images, which are the main subject of study here, besides the morphological nature of the regions in the images, which may create highly similar structures, noise may be present due to particles in the water and the acquisition may produce distortions and illumination differences.

Commonly, the correspondences of images are computed by pairwise comparison of their interest point local descriptors, such as performed by the  $k$ -nearest neighbor algorithm [62, 84]. This approach is computationally efficient and suitable for real-time applications or for problems that deal with massive amounts of data. However, the difficulties arising from point detection and description are propagated and not properly handled at the matching stage. Figure 5.1 shows some examples of incorrectly matched points.

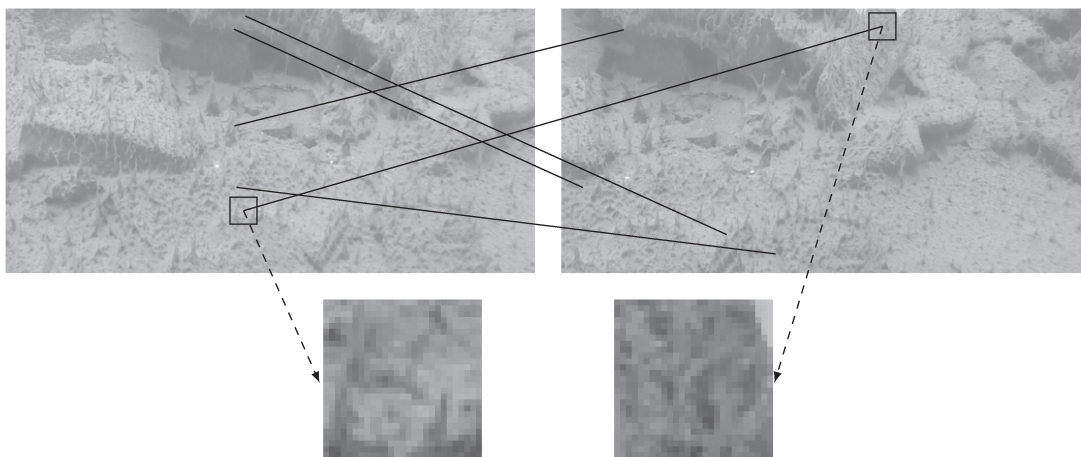


Figure 5.1: Examples of incorrect matches computed due to regions of high similarity. The descriptor by itself is not able to discriminate between some regions. The zoomed regions show the pixel level texture similarities that produce close descriptors and consequently difficulties for the correspondence of interest points.

Cases of incorrect correspondences are very common and there are some different levels in which it is possible to consider ways of eliminating them. At the early detection step, it is possible to threshold the detected interest points by using a measure of importance or of how salient the interest point is. In such a manner, it is possible to avoid detecting interest points in highly homogeneous regions. At the matching level, approaches such as the one used in the Scale-Invariant Feature Transform (SIFT), the SIFT-ratio [59],

perform a comparison of similarity between the  $k$ -closer matches of an interest point (usually  $k$  is 2) and, therefore, when they are too similar in terms of descriptors, the matches are removed. At another level, approaches such as RANdom SAmple Consensus (RANSAC) [29] use a model fitting to find a transformation such that outliers are removed from the correspondence set.

It is clear that all these approaches reduce the number of correspondences obtained without trying to correct the matches. In fact, the initial number of correspondences invariantly suffers a drastic reduction. Figure 5.2 shows an example of correspondences obtained through the application of interest point thresholding and SIFT-ratio. A set of good matches is acquired and can be useful in many applications, such as registration. However, the correspondences are sparse and not in agreement with the requirements of spread matches stated in our problem.

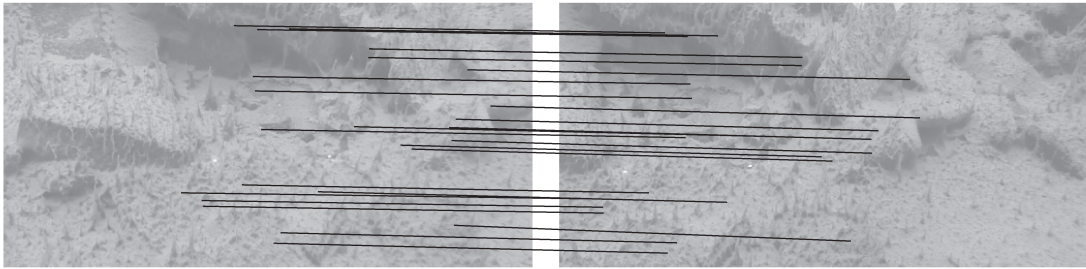


Figure 5.2: Examples of output matches obtained through SIFT method. Incorrect correspondences are filtered out, however, the number of correspondences is drastically reduced and many regions do not contain any paired points.

A region growing approach is presented in [13] for correspondence verification. The method outperforms the selection of SIFT. However, it is still not a corrective method. A small number of correspondences may not be enough for obtaining denser matches for some images, such as the ones considered in this chapter.

Alternatively, the structural relations between interest points can be introduced to obtain more global information. Such relations are commonly modeled by using graphs and allow the exploration of structural arrangements to better discriminate regions in images. Successful applications to find or discriminate sparse number of interest points [14, 42, 45, 94] have been reported. However, finding correspondences within dense sets of points using graphs is still a challenging problem. Algorithms for this purpose are computationally expensive and sensitive to noise in the data [49, 74, 95]. Much of the efforts to solve correspondences using graphs deal with complete or very large graphs [8, 20, 25, 34, 74, 80], which makes the limitations of graph matching even more severe.

We present an approach based on the topology of functions, given by discrete Morse



complexes, to defining locally meaningful connectivity of interest points. We have devised and tested the method for obtaining correspondences in images and to demonstrate its contributions:

- a general neighborhood relation which can be adapted to different applications.
- a manner of avoiding incorrect correspondences in high similarity images.
- an approach to evaluating the confidence of a correspondence being correct.

The matching algorithm developed is conceptually similar to pixel-based seed-growing methods such as [13, 15, 43, 55] that have been proved useful. These are, however, mostly post-processing steps used after the matching of interest points. Our algorithm differs in its objective as it is used to compute more correct correspondences through interest point-based growing.

The remaining of this chapter is organized as follows. The proposed topological neighborhood is formulated in Section 5.1. In Section 5.2, the matching algorithm is developed. Experimental results are presented and discussed in Section 5.3.

## 5.1 Local Morse Context

In this section, we define image interest points in terms of critical Morse cells and, mainly, introduce the Local Morse Context (LMC) operator to obtain the local neighborhood relation between these interest points.

### 5.1.1 Morse Complex and Critical Cells

As described in Chapter 3, minima and maxima are Morse critical cells. In computer vision, such features are usually obtained from a derivative of the input image function [63]. In the same manner, we use a derivative of the input image, the Laplacian of the Gaussian (LoG) [84], modeled as an image cell complex (see Section 2.3), to construct the discrete Morse complex (Chapter 4). Figure 5.3 illustrates the process. Given the input image, the LoG of it is computed and used to obtain the discrete Morse complex.

Critical cells of maxima and minima are analogous to the maxima and minima points usually detected as interest points of images. For that reason and to be in accordance with the developed method, interest points will denote Morse critical cells and we will use both terms interchangeably in the remaining of the text. Critical Morse cells can also be saddles, however, they are not stable to perturbations in the input function and are not used as interest points. Therefore, given the discrete Morse complex  $M$ , of an image,

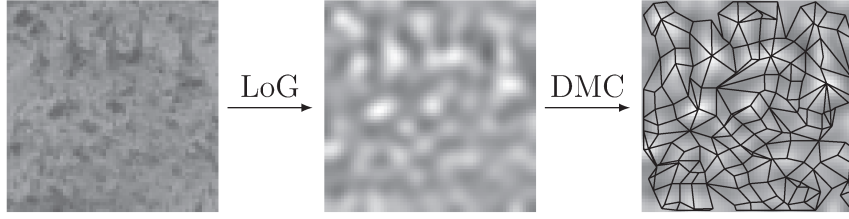


Figure 5.3: Example of LoG and DMC of an image.

with  $n$  critical cells  $\alpha_k^p$ ,  $k = 1, \dots, n$ , we define the interest points to be the set of minima and maxima

$$C = \{\alpha_k^p \mid \alpha_k^p \in M \text{ with } p = \{0, 2\}, k = 1, \dots, n\}. \quad (5.1)$$

Usually, an image interest point is a pixel for which the neighborhood relation is defined over its closest pixels in the discrete lattice (such as a 4- or 8-neighborhood). Such relations do not allow an easy determination of which interest points are close to each other. By using the discrete Morse complex, it is possible to define the neighborhood as a relation over closest interest points. The computation of such a relation is the topic of the next section.

### 5.1.2 Local Morse Context

The Local Morse Context (LMC) is a relation over discrete Morse complexes for acquiring information regarding the neighborhood of interest points. In order to understand the computation of the LMC, one should recall the definitions of star, closure and link presented in Section 2.4.

The main concepts of the proposed method are presented next. Given an interest point  $\alpha_k \in C$  (see Section 5.1.1), the star of order  $i$  (or  $i$ -th iterated star) of  $\alpha_k$ ,  $\text{St}_i(\alpha_k)$ , is a recursion

$$\text{St}_i(\alpha_k) = \begin{cases} \alpha_k & \text{if } i = 0 \\ \text{St}(\alpha_k) & \text{if } i = 1 \\ \text{St}(\text{Cl}(\text{St}_{i-1}(\alpha_k))) & \text{otherwise.} \end{cases} \quad (5.2)$$

The  $i$ -th order local Morse link (LML) of  $\alpha_k$  is defined to be the link of the  $i$ -th iterated star of  $\alpha_k$ ,

$$\text{LML}_i(\alpha_k) = \{\tau \mid \tau \in \text{Lk}(\text{St}_{i-1}(\alpha_k))\}. \quad (5.3)$$

The LML is defined over the  $(i - 1)$ -th closure of the star since the link itself is defined over the star of a subcomplex (Definition 2.1), therefore, the  $i$ -th star is implicitly applied by the link.

The  $i$ -th order local Morse context of a critical cell  $\alpha_k$  is defined as the set of minimum and maximum critical cells in the LMLs up to order  $i$ ,

$$\text{LMC}_i(\alpha_k) = \{\tau^p \mid \tau^p \in \text{LML}_j(\alpha_k), p = 0, 2, j = 1, \dots, i\}. \quad (5.4)$$

Figure 5.4 illustrates the computation of a second order local Morse context ( $\text{LMC}_2$ ) given a particular critical cell  $\alpha$  in  $M$ . The second column shows the iterated stars of order 0 and 1, of  $\alpha$ , while the third column presents their links, namely  $\text{LML}_1$  and  $\text{LML}_2$ . The selection of maxima and minima is performed next, producing the sets shown in the fourth column. Finally, the union of the two sets of the fourth column produces the  $\text{LMC}_2$ , as can be seen in the rightmost image. Notice that the LMC is a nested operator, that is,

$$\text{LMC}_0(\alpha_k) \subseteq \text{LMC}_1(\alpha_k) \subseteq \dots \subseteq \text{LMC}_i(\alpha_k). \quad (5.5)$$

Figure 5.4 shows the  $\text{LMC}_1$  in the lighter region inside the darker rectangle related to the  $\text{LMC}_2$ .

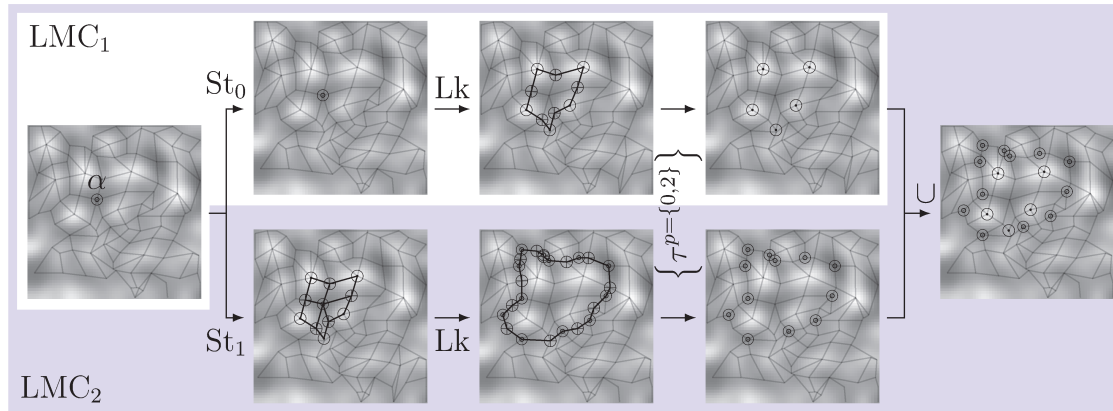


Figure 5.4: Example of computing the  $\text{LMC}_2$  of a maximum critical cell  $\alpha$ . Increasing the order augments the number of critical points in the set and expands the neighborhood.

The order of the LMC controls the extent of the neighborhood to be used. The higher is the order, the larger is the number of connected points and the region covered by the neighborhood. Therefore, more global information is captured as the order increases. Figure 5.4 shows how the LMCs of order 1 (first row of fourth column) and 2 (fifth column), computed for a critical cell  $\alpha$ , influence in the region covered around  $\alpha$ .

The neighborhood given by the LMC makes it possible to explore the local structural information to characterize a pattern and/or increase the information of a local descriptor. Since the LMC is computed over a Morse complex, the neighborhood is based on the topology of a function and, therefore, it is expected to be independent of geometrical transformations that may be applied to the function.



## 5.2 Image Matching Using the LMC

This section presents how the LMC can be used to correspond points in pairs of images. Initially, we show a method for corresponding points within two LMCs, which is the core step for obtaining the correspondences between images. The correspondence of LMCs is used to guide the matching and, as a consequence, it helps to avoid incorrect matches due to similarities (as explained in the introduction).

### 5.2.1 Matching of LMCs

Given a critical cell  $\alpha_k$  of a Morse complex  $M_1$  and a critical cell  $\beta_l$  of a Morse complex  $M_2$ , let their LMCs be  $\text{LMC}_i(\alpha_k)$  and  $\text{LMC}_i(\beta_l)$ , as illustrated in Figure 5.6. Clearly  $\alpha_k$  and  $\beta_l$  are corresponding points as can be noticed by the high similarity of the regions depicted in Figures 5.5a and 5.5b. It can also be noticed that, as expected, the LMCs are very similar, suggesting that the correspondence of LMCs can be formulated as the correspondence of structured data. The task becomes a graph matching problem which is well known and have various proposed solutions [20, 34, 89].

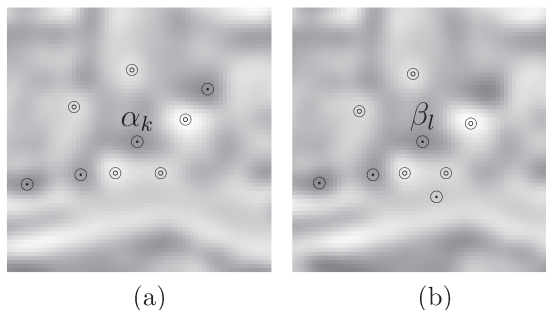


Figure 5.5: Example of a pattern between two related critical cells. Missing or added critical cells are likely to occur.

The standard solution of graph matching is carried out by means of graph isomorphisms or subgraph isomorphism if the graphs have different sizes. However, these methods can only find a solution if there is a perfect match [88]. In real world applications, finding isomorphisms is unfeasible because identical structures are very unlikely to occur due to noise and distortions present in the data.

There are many factors that may introduce noise and inaccuracy to image data. Such issues are also a problem between LMCs. The LMCs of two corresponding regions, in different images, should not be expected to be the same. The number of critical cells between LMCs will probably be different. Taking one of the LMCs as a base for comparison, some of its critical cell may be missing in the second LMC or some cells that do

not appear in it could be added to the second LMC. In Figure 5.5, one minimum point is missing in the second set and one maximum point appears in the second set, but not in the first.

Noisy or inconsistent cases are also a concern of various applications and extensively studied [8, 20, 34, 74, 79, 87]. Usually, the matching constraints are relaxed in search of non-exact correspondences. The problem is known as inexact graph matching [20, 34]. We solve our problem with a method based on eigendecomposition approaches [78, 80, 89], specifically the one proposed by Scott and Longuet-Higgins [78], that provides an elegant solution with one-to-one correspondences and no explicit iterations.

Suppose  $\text{LMC}_i(\alpha_k)$  has  $m$  interest points,  $\sigma_r$ ,  $r = 1, \dots, m$ ; and  $\text{LMC}_i(\beta_l)$  has  $n$  interest points  $\tau_s$ ,  $s = 1, \dots, n$ . Let  $d_{rs} = \text{dist}(\text{desc}(\sigma_r), \text{desc}(\tau_s))$  be the Euclidean distance between the descriptors ( $\text{desc}$ ) of  $\sigma_r$  and  $\tau_s$ . Individually, each feature can be described by some local measure such as in HOG [23], HSC [77], SURF [3] or SIFT [58].

The method computes an  $m \times n$  matrix  $G$  with pairwise affinities

$$G_{rs} = \exp\left(\frac{-d_{rs}^2}{2t^2}\right) \quad (5.6)$$

of interest points, where the parameter  $t$  controls the degree of proximity between descriptors and it is suggested to be 4 in [80]. Perfect matches ( $d_{rs} = 0$ ) have affinity value 1. The farther the distance between descriptors, the more the affinity approaches 0.

A singular value decomposition (SVD) is performed to factorize  $G$  as

$$G = UDV^T \quad (5.7)$$

where  $U$  is an  $m \times m$  orthogonal matrix,  $D$  is a  $m \times n$  diagonal matrix and  $V^T$  is the transpose of an  $n \times n$  orthogonal matrix  $V$ . Every element of the principal diagonal of  $D$  is replaced by 1 to create the matrix  $E$ . The association matrix is computed as

$$P = UEV^T \quad (5.8)$$

such that the rows of  $P$  index the interest points in  $\text{LMC}_i(\alpha_k)$  and the columns index the interest points in  $\text{LMC}_i(\beta_l)$ .

If  $P_{rs}$  is the largest element both in row  $r$  and column  $s$ , then a strong correspondence will be achieved. However, if  $P_{rs}$  is the largest element in its column but not in its row, or, similarly, in row but not in column, then multiple points compete for the match and the correspondence is weak. Correspondences in the LMCs are obtained by retrieving the strong correspondences of  $P$ . The numbers in Figures 5.6a and 5.6b show the correspondence between the two LMCs in our example.

The SVD computation can be performed in  $O(mn^2)$  [86]. Therefore, the choice of low order LMCs is important if fast computations are required.

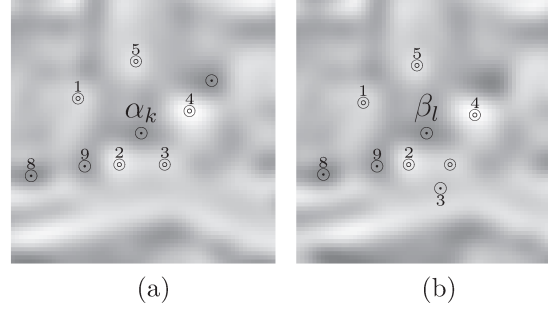


Figure 5.6: Corresponding points between LMCs, the correspondences are represented with numbers.

### 5.2.2 Image Match

Let  $C_I$  and  $C_J$  be the sets of interest points of two images  $I$  and  $J$  and also let  $S$  be a set with some correctly matched pairs  $(\alpha_k, \beta_l)$ , called seeds, such that  $\alpha_k \in C_I$  and  $\beta_l \in C_J$ . Assume  $S$  is given (we show one way to compute it in Section 5.3). These sets are the inputs for Algorithm 5.1, which computes all the correspondences from  $C_I$  to  $C_J$  starting from the initial matches in  $S$ . The key idea is to increase the number of matchings at each iteration from matches already computed.

---

**Algorithm 5.1:** LMCImageMatching

---

**Input:** Set of interest point  $C_I$  from image  $I$ ; set of interest point  $C_J$  from image  $J$ , set of seeds  $S$ .

**Output:** Putative matches.

```

1  $Q \leftarrow S$ 
2 while  $Q$  is not empty do
3   extract match  $(\alpha_k, \beta_l)$  from  $Q$ 
4   compute correspondences  $(\sigma_r, \tau_s)$  for  $\sigma_r \in \text{LMC}(\alpha)$  and  $\tau_s \in \text{LMC}(\beta)$ 
5   for each  $(\sigma_r, \tau_s)$  do
6     if  $\text{dist}(\text{desc}(\sigma_r), \text{desc}(\tau_s)) < \text{dist}(\text{desc}(\sigma_r), \text{desc}(\text{pair}(\sigma_r)))$  then
7       if  $(\sigma_r, \text{pair}(\sigma_r)) \in Q$  then
8         remove  $(\sigma_r, \text{pair}(\sigma_r))$  from  $Q$ 
9       end
10       $\text{pair}(\sigma_r) \leftarrow \tau_s$ 
11       $Q \leftarrow Q \cup (\sigma_r, \tau_s)$ 
12    end
13  end
14 end

```

---

An auxiliary queue  $Q$  is initialized in line 1 of the algorithm with the seeds in  $S$ . In the

example of Figure 5.7a, the set of seeds starts with three correspondences. Line 3 removes one putative match  $(\alpha_k, \beta_l)$  from  $Q$  and line 4 computes the correspondences of the interest points in their LMCs (see matching of LMCs in Section 5.2.1). The correspondences in the LMCs provide new matches  $(\sigma_r, \tau_s)$  in the neighborhoods of  $\alpha_k$  and  $\beta_l$ . In the loop of line 5, the new matches are checked against possible previous matches of the same interest points. The loop goes through all  $(\sigma_r, \tau_s)$  obtained checking whether the distance (`dist`) between the descriptors (`desc`) of  $\sigma_r$  and  $\tau_s$  is closer than the descriptors of  $\sigma_r$  and a possible match previously found for it (line 6). The previous match is retrieved by the function `pair`. If the descriptors of  $\sigma_r$  and  $\tau_s$  are closer, then `pair`( $\sigma_r$ ) will be set to  $\tau_s$  and the putative match  $(\sigma_r, \tau_s)$  will be inserted into the queue (lines 10 and 11). In such a way, new matches will be computed from the LMCs of  $\sigma$  and  $\tau$  in a subsequent iteration of the outer loop of line 2. The condition in line 7 tests if a previous match of  $\sigma$  is in the queue and, if so, it is removed in line 8.

From the set of seeds, the number of correspondences grows until all interest points are matched, that is, until the queue  $Q$  becomes empty. The matching of LMCs, performed in line 4, enforces that interest points in a neighborhood of  $I$  are matched in the corresponding neighborhood of  $J$ . Therefore, the LMC guides the acquisition of new matches in Algorithm 5.1.

Figures 5.7b, 5.7c and 5.7d depict different iterations in the growing process: initial, intermediary and final. The initial steps of Figure 5.7b show that the matches follow local restrictions. In such a manner, it is possible to avoid many incorrect matches as when using only the information of descriptors to perform the correspondences. Figure 5.7c shows that the growing fronts from different seeds meet at some point. If the seeds were correctly initiated, and consequently the matches grew correctly, then the matches at the boundary of the fronts would agree and the growing process would stop due to the test of line 6. However, if the matches do not agree, one front will take over another front, correcting the previous matches. That can happen if one of the seeds is not a true match. In such a case, the matches will be corrected by the front of the correct seed. That means that the set  $S$  does not necessarily need to contain only correct matches, but at least one. When the all fronts meet and no better matches are found, the process ends as shown in Figure 5.7d.

### 5.3 Experimental Results

In this section, we present experimental results that show the effectiveness of the LMC to improve the matching of interest points and also to compute a score that characterizes correct and incorrect matches. The latter result is applied in the selection of seeds for Algorithm 5.1 and it is part of the setup of parameters for the algorithm.

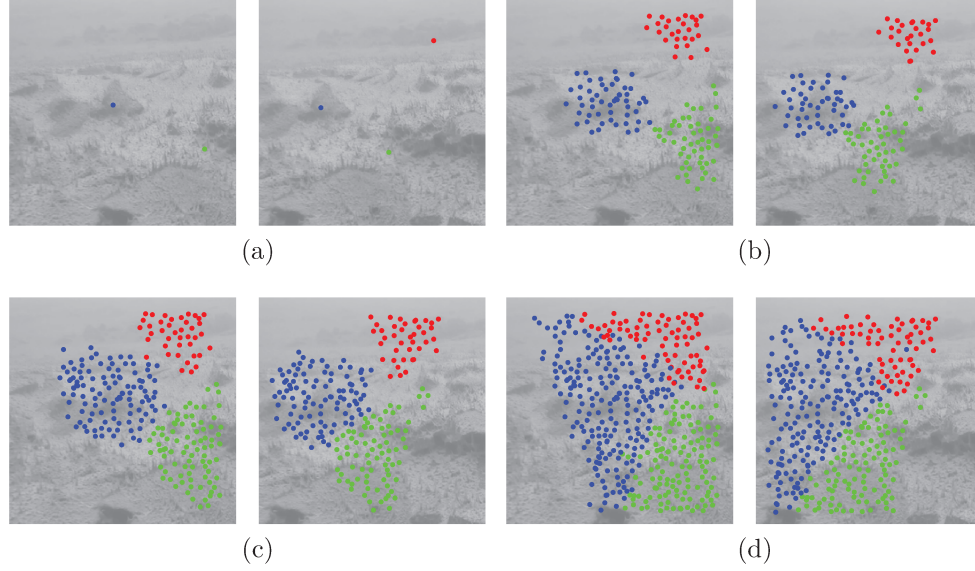


Figure 5.7: Correspondences growing from initial seed set (a). The LMC locally guides the matches (b-c) until all interest points have correspondences established.

### 5.3.1 Datasets and Ground Truth

Two datasets were used to evaluate our method. The first dataset contains synthetic stereo pairs at three different baseline separations and ground truth disparity maps. The dataset is made available by the University of Alberta and was proposed in the paper [69]. The second dataset consists of pairs of underwater images taken from the bottom of lakes in Antarctica and they are part of a project between the Computer Science and Geology Departments of the University of California, Davis [83], with whom we have been collaborating.

The first dataset is used as a reference set of images since it is known in the literature to evaluate methods on stereo image pairs and also because the disparity maps make it possible to directly evaluate the correctness of obtained matches. The set is interesting to test the requirements of our method stated in the beginning of this chapter. The synthetic images are built with different types of textures that contain repeated patterns and similar regions.

The second dataset, due to several reasons, is more challenging in terms of matching interest points. These are non-calibrated images depicting ridged and peaked morphologies found on the bottom of lakes that create complex structural formations. In many cases, similar structures can be found all over the images, making it difficult for local descriptors to capture the differences between some regions. Besides the morphological nature of the regions in the images, many other difficulties arise from the acquisition of

underwater images, such as distortions, illumination differences, and noise produced by particles in the water. A subset of images from both datasets is shown in Figure 5.8.

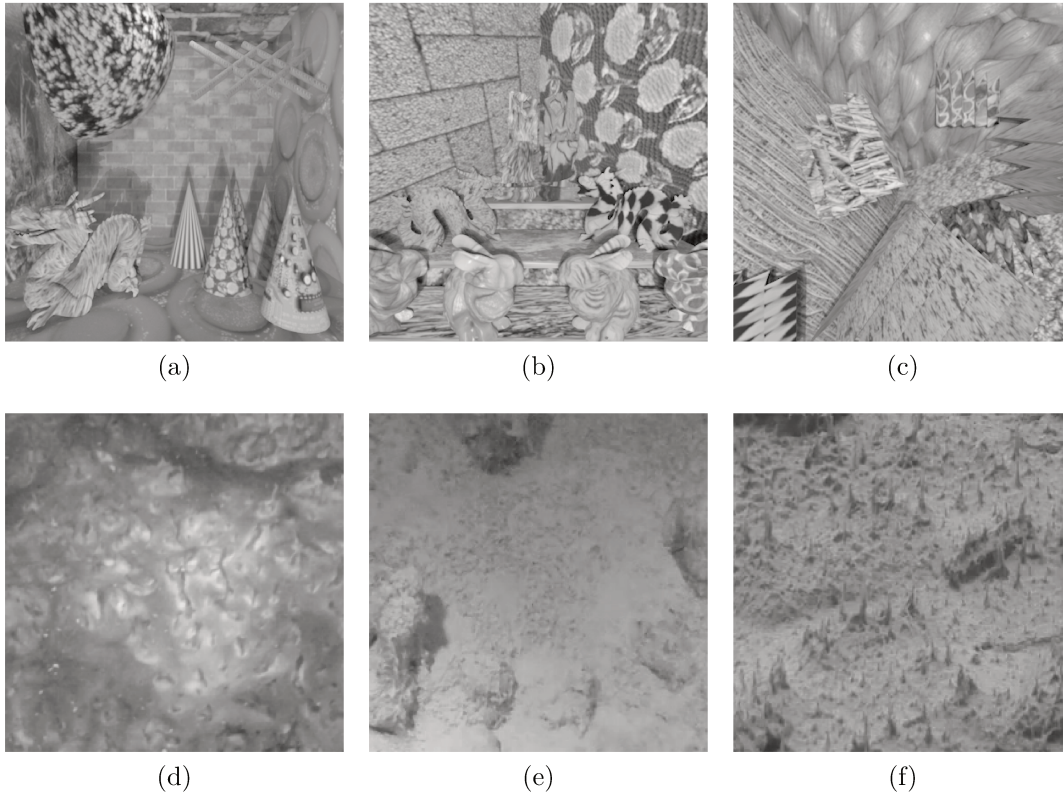


Figure 5.8: Examples of synthetic (a)-(c) and underwater images (d)-(f) used in our experiments.

Unlike the synthetic pairs, which have an available ground truth, the pairs of underwater images do not have a known ground truth characterizing matches in the images. Therefore, based on [62], which addresses a similar problem, we estimate a ground truth using homography matrices [39]. Given a pair of images to be matched, a set of visually confirmed matches is chosen and used to compute a homography matrix  $H$ . In such a way, given an interest point  $p$  in one image, it is possible to estimate its corresponding point  $q$  in the second image by computing the homographic transformation of  $p$ , that is,  $q = Hp$ .

The tested datasets are composed of 90 pair of synthetic images and 21 pairs of underwater images. Both sets were subdivided into subsets of validation images (9 synthetic pairs and 6 underwater pairs), used to set parameters; and of test images (81 synthetic and 16 underwater images) pairs, used to measure the quality of the matchings. The Morse complexes and interest points were obtained following the description in Section 5.1.1.



### 5.3.2 Evaluation Metrics

In order to evaluate our results, we use metrics based on the number of correct and false matches, namely recall and 1-precision, since they are widely employed for similar evaluations [44, 62].

Given the Morse interest points for all of the images in a dataset, two interest points  $\alpha$  and  $\beta$  are considered a match if the distance between their descriptors (Chi-squared distance in our experiments) is below a threshold  $t$ . A match is true positive if the interest points correspond to the same physical location (as determined by a ground truth), whereas a match is false (positive) if the matched interest points correspond to different physical locations.

The correct correspondence of physical locations is determined by the overlap error [62]. Suppose  $A$  and  $B$  are the regions around  $\alpha$  and  $\beta$ , respectively, for which the descriptors are computed. The overlap of  $A$  and  $B$  is defined by the ratio of the intersection and union of the regions  $\epsilon_S = 1 - (A \cap TB)/(A \cup TB)$  under a transformation  $T$ . The transformation  $T$  is the disparity for the synthetic image pairs and the homography matrix for the underwater image pairs. As in [62], we assume that a match is correct if  $\epsilon_S < 0.5$ , so that the area covered by two corresponding regions is less than 50 percent of the region union.

Recall and 1-precision are defined as [44]:

$$\text{recall} = \frac{\text{number of true positives}}{\text{total number of positives}} \quad (5.9)$$

and

$$\text{1-precision} = \frac{\text{number of false positives}}{\text{total number of matches (correct or false)}}. \quad (5.10)$$

The total number of positives for the given dataset is computed by comparing the overlap error of all interest points. The recall versus 1-precision graphs are obtained by varying the value of  $t$ .

### 5.3.3 Method Setup

In this section, we discuss how to tune the following parameters of the matching algorithm: (i) order of the LMC used to grow matches; (ii) order of the LMC to score matches and select seeds; (iii) number of seeds.

We used the histogram of oriented gradients (HOG) [23], a well known descriptor, to describe each interest point of the images. However, the proposed method is not attached to a specific descriptor.

### Order of LMC to grow matches

This test evaluates the choice of the LMC order to grow the number of matches in Algorithm 5.1. The order is chosen to maximize the confidence of the resulting matches. For these tests, the seeds for Algorithm 5.1 were visually chosen so that the algorithm grows from true matches.

The results for the four orders studied are shown in Figure 5.9, both for synthetic and underwater datasets. The LMC of order 1 achieves the best results for this test. This behavior can be expected since the local restriction to grow is loosened as the order increases, allowing farther points to be matched. At lower orders, the local restriction is stronger, meaning that points that are close to one region of the base image should correspond to close points in the related region of the pair image. In this case, the LMC works as a neighborhood analogous to 4-pixel, 8-pixel or 16-pixel neighborhood of an image, except that in the LMC the neighbors are interest points instead of pixels.

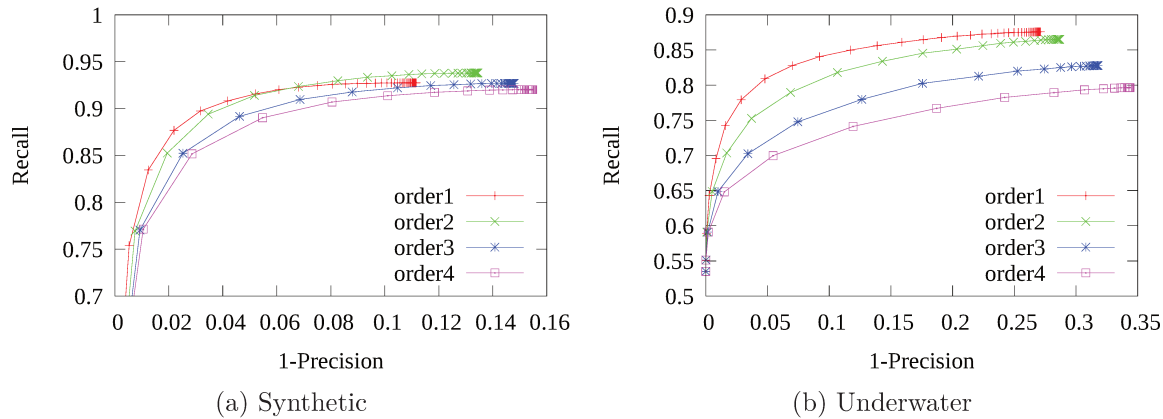


Figure 5.9: LMC order influence on growing the number of matches. The lower orders have a higher recall value. Therefore, the LMC-based algorithms performs better if the neighborhood considered to grow matches is smaller, the search for new correspondences in related regions between images is more restrict.

### Order of LMC to score matches

The second test we performed was to choose an order of the LMC to score matches and consequently estimate correct and incorrect matches. This is another application of the LMC we introduce and apply directly to choose seeds for our matching algorithm.

Given two matched interest points  $\alpha_k \in M_1$  and  $\beta_l \in M_2$ , we intend to estimate if they are a correct match. The distance between their local descriptors, as in SIFT-ratio, is a weak measure due to the limited information of local descriptors. The LMCs of two



matched points, however, have relational information that can be used to increase the knowledge of how reliable their matching is.

If  $\alpha_k$  and  $\beta_l$  are correctly matched, it is expected that their LMCs share similar structural relations, since the corresponding regions of the images are the same. Therefore, given a correspondence of the points in their LMCs, it is possible to obtain a measure of how similar the two patterns are. We propose to compute a matching score as

$$\text{score}(\lambda_r, \tau_s) = \sharp\{(\lambda_r, \tau_s) \mid Pr(\lambda_r) \approx Pr(\tau_s)\} \quad (5.11)$$

where  $(\lambda_r, \tau_s)$  are corresponding cells between LMCs that have values  $Pr(\lambda_r)$  and  $Pr(\tau_s)$  for some property on the structural patterns. The symbol  $\sharp$  denotes set cardinality so that the score counts the number of corresponding cells that agree with respect to property  $Pr$ . The property can be some invariant characteristic shared by the images in one application. Notice that the range of score values is dependent on the number of points in the LMCs.

Taking Figure 5.10 as an example, we need to find a property that holds for the pairs of images from our dataset. The images considered in this work present perspective transformations which do not preserve angles and ratios of lines linking interest points. Let the central circles be a matched pair of points  $(\alpha_k, \beta_l)$  to be scored. The other circles are interest points in the LMCs such that the numbers define their correspondences. Consider also the horizontal lines based on  $\alpha_k$  and  $\beta_l$ . Due to the nature of the stereo pairs, the points in the contexts may have significant horizontal displacements relative to the central point. However, the distances from corresponding points in the LMCs to the horizontal lines are expected to be similar (see vertical, solid lines in Figure 5.10).

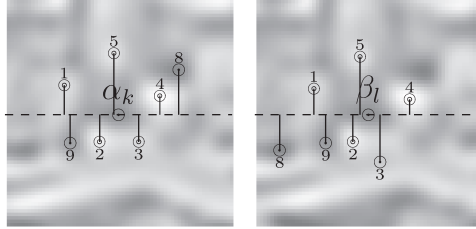


Figure 5.10: Geometrical property for score where corresponding points in a pair of LMCs have similar distances to the horizontal lines.

We use this fact to score matches. Given a match pair  $(\alpha_k, \beta_l)$ , we define its score as

$$\text{score}(\alpha_k, \beta_l) = \sharp\{(\sigma_r, \tau_s) \mid \text{disp}(\alpha_k, \sigma_r) - \text{disp}(\beta_l, \tau_s) \leq c\} \quad (5.12)$$

where  $\text{disp}(\alpha_k, \sigma_r)$  ( $\text{disp}(\beta_l, \tau_s)$ ) is the vertical difference from the points  $\sigma_r$  ( $\tau_s$ ) in the LMC of  $\alpha_k$  ( $\beta_l$ ) to the horizontal lines of  $\alpha_k$  ( $\beta_l$ ). The difference between displacements is

Table 5.1: Best scores for each order of LMC. For each order of LMC, the best score was chosen, that is, a correspondence scored with this or a greater value is mostly probable to be correct (fifth column) and the probability of finding such a correspondence is also high (fourth column). Higher orders perform better in this case. We find out that an order of 3 and score 11 are good choices for measuring the confidence of correspondences.

Dataset	Order	Score	Number of matches	True positives
			(from total of matches)	
Synthetic	1	2	0.81	0.94
	2	5	0.84	0.95
	3	9	0.85	0.96
	4	13	0.85	0.96
Underwater	1	2	0.44	0.75
	2	7	0.56	0.91
	3	11	0.59	0.91
	4	15	0.60	0.91

signed to differentiate between points lying below and above the horizontal lines. The constant  $c$  controls how much the vertical distance can differ between corresponding points. For our images, we have found that  $c = 3$  leads to satisfactory results. In our example, except from interest points identified with values 3 and 8, the displacements of five interest points are approximately the same. Therefore, the score for the matching  $(\alpha_k, \beta_l)$  equals 5 and it suggests that the matching is probably a correct one. This is the behavior studied in the following tests.

The validation images were matched using a 1-NN algorithm and all pairs of matched points were scored using orders 1 to 4 for the LMC. An adequate choice of the score is achieved when the number of correct matches becomes significantly larger than the number of incorrect matches and also the number of correct matches is not too small.

Therefore, for each order we chose the score which returned a considerable amount of matches with great probability of being a correct one. The results are summarized in Table 5.1, which shows the best score (column 3) for each order of LMC (column 2) for the two datasets. Column 3 shows the number of matches from the total expected to have a score equal to or greater than the chosen score, and column 4 shows the probability of having a correct match given that the score is equal to or greater than the chosen score.

The scores become more discriminative as the order increases. Orders 3 and 4 are highly discriminative, however, the results are very similar both for the number of matches and the probability of a good match. We use order 3 to compute scores, since it achieves similar behavior when compared to order 4 with less points in the LMC. The chosen score

was 11 since the number of correct matches is substantially superior than the incorrect matches and it is a good score for both synthetic and underwater cases (from the score of 9 of the synthetic images there is only a small reduction in the number of matches to 0.84).

The score measure is specific for the types of images considered in our work. Other types of images would need another metric for the score or a more general measure could be pursued by exploring the structure of LMCs differently, possibly a measure based on graph edition distances [34].

### Number of Seeds

We use the previous results to compute seeds for Algorithm 5.1. We intend to choose  $k$  seeds such that the resulting matches are optimized. Particularly, the  $k$  seeds should be mostly correct seeds, so they were randomly chosen from matches agreeing with our previous choice of order 3 and score 11. The influence of number  $k$  of seeds is shown in Figure 5.11. The matching results become better as the number of seeds increases, but the gain practically stabilizes after 10 seeds. From 20 to 25 seeds, the difference is very small, which means that about 20 seeds suffice for the images in our dataset.

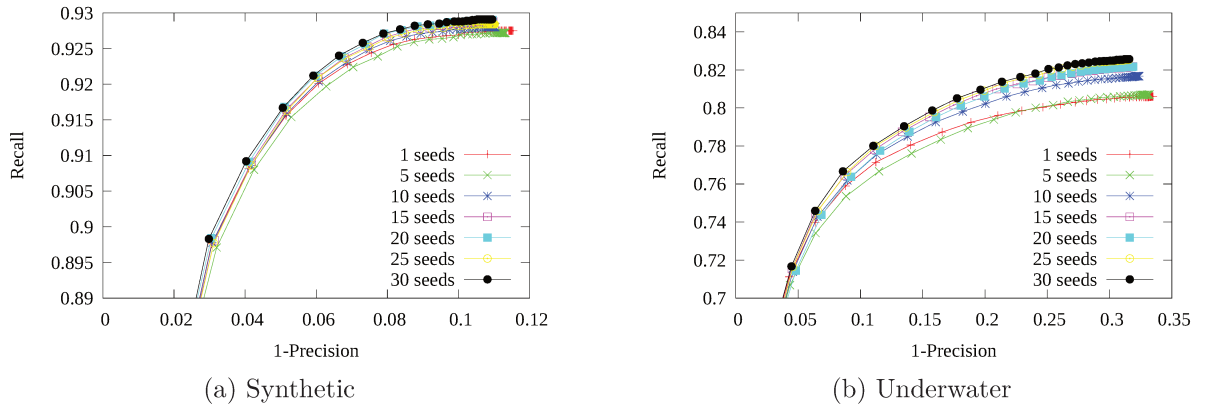


Figure 5.11: Relation between the number of seeds and number of matches. The number of correctly corresponded points increases with the number of randomly chosen seeds, but converges near 20 seeds.

#### 5.3.4 Matching Results

In this section, we show that our structural approach is more robust to find matches than an approach without structural information (1-NN). To select the seeds for our method, we randomly choose an interest point in one image and match it in the second image

using the 1-NN approach. The score for the match is computed by using LMC of order 3 and, if it is greater than or equal to 11, the match will be kept as a seed. The process is repeated until a set of 20 seeds (ideal number from the previous section) is acquired. The seeds computed in such a manner are the inputs for Algorithm 5.1.

The graph of Figure 5.12 shows the average results for all the test images. The nearest neighbor and LMC matching algorithms are performed using three different descriptors: a simple histogram of gray values (referred to as HGV), HOG descriptor and SIFT descriptor. We show that the LMC-guides matching is able to improve the results for weak descriptors (HGV) and strong descriptors (HOG, SIFT). When using the LMC-based match, the respective curve for a given descriptor is pushed towards the top-left corner to a greater degree when compared to the nearest neighbor matching and the same descriptor.

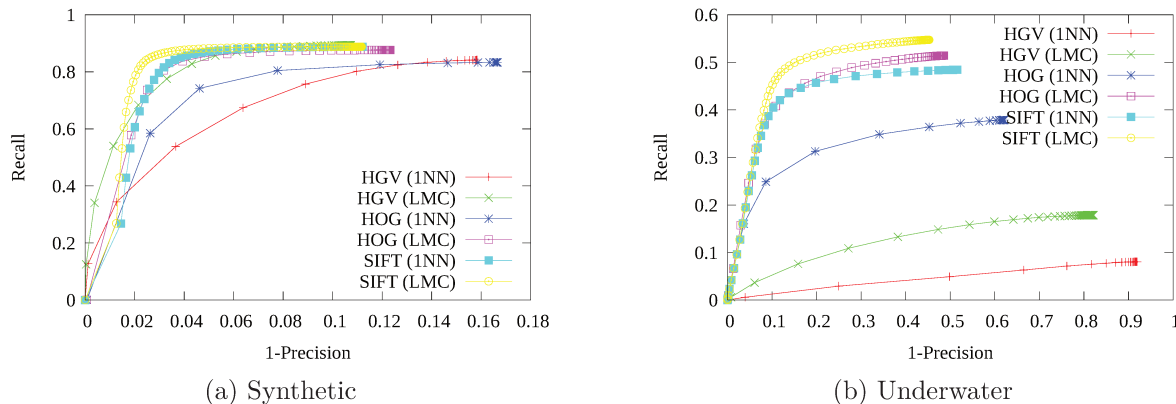


Figure 5.12: Average results for all the test images using HGV, HOG and SIFT descriptors. The recall of the LMC-based matching is higher, independent of the descriptor, when compared to the results obtained using the 1-NN matching. The conflicts of descriptors are avoided with the LMC neighborhood restriction, allowing to obtain more correct correspondences.

The HGV is considered to obtain a baseline for the results. Since the descriptor is weak, many incorrect correspondences are expected, specially for the challenging images such as the ones from the underwater dataset. The lines of the matching driven by the LMC show the improvement in correct matches when compared to the nearest neighbor approach.

The HOG and SIFT descriptors, as expected, improve the results for both methods (ours and nearest neighbor) when compared to the HGV. However, the LMC method still is able to obtain further improvement for the matches. Such a behavior reflects the properties of the structure-based matching carried out by our method. The number of

correct matches increases since conflicts of descriptors (interest points in different regions of an image but with close descriptors) are avoided with the neighborhood restriction to grow matches. Such behavior can be noticed in Figure 5.13, the solid lines show examples of matches that were correctly computed using the LMC-based algorithm while incorrectly corresponded with 1-NN.

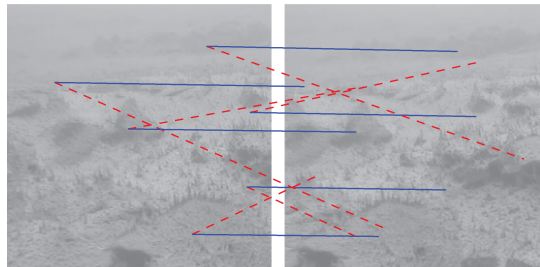


Figure 5.13: Examples of interest points correctly corresponded due to the use of the LMC (solid lines) but incorrectly corresponded by using the nearest neighbor approach (dashed lines).

The results show that LMC is a neighborhood relation that can be used to support image processing tasks. The correspondence of the images tested shows a particular case in which exploring the LMC helps traditional approaches. The correspondence algorithms described is limited to the types of images and transformations in the experimented datasets. Rotations between images could also be evaluated with the current method since the matching between LMCs is not dependent on the image type and the score could be adapted to consider the direction of the central points in LMCs to find the horizontal base line. Scale transformations is still a challenge that would require further studies on computing multiscale Morse complexes and neighborhood relations.

The complexity of the correspondence algorithm can be divided into three steps: the construction of a heap for the 1-NN used to find seeds; the process of actually finding seeds and the matching given by Algorithm 5.1. The heap construction can be done in linear time on the number of interest points. The results on score and seeds suggest that we have more than a 50% chance of finding a seed at each random run of the 1-NN. This expectation has been confirmed by our experiments and, therefore, for 20 seeds we expect the process to run in  $O(k \log n)$  with  $k$  approximately 40 and  $n$  features in a LMC. Finally, the computation of correspondences empirically suggests a linear time algorithm for small orders of the LMC. As the order increases, however, the constant multiplying the linear function can produce drastic augments on the computational time. The method is bounded by the SVD algorithm and, therefore, would take  $O(mn^2)$  time if we had LMCs with all the interest points in the images,  $m$  in the first image and  $n$  in the second.



## Chapter 6

# Conclusions and Future Work

In this thesis, we have presented algorithms for computing the discrete Morse complex of 2-dimensional images. The complex is modeled by the quad-edge data structure. We have also presented a topological operator, the Local Morse Context (LMC), to obtain neighborhoods of interest points.

The presented method is optimal since only paths in a vector field that may lead to paths in a Morse complex are processed. The algorithms properly deal with merging and branching cases and produce a consistent representation of the quad-edge model of the complex. The proposed simplified complex is easy to manipulate and useful to compute local topological operations. Furthermore, the model used for the complex is suitable for visualization tasks. Therefore, the way the complexes are modeled allows efficient numerical and graphical computations.

Theoretical and applied results have been presented to show the effectiveness of the method. The proposed complex is suitable for the computation of persistent Betti numbers [99], removal of topological noise [93] and acquisition of hierarchies of the discrete Morse complex [6, 26].

The LMC has been applied to find correspondences between interest points of stereo image pairs and to compute a measure for quantifying the confidence of matched points. This measure, denominated score, is effective for selecting matched pairs as seeds from which the number of matches is grown.

The matching algorithm explores the LMC neighborhood to produce correspondences in images agreeing with local proximity restrictions. As a consequence, the use of LMC avoids incorrect matches when the limitations of local descriptors do not allow a discrimination between various interest points. Finally, the LMC makes it possible to explore the topological relations between interest points in a general way that can be used for different types of images and applications.

Computational topology is a challenging field in computer science with many open

research possibilities involving algorithmic methods, modeling and applications in computational geometry, graphics, robotics, structural biology, and chemistry. Some directions for future work are listed below:

- a) Algorithms: the nature of the developed algorithms suggests that parallelizations can be performed to improve computations. All the algorithms we introduced for constructing the discrete Morse complexes are based on local computations, involving a cell, its faces or cofaces, and just a few other cells connected to the previous ones. Therefore, a speed up in the algorithms is expected to be achieved by treating these computations simultaneously for different paths and treating properly cases such as merges and branches. The extension of the algorithms and data structures for 3-dimensional images or even  $n$ -dimensional functions would be a challenging task. An extension and a generalization of the quad-edge are proposed in the works [24, 56]. Images in 3-dimensions are obtained, for instance, in physical simulations such as fluid dynamics, and from medical imagery such as tissue sections.
- b) Persistent homology applications: the persistent homology computed over Morse complexes is another point of future studies. Taking medical imagery again as an example, the persistent homology can be useful in the discrimination of patterns and classification of cells (here, as a biological concept) for computer-aided diagnosis. These types of microscopic images usually present variations in molecules that are important for the identification of diseases. The topological structure captured by persistent homology diagrams seems to be helpful in such case since it represents the prominence of a feature and can quantify features across different scales simultaneously. Nonetheless, a recurrent problem in such images is the presence of noise due to the acquisition process. The theory of persistence homology deals with noise [93]. Cohen-Steiner et al. [18] show that the persistence diagram is stable (noise is not a problem to capture the most important topological features), providing a useful tool in the analysis of noisy signals.
- c) Multiscale neighborhood: the discrete Morse complexes presented are computed at a specific scale. Similar to what we have discussed in the paper [22], a pyramid of images [82], such as the pyramid of Gaussians [57], can be used for computing Morse complexes at various scales. However, this would not be an efficient computation. We intend to study how to compute and model different scales of the Morse complexes such that the neighborhood can be generalized to multiple scales and applied to the correspondence of images under scale transformations.
- d) General scoring correspondences: the presented score measure is dependent on the



definition of a geometrical property shared by the type of images being studied. It is interesting to produce a type of score measure that does not need adaptation for every other type of image that may arise. This measure could be influenced by the graph matching algorithm. For example, if the score is a measure computed for the  $k$ -most similar correspondences between LMCs. In such a case it is important to we should guarantee that LMCs have at least  $k$  interest points and that  $k$  is enough to produce a good measure.

- e) Inter LMCs correspondences: the literature of graph matching is very extensive. We have considered a method that works based on complete graphs so that the 1-cells connecting critical cells are not taken into consideration. There are some graph matches based on edition distances that are worth being experimented. The 1-cells introduce restrictions on how the points can be corresponded and, therefore, the correspondence between LMCs is stronger. Another consideration about graph matches is the associated time cost, which should also be investigated so that LMCs of higher orders can be used.
- f) Descriptors based on LMCs: an interesting investigation topic is how to produce topological descriptors based on the LMCs. Topological descriptors based on persistence, such as barcodes, have been proved useful for some image processing applications. These are commonly descriptors for the images as a whole or for shapes. For the LMCs, the descriptor should consider the local point persistence.
- g) LMC order estimation: we intend to apply automatic optimization methods so that the orders of LMCs can be learned from the images.



# Bibliography

- [1] T. F. Banchoff. Critical Points and Curvature for Embedded Polyhedral Surfaces. *The American Mathematical Monthly*, 77(5):475–485, 1970.
- [2] T. Basak. Combinatorial Cell Complexes and Poincaré Duality. *Geometriae Dedicata*, 147(1):357–387, 2010.
- [3] H. Bay, T. Tuytelaars, and L. J. V. Gool. SURF: Speeded Up Robust Features. In *European Conference on Computer Vision*, pages 404–417, Graz, Austria, 2006.
- [4] K. Beketayev, G. Weber, M. Haranczyk, P.-T. Bremer, M. Hlawitschka, and B. Hamann. Topology-based Visualization of Transformation Pathways in Complex Chemical Systems. *Computer Graphics Forum*, 30(3):663–672, 2011.
- [5] P. T. Bremer, E. M. Bringa, M. A. Duchaineau, A. G. Gyulassy, D. Laney, A. Mascarenhas, and V. Pascucci. Topological Feature Extraction and Tracking. *Journal of Physics: Conference Series*, 78(1):1–5, 2007.
- [6] P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A Multi-Resolution Data Structure for Two-Dimensional Morse-Smale Functions. In *IEEE Visualization*, pages 139–146, Washington, DC, USA, 2003.
- [7] M. Brown, R. I. Hartley, and D. Nistér. Minimal Solutions for Panoramic Stitching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, MN, USA, 2007.
- [8] H. Bunke. Error-Tolerant Graph Matching: A Formal Framework and Algorithms. In *International Workshop on Advances in Pattern Recognition*, pages 1–14, London, UK, 1998.
- [9] G. Carlsson, G. Singh, and A. Zomorodian. Computing Multidimensional Persistence. In Y. Dong, D.-Z. Du, and O. Ibarra, editors, *Algorithms and Computation*, volume 5878 of *Lecture Notes in Computer Science*, pages 730–739. Springer Berlin Heidelberg, 2009.

- [10] G. Carlsson and A. Zomorodian. The Theory of Multidimensional Persistence. In *Symposium on Computational Geometry*, pages 184–193, Gyeongju, South Korea, 2007.
- [11] G. Carlsson, A. Zomorodian, A. Collins, and L. Guibas. Persistence barcodes for shapes. In *Eurographics Symposium on Geometry Processing*, pages 124–135, New York, NY, USA, 2004.
- [12] A. Cayley. On Contour and Slope Line. *The Philosophical Magazine*, 18(120):264–268, 1859.
- [13] J. Cech, J. Matas, and M. Perdoch. Efficient Sequential Correspondence Selection by Cosegmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1568–1581, 2010.
- [14] M. Chertok and Y. Keller. Efficient High Order Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2205–2215, Dec. 2010.
- [15] M. Cho and K. M. Lee. Bilateral Symmetry Detection via Symmetry-Growing. In *British Machine Vision Conference*, pages 4.1–4.11, London, UK, 2009.
- [16] M. K. Chung, P. Bubenik, and P. T. Kim. Persistence Diagrams of Cortical Surface Data. In *Conference on Information Processing in Medical Imaging*, pages 386–397, Williamsburg, VA, USA, 2009.
- [17] M. M. Cohen. *A Course in Simple-Homotopy Theory*. Springer, 1982.
- [18] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of Persistence Diagrams. *Discrete & Computational Geometry*, 37(1):103–120, Jan. 2007.
- [19] A. Collins, A. Zomorodian, G. Carlsson, and L. J. Guibas. A barcode shape descriptor for curve point cloud data. *Computers and Graphics*, 28(6):881–894, Dec. 2004.
- [20] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty Years of Graph Matching in Pattern Recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298, 2004.
- [21] B. Cyganek. *An Introduction to 3D Computer Vision Techniques and Algorithms*. John Wiley & Sons, 2007.

- [22] R. D. da Silva, W. R. Schwartz, and H. Pedrini. Scalar Image Interest Point Detection and Description Based on Discrete Morse Theory and Geometric Descriptors. In *IEEE International Conference on Image Processing*, pages 1877–1880, Orlando, FL, USA, 2012.
- [23] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, Washington, DC, USA, 2005.
- [24] D. P. Dobkin and M. J. Laszlo. Primitives for the Manipulation of Three-Dimensional Subdivisions. In *Symposium on Computational geometry*, pages 86–99, Waterloo, ON, Canada, 1987.
- [25] O. Duchenne, F. Bach, I.-S. Kweon, and J. Ponce. A Tensor-Based Algorithm for High-Order Graph Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2383–2395, Dec. 2011.
- [26] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse Complexes for Piecewise Linear 2-Manifolds. In *Symposium on Computational Geometry*, pages 70–79, Medford, MA, USA, 2001.
- [27] B. D. Fabio. *Shape from Functions: Enhancing Geometrical-Topological Descriptors*. PhD thesis, University of Bologna, Bologna, Italy, 2009.
- [28] B. D. Fabio and C. Landi. Persistent Homology and Partial Similarity of Shapes. *Pattern Recognition Letters*, 33(11):1445–1450, 2012.
- [29] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, June 1981.
- [30] R. Forman. Morse Theory for Cell Complexes. *Advances in Mathematics*, 134(1):90–145, 1998.
- [31] R. Forman. Morse Theory and Evasiveness. *Combinatorica*, 20(4):489–504, 2000.
- [32] R. Forman. A User’s Guide To Discrete Morse Theory. *Séminaire Lotharingien de Combinatoire*, 48:1–35, 2002.
- [33] J. Gamble and G. Heo. Exploring Uses of Persistent Homology for Statistical Analysis of Landmark-Based Shape Data. *Journal of Multivariate Analysis*, 101(9):2184–2199, Oct. 2010.

- [34] X. Gao, B. Xiao, D. Tao, and X. Li. A Survey of Graph Edit Distance. *Pattern Analysis & Applications*, 13(1):113–129, Jan. 2010.
- [35] R. Ghrist. Barcodes: The Persistent Topology of Data. *Bulletin of the American Mathematical Society*, 45:61–75, 2008.
- [36] L. Grady and J. R. Polimeni. *Discrete Calculus - Applied Analysis on Graphs for Computational Science*. Springer, 2010.
- [37] L. Guibas and J. Stolfi. Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. *ACM Transactions on Graphics*, 4(2):74–123, Apr. 1985.
- [38] F. Harary. *Graph Theory*. Addison-Wesley, 1969.
- [39] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2003.
- [40] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes. In *Conference on Computer Graphics and Interactive Techniques*, pages 203–212, Los Angeles, CA , USA, 2001.
- [41] T. Ishkhanov. A Topological Method for Shape Comparison. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–4, Anchorage, AK, USA, 2008.
- [42] H. Jiang, T.-P. Tian, and S. Sclaroff. Scale and Rotation Invariant Matching using Linearly Augmented Trees. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2473–2480, Colorado Springs, CO, USA, 2011.
- [43] J. Kannala, E. Rahtu, S. Brandt, and J. Heikkila. Object Recognition and Segmentation by Non-Rigid Quasi-Dense Matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Anchorage, AK, USA, 2008.
- [44] Y. Ke and R. Sukthankar. PCA-SIFT: a More Distinctive Representation for Local Image Descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 506–513, Washington, DC, USA, 2004.
- [45] G. Kim, C. Faloutsos, and M. Hebert. Unsupervised Modeling and Recognition of Object Categories with Combination of Visual Contents and Geometric Similarity Links. In *ACM International Conference on Multimedia Information Retrieval*, pages 419–426, Vancouver, Canada, 2008.

- [46] H. King, K. Knudson, and N. Mramor. Generating Discrete Morse Functions from Point Data. *Experimental Mathematics*, 14(4):435–444, 2005.
- [47] V. A. Kovalevsky. Finite Topology as Applied to Image Analysis. *Computer Vision, Graphics and Image Processing*, 46(2):141–161, May 1989.
- [48] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Efficient Subwindow Search: A Branch and Bound Framework for Object Localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2129–2142, Dec. 2009.
- [49] M. Leordeanu and M. Hebert. A Spectral Technique for Correspondence Problems Using Pairwise Constraints. In *IEEE International Conference on Computer Vision*, pages 1482–1489, Beijing, China, 2005.
- [50] D. Letscher and J. Fritts. Image Segmentation Using Topological Persistence. In *Conference on Computer Analysis of Images and Patterns*, pages 587–595, Vienna, Austria, 2007.
- [51] T. Lewiner. Constructing Discrete Morse Functions. Master’s thesis, Pontifical Catholic University, Rio de Janeiro, RJ, Brazil, July 2002.
- [52] T. Lewiner. *Geometric Discrete Morse Complexes*. PhD thesis, Pontifical Catholic University, Rio de Janeiro, RJ, Brazil, Aug. 2005.
- [53] T. Lewiner, H. Lopes, and G. Tavares. Optimal Discrete Morse Functions for 2-manifolds. *Computational Geometry: Theory and Applications*, 26(3):221–233, Nov. 2003.
- [54] T. Lewiner, H. Lopes, and G. Tavares. Applications of Forman’s Discrete Morse Theory to Topology Visualization and Mesh Compression. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):499–508, Sept. 2004.
- [55] M. Lhuillier and L. Quan. A Quasi-Dense Approach to Surface Reconstruction from Uncalibrated Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):418–433, 2005.
- [56] P. Lienhardt. Subdivisions of  $n$ -Dimensional Spaces and  $n$ -Dimensional Generalized Maps. In *Symposium on Computational geometry*, pages 228–236, Saarbruchen, Germany, 1989.
- [57] T. Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, Netherlands, 1994.

- [58] D. G. Lowe. Object Recognition from Local Scale-Invariant Features. In *IEEE International Conference on Computer Vision*, pages 1150–1157, Washington, DC, USA, 1999.
- [59] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004.
- [60] P. Magillo, E. Danovaro, L. Floriani, L. Papaleo, and M. Vitali. A Discrete Approach to Compute Terrain Morphology. In J. Braz, A. Ranchordas, H. Araújo, and J. Pereira, editors, *Computer Vision and Computer Graphics. Theory and Applications*, volume 21 of *Communications in Computer and Information Science*, pages 13–26. Springer Berlin Heidelberg, 2009.
- [61] W. S. Massey. *A Basic Course in Algebraic Topology*. Springer-Verlag, 1991.
- [62] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, Oct. 2005.
- [63] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, 65(1-2):43–72, Nov. 2005.
- [64] J. W. Milnor. *Morse Theory*. Annals of Mathematics Studies. Princeton University Press, 1963.
- [65] E. G. Minian. Some Remarks on Morse Theory for Posets, Homological Morse Theory and Finite Manifolds. *Topology and its Applications*, 159(12):2860–2869, 2012.
- [66] K. Mischaikow and V. Nanda. Morse Theory for Filtrations and Efficient Computation of Persistent Homology. *Discrete & Computational Geometry*, 50(2):330–353, 2013.
- [67] H. Molina-Abril and P. Real. Homological Optimality in Discrete Morse Theory Through Chain Homotopies. *Pattern Recognition Letters*, 33(11):1501–1506, 2012.
- [68] J. R. Munkres. *Topology*. Prentice Hall, 2000.
- [69] D. Neilson and Y.-H. Yang. Evaluation of Constructable Match Cost Measures for Stereo Correspondence Using Cluster Ranking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Anchorage, AK, USA, 2008. address <http://webdocs.cs.ualberta.ca/~stereo/datasets/>.



- [70] E. Nowak, F. Jurie, and B. Triggs. Sampling Strategies for Bag-of-Features Image Classification. In *European Conference on Computer Vision*, pages 490–503, Graz, Austria, 2006.
- [71] K. Oliver, W. Hou, and S. Wang. Feature Matching in Underwater Environments Using Sparse Linear Combinations. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 60–67, San Francisco, CA, USA, 2010.
- [72] ORL. The Database of Faces, 2013. <http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>.
- [73] S. Paris and F. Durand. A Topological Approach to Hierarchical Segmentation using Mean Shift. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Los Alamitos, CA, USA, 2007.
- [74] R. Raveaux, J.-C. Burie, and J.-M. Ogier. A Graph Matching Method and a Graph Matching Distance Based on Subgraph Assignments. *Pattern Recognition Letters*, 31(5):394–406, Apr. 2010.
- [75] J. Reininghaus, N. Kotava, D. Gunther, J. Kasten, H. Hagen, and I. Hotz. A Scale Space Based Persistence Measure for Critical Points in 2D Scalar Fields. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2045–2052, 2011.
- [76] V. Robins, P. J. Wood, and A. P. Sheppard. Theory and Algorithms for Constructing Discrete Morse Complexes from Grayscale Digital Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1646–1658, Aug. 2011.
- [77] W. R. Schwartz, R. D. da Silva, L. S. Davis, and H. Pedrini. A Novel Feature Descriptor Based on the Shearlet Transform. In *IEEE International Conference on Image Processing*, pages 1033–1036, Brussels, Belgium, 2011.
- [78] G. L. Scott and H. C. Longuet-Higgins. An Algorithm for Associating the Features of Two Images. *Royal Society London*, 244(1309):21–26, 1991.
- [79] L. G. Shapiro and R. M. Haralick. Structural Descriptions and Inexact Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(5):504–519, Sept. 1981.
- [80] L. S. Shapiro and J. M. Brady. Feature-Based Correspondence: An Eigenvector Approach. *Image and Vision Computing*, 10(5):283–288, 1992.

- [81] A. Sole, V. Caselles, G. Sapiro, and F. Arandiga. Morse Description and Geometric Encoding of Digital Elevation Maps. *IEEE Transactions on Image Processing*, 13(9):1245–1262, 2004.
- [82] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis and Machine Vision*. Thomson Learning, 2007.
- [83] D. Sumner. Antarctic Microbial Mats, 2013. <http://mygeologypage.ucdavis.edu/sumner/Antarctica.html>.
- [84] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., 1st edition, 2010.
- [85] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing Visual Features for Multi-class and Multiview Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):854–869, May 2007.
- [86] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.
- [87] W.-H. Tsai and K.-S. Fu. Error-Correcting Isomorphisms of Attributed Relational Graphs for Pattern Analysis. *IEEE Transactions on Systems, Man and Cybernetics*, 9(12):757–768, Dec. 1979.
- [88] J. R. Ullmann. An Algorithm for Subgraph Isomorphism. *Journal of the ACM*, 23(1):31–42, 1976.
- [89] S. Umeyama. An Eigendecomposition Approach to Weighted Graph Matching Problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):695–703, Sept. 1988.
- [90] USGS. United States Geological Survey, 2013. <http://www.usgs.gov>.
- [91] L. Čomić, L. De Floriani, and F. Iuricich. Simplifying Morphological Representations of 2D and 3D Scalar Fields. In *ACM International Conference on Advances in Geographic Information Systems*, pages 437–440, Chicago, IL, USA, 2011.
- [92] H. Wagner, P. Dłotko, and M. Mrozek. Computational Topology in Text Mining. In M. Ferri, P. Frosini, C. Landi, A. Cerri, and B. Fabio, editors, *Computational Topology in Image Context*, volume 7309 of *Lecture Notes in Computer Science*, pages 68–78. Springer Berlin Heidelberg, 2012.

- [93] B. Wang. *Separating Features from Noise with Persistence and Statistics*. PhD thesis, Duke University, Durham, NC, USA, 2010.
- [94] C. Wang and K.-K. Ma. Common Visual Pattern Discovery via Directed Graph Model. In *IEEE International Conference on Image Processing*, pages 2957–2960, Brussels, Belgium, 2011.
- [95] L. Wang, F. Tang, Y. Guo, S. H. Lim, and N. L. Chang. Exploiting Feature Correspondence Constraints for Image Recognition. In *IEEE International Conference on Image Processing*, pages 1769–1772, Brussels, Belgium, 2011.
- [96] K. J. Weiler. *Topological Structures for Geometric Modeling*. PhD thesis, Rensselaer Polytechnic Institute, Troy, NY, USA, Aug. 1986.
- [97] J. H. C. Whitehead. Simplicial Spaces, Nuclei and m-Groups. *Proceedings of the London Mathematical Society*, 45(1):243–327, 1939.
- [98] B. Zitová and J. Flusser. Image Registration Methods: a Survey. *Image and Vision Computing*, 21(11):977–1000, 2003.
- [99] A. Zomorodian. *Computing and Comprehending Topology: Persistence and Hierarchical Morse Complexes*. PhD thesis, University of Illinois at Urbana-Champaign, 2001.
- [100] A. Zomorodian. Computational Topology. In M. Atallah and M. Blanton, editors, *Algorithms and Theory of Computation Handbook*, volume 2, chapter 3. Chapman & Hall/CRC Press, second edition, 2010.
- [101] A. Zomorodian, M. J. Ablowitz, S. H. Davis, E. J. Hinch, A. Iserles, J. Ockendon, and P. J. Olver. *Topology for Computing*. Cambridge University Press, 2005.