SKELETONIZATION METHODS FOR IMAGE AND VOLUME INPAINTING

ANDRÉ SOBIECKI

This research was supported by the Brazilian national research program *Science without Borders*[1] under the project number 202535/2011-8.

Cover: TODO

Skeletonization Methods for Image and Volume Inpainting

André Sobiecki
Thesis Rijksuniversiteit Groningen

ISBN TODO (printed version)
ISBN TODO (electronic version)

1 http://www.cienciasemfronteiras.gov.br/web/csf-eng

university of groningen

Skeletonization Methods for Image and Volume Inpainting

PHD THESIS

to obtain the degree of PhD at the
University of Groningen
on the authority of the
Rector Magnificus, dr. E. Sterken
and in accordance with
the decision by the College of Deans.

This thesis will be defended in public on

final date to be inserted

by

ANDRÉ SOBIECKI

born on October 29, 1988
in São Bento do Sul, Brazil.

PROMOTOR:	Prof. dr. A. C. Telea
COPROMOTOR:	Dr. A. C. Jalba
ASSESSMENT COMMITTEE:	Prof. dr. *Marcelo Walter*
	Prof. dr. *Alexandre Falcão*
	Prof. dr. *Nicolai Petkov*

ABSTRACT

Image and shape restoration techniques are increasingly important in application domains which require high-quality 2D images and 3D shapes, but where only shapes and images damaged by various types of defects are available. Many types of restoration techniques have been proposed in the 2D image-processing and the 3D shape-processing context, respectively. Well-known examples of such techniques include digital inpainting, denoising, and morphological gap filling. However efficient and effective, such methods have several limitations with respect to the shape, size, distribution, and nature of the defects they can find and eliminate.

Skeletonization describes a set of techniques aiming at the detection and computation of simple descriptors that capture well a shape's geometry, topology, and symmetry. Many such descriptors have been proposed in the literature, ranging from simple 2D medial axes of digital binary images, up to complex medial surfaces and curve skeletons used to describe high-resolution 3D shapes.

This thesis researches the possibility of using skeletal structures for the design and implementation of 2D image and 3D shape inpainting methods, based on the underlying intuition that the geometry, topology, and symmetry data jointly captured by skeletons greatly help controlling the restoration process. We start exploring the above hypothesis by studying the use of 2D skeletons for the restoration of two-dimensional images. To this end, we show that skeletons are indeed useful and efficient descriptors that support three different kinds of image restoration – semantic inpainting, salient gap filling, and digital hair removal. To explore our hypothesis in the 3D case, we first overview the existing state-of-the-art 3D skeletonization methods, and conclude that no such method provides us with the features required by efficient and effective practical usage. We next propose a novel method for 3D skeletonization, and show how it complies with our desired quality requirements, which makes it thereby suitable for our shape restoration context. Based on this method, we describe two different types of 3D shape restoration – gap filling and wire-artifact removal.

The joint results of our study show that skeletons are indeed effective tools to design a variety of shape restoration methods. Separately, our results show that suitable algorithms and implementations can be conceived to yield high end-to-end performance and quality of skeleton-based restoration methods. Finally, our practical applications involving the above-mentioned restoration methods show that these can generate competitive results when compared to existing state-of-the-art in application areas such as digital hair removal and wire artifact removal.

SAMENVATTING

Technieken voor restauratie van digitale afbeeldingen en vormen worden steeds belangrijker voor applicaties waarin hoge kwaliteit 2D beelden en 3D vormen nodig zijn, maar waarin men alleen vormen en beelden heeft die beschadigd zijn door verschillende types defecten. Veel technieken hiervoor zijn bekend in de context van 2D beeldbewerking en 3D vormbewerking. Voorbeelden hiervan zijn *digital inpainting*, ruisverwijdering (*denoising*), en morphologische *gap filling*. Hoewel efficiënt en effectief, dergelijke methodes hebben beperkingen ten opzichte van de vorm, grootte, distributie, en aard van de defecten die zij kunnen vinden en verwijderen.

Skeletonisatie beschrijft een set van technieken gericht op de detectie en berekening van simpele descriptoren die de geometrie, topologie, en symmetrie van vormen afvangen. Veel dergelijke descriptoren zijn voorgesteld in de literatuur, variërend van eenvoudige 2D mediale assen (*medial axes*) tot complexe mediale oppervlakken en curve skeletten die worden gebruikt om hoge-resolutie 3D vormen te beschrijven.

Dit proefschrift onderzoekt de mogelijkheid om skeletale structuren te gebruiken voor het ontwerp and de implementatie van 2D-beeld en 3D-vorm *inpainting*, gebaseerd op de intuitie dat de geometrie, topologie, en symmetrie afgevangen door skeletten van groot hulp zijn voor het restauratieproces. We beginnen de bovengenoemde hypothese te verkennen door het bestuderen van het gebruik van 2D skeletten voor de restauratie van tweedimensionale beelden. Daartoe laten we zien dat skeletten inderdaad nuttige en efficiënte descriptoren zijn voor de ondersteuning van drie types herstel van beelden – semantische inpainting, saillante gap-filling en digitale ontharing. Om onze hypothese in het 3D geval te verkennen, creëren wij eerst een overzicht van de huidige state-of-the-art 3D skeletonisatiemethodes, waaruit we afleiden dat geen dergelijke methode de eigenschappen heeft voor efficiënt en effectief praktisch gebruik. Vervolgens stellen wij een nieuwe 3D skeletonisatiemethode voor en laten zien hoe deze voldoet aan onze gewenste kwaliteitseisen, waardoor deze geschikt is voor ons vormrestauratiecontext. Op basis van deze methode beschrijven wij twee verschillende types van 3D vormrestauratie – 3D *gap filling* en draad-artefact verwijdering.

De gezamenlijke resultaten van onze studie laten zien dat skeletten inderdaad effectieve instrumenten om een verscheidenheid van vormrestauratiemethodes te ontwerpen. Daarnaast laten onze resultaten zien dat geschikte algoritmen en implementaties kunnen worden bedacht om hoge prestaties en kwaliteit van skeletgebaseerde restauratiemethoden te garanderen. Tenslotte laten onze praktische toepassingen, die de bovengenoemde restauratiemethodes gebruiken, zien dat deze concurrerende resultaten kunnen genereren in vergelijking met de state-of-the-art op toepassingsgebieden zoals digitale ontharing en metaaldraad-artefact verwijdering.

PUBLICATIONS

This thesis is based on material which has appeared in the publications listed below.

CONFERENCES AND JOURNALS

- A. Sobiecki, A. Telea, G. Giraldi, L. Neves, and C. Thomaz. Low-cost automatic inpainting for artifact suppression in facial images. In *Proc. 8^{th} International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*, pages 324–335, 2013

- A. Sobiecki, H. Yasan, A. Jalba, and A. Telea. Qualitative comparison of contraction-based curve skeletonization methods. In *Mathematical Morphology and Its Applications to Signal and Image Processing (Proc. ISMM)*, pages 425–439. Springer LNCS 7883, 2013

- A. Sobiecki, A. Jalba, D. Boda, A. Diaconeasa, and A. Telea. Gap-sensitive segmentation and restoration of digital images. In *Proc. Computer Graphics and Visual Computing (CGVC)*, pages 136–144. Eurographics, 2014

- A. Sobiecki, A. Jalba, and A.C.Telea. Comparison of curve and surface skeletonization methods for voxel shapes. *Pattern Recogn. Lett.*, 47:147–156, October 2014

- J. Koehoorn, A. Sobiecki, D. Boda, A. Diaconeasa, S. Doshi, S. Paisey, A. Jalba, and A. Telea. Automated digital hair removal by threshold decomposition and morphological analysis. In *Mathematical Morphology and Its Applications to Signal and Image Processing (Proc. ISMM)*, pages 15–26. Springer LNCS 9082, 2015

- A. Jalba, A. Sobiecki, and A. Telea. An unified multiscale framework for planar, surface, and curve skeletonization. *IEEE TPAMI*, 2015. DOI 10.1109/TPAMI.2015.2414420

- J. Koehoorn, A. Sobiecki, P. Rauber, A.Jalba, and A. Telea. Efficient and effective automated digital hair removal from dermoscopy images. *submitted*, 2015

EXTENDED ABSTRACTS AND POSTERS

- A. Sobiecki, A. Jalba, and A. Telea. Qualitative and quantitative comparison of curve and surface skeletons – a state of the art review. In *ICTOpen, Eindhoven, The Netherlands*, 2013

- A. Sobiecki, A. Jalba, and A. Telea. Automatic gap restoration in 2D and 3D digital images. In *ICTOpen, Amersfoort, The Netherlands*, 2015

- A. Sobiecki, J. Koehoorn, D. Boda, C. Solovan, A. Diaconeasa, A. Jalba, and A. Telea. A new efficient method for digital hair removal by dense threshold analysis. In *Proc. 4^{th} World Congress of Dermoscopy (IDS), Vienna, Austria*, 2015

CONTENTS

1 INTRODUCTION 1
1.1 The Regularization Challenge 2
1.2 Shape and Image Restoration 2
1.3 Structural Shape Descriptors 3
1.4 Research Question 4
1.5 Structure of the Thesis 5

2 RELATED WORK 9
2.1 Digital Image Restoration and Inpainting 9
2.1.1 Patch-based inpainting methods 10
2.1.2 Sparse inpainting methods 11
2.1.3 Differential/variational inpainting methods 11
2.1.4 3D inpainting 14
2.2 Shape Skeletonization 14
2.2.1 Definitions 15
2.2.2 Overview of Skeletonization Methods 17
2.2.3 Properties 20
2.3 Conclusions 23

3 INPAINTING IN FACIAL IMAGES 25
3.1 Introduction 25
3.2 Related Work 26
3.3 Proposed Method 27
3.3.1 Image Quality Index 28
3.3.2 Statistical Artifact Segmentation 29
3.3.3 Semantic Inpainting 30
3.4 Results 33
3.5 Conclusions 39

4 SKELETON-BASED GAP DETECTION AND FILLING IN BINARY IMAGES 41
4.1 Introduction 41
4.2 Related work 42
4.3 Our Proposal 43
4.3.1 Gap Closing 43
4.3.2 Gap Classification 44
4.3.3 Error Gap Restoration 45
4.3.4 Implementation 46
4.4 Results 46
4.5 Applications for Skin Imaging 49
4.6 Discussion 52
4.7 Conclusion 53

5 SKELETON-BASED DIGITAL HAIR REMOVAL 55
5.1 Introduction 55
5.2 Related Work 56

5.3 Proposed Method 57
5.3.1 Threshold-set Decomposition 57
5.3.2 Potential Long Hair Detection 59
5.3.3 False Positive Elimination 60
5.3.4 Long Hair Removal 62
5.3.5 Stubble Detection and Removal 63
5.4 Implementation 65
5.5 Results and Comparison 66
5.6 Discussion 70
5.7 Conclusions 75

6 CURVE SKELETONIZATION COMPARISON 77
6.1 Introduction 77
6.2 Related work 78
6.3 Comparison criteria 79
6.4 Comparison 80
6.4.1 Overview 82
6.4.2 Homotopy 82
6.4.3 Centeredness 83
6.4.4 Detail preservation 84
6.4.5 Smoothness 85
6.4.6 Sampling robustness 86
6.5 Discussion 87
6.6 Conclusions 88

7 CURVE AND SURFACE SKELETON COMPARISON 91
7.1 Introduction 91
7.2 Related work 92
7.2.1 Skeletonization methods 92
7.2.2 The challenge of comparison 93
7.3 Methods 94
7.3.1 Comparison criteria 94
7.3.2 Selected methods for comparison 96
7.4 Comparison methodology 97
7.5 Results 99
7.5.1 Global comparison 99
7.5.2 Detailed comparison 104
7.6 Discussion 107
7.7 Conclusion 109

8 UNIFIED SKELETONIZATION 111
8.1 Introduction 111
8.2 Related work 112
8.3 Proposed framework 114
8.3.1 Preliminaries 114
8.3.2 Physically-based skeletonization model 114
8.4 Solving the system 116
8.4.1 Topologically-constrained boundary evolution by density-ordered thinning 116
8.4.2 Algorithm 117
8.4.3 Density transport 121

8.4.4 Detecting different skeleton types 124
8.4.5 Implementation details and parameter settings 126
8.5 Comparative results 126
8.5.1 Two-dimensional skeletons 126
8.5.2 Surface skeletons 128
8.5.3 Curve skeletons 131
8.6 Discussion 132
8.6.1 Method properties 132
8.6.2 Comparison with Hamiltonian methods 135
8.7 Conclusions 135

9 GAP DETECTION AND REMOVAL FOR 3D SHAPES 137
9.1 Introduction 137
9.2 Removal of One-Dimensional Gaps from Grayscale Volumes 138
9.2.1 Application context 138
9.2.2 Proposed method 140
9.2.3 Results 141
9.3 Removal of One- and Two-Dimensional Gaps from Binary Volumes 143
9.3.1 Proposed method 144
9.3.2 Results 146
9.3.3 Discussion 148
9.4 Conclusions 149

10 CONCLUSION 151
10.1 Summary 151
10.2 Part 1: Repairing two-dimensional images 152
10.2.1 Facial Image Restoration (Chapter 3) 152
10.2.2 Gap-Sensitive Segmentation and Restoration of Images (Chapter 4) 153
10.2.3 Digital hair removal in skin tumor images (Chapter 5) 153
10.3 Part 2: Repairing three-dimensional shapes 154
10.3.1 3D Curve Skeletonization Comparison (Chapter 6) 154
10.3.2 3D Voxel-based Skeletonization Comparison (Chapter 7) 154
10.3.3 Unified Curve-and-Surface Skeletonization Framework (Chapter 8) 155
10.3.4 3D Artifact Detection and Removal (Chapter 9) 155

BIBLIOGRAPHY 157

ACKNOWLEDGEMENTS 173

1 INTRODUCTION

The last decades have witnessed an enormous increase of applications revolving around the manipulation of two-dimensional and three-dimensional image content. This increase has been fueled by several factors, such as the advent of low-cost and high-performance computing platforms; the appearance of affordable high-resolution content acquisition devices such as two-dimensional (2D) digital cameras and three-dimensional (3D) scanners; and the development of increasingly more powerful and versatile algorithms and techniques for the analysis, synthesis, and processing of the aforementioned content. Many industries are thriving on such applications, such as medical imaging, videosurveillance, 3D printing, geoinformatics, video and movie production, and computer games, to mention just a few.

While the use of 2D and 3D digital content has many flavors, a significant fraction of applications thereof focuses on the extraction, interpretation, and manipulation of high-level information from plain 2D images and 3D scans. Arguably the most salient high-level information type pertaining to such digital content are *shapes*. Loosely put, shapes are subsets of 2D images or 3D volumes that share specific topological, geometrical, spatial, or appearance-related properties, and describe particular classes of objects present in the acquired content. Examples hereof are many: humans or vehicles identified in 2D still images or motion videos acquired by cameras; organs or parts thereof present in 2D X-ray imagery, 2D microscope imagery, or 3D computer tomography (CT) or magnetic resonance imaging (MRI) scans; geographical relief forms present in 2D satellite and geosurveillance imagery; and physical objects scanned by 3D laser scanners or 3D time-of-flight cameras. Reasoning about shapes present in an image offers significantly more powerful ways to interpret and leverage the flood of imagery 'big data' available to us. Indeed, the majority of applications using 2D and 3D images center on detecting, quantifying, and interpreting such shapes present in the raw image data, as these, and not the raw image data, capture the actual concerns and questions of the involved stakeholders.

However, the path from acquiring raw 2D and 3D images to extracting useful and usable shapes embedded in such images, is long and complicated. Many problems can occur in this process – the separation of shapes from surrounding unstructured so-called background (a process usually known as *segmentation*); the elimination of irrelevant variations in shape size, orientation, and noise (a process usually known as *regularization*); the computation of relevant metrics on the extracted shapes, such as size, thickness, topological structure, and boundary curvature (a process usually known as *quantification*); and the encoding of shapes into simple and compact representations which allow efficient operations such as shape classification, registration, matching, and retrieval (a process known as *descriptor extraction*).

1.1 THE REGULARIZATION CHALLENGE

Among the above-mentioned challenges pertaining to the usage of 2D and 3D shapes, a fundamental one sticks out: regularization. As outlined above, regularization includes all processing operations applied to a shape, extracted from raw 2D or 3D imagery by segmentation methods, aiming to eliminate many types of variations present in segmented shapes which, by themselves, do not further provide information or added-value to the subsequent shape-analysis operations implied by the application domain.

In general, shape regularization is a hard problem, as it attempts to enforce high-level constraints on the shapes extracted by potentially any types of methods. Many classes of regularization methods exist, depending on the specific shape aspects which are to be guaranteed or, conversely, on the shape aspects which are deemed irrelevant to the application context, and thus are to be eliminated. For example, geometrical regularization methods aim to produce shapes having a given boundary smoothness, which is achieved by applying various filtering techniques that preserve coarse-scale details and eliminate small-scale details deemed to be noise [223]. Morphological regularization methods aim to produce shapes having given sizes, genii, or minimal thicknesses [197]. Other more specialized methods aim to produce shapes that respect higher-level invariants such as branching structure or correspondence to a given prior family [61].

1.2 SHAPE AND IMAGE RESTORATION

Within the family of shape regularization methods, two important classes can be identified. The first class can be loosely described as working *globally*: Given a shape S and a shape template T, one aims to produce a regularized version S_R of S so that S_R obeys the characteristics imposed by T. Such methods are extremely powerful as they aim to guarantee high-level properties of the regularization S_R. An example of global regularization methods are techniques based on active contours, which enforce desirable geometric and topological constraints on the regularized shapes by explicitly building these constraints into the regularization model [119]. However, designing such global regularization methods is very hard, since, in general, there are many ways in which a shape S can be modified to comply with a set T of global properties. The second class of regularization methods has a *local* nature: Given a shape S and a set of (spatially) local criteria C, one aims to produce a regularized version S_R of S so that S_R obeys C over every local neighborhood of S_R having a given size. Local regularization methods are, in general, easier to design and implement than their global counterparts, and also more computationally efficient, given their local nature. A well-known example of local regularization is low-pass filtering aimed at ensuring that the resulting shape S_R has a given boundary smoothness [61].

Within the class of local regularization methods, *restoration* methods occupy a particular place. The origin of such methods is based on the fact that 2D and 3D image acquisition is subject to many local errors. Examples of such errors are color artifacts, cracks, and fine-scale noise present in digitized versions of old paper photographs or prints [23]; geometrical and topological noise present on the boundaries of shapes seg-

mented from low-contrast, high-noise scans [225]; larger-scale details, such as inscriptions or logos that obliterate parts of images [64, 229] or hairs that obliterate tumor areas in dermoscopic images [104, 138, 241]. As such, the acquired shapes S can be thought of being versions of the original shapes under scrutiny, but 'contaminated' by local defects. Restoration methods propose to eliminate such defects by a two-step process: First, the defect areas are *detected* using priors describing the expected shapes. Next, the defect areas are *restored* using information in neighboring reliable areas, and possibly global constraints. Such methods are, in many contexts, also known under the name of *inpainting* methods [26]. This name stems from the traditional procedure of manually restoring 2D paintings by locating damaged regions, followed by reconstructing image content in these regions by extrapolating, or 'in painting', content outside these regions into the regions themselves [23, 24, 38, 46, 64, 144, 169, 229].

Inpainting methods have shown their strong added-value in several contexts involving the restoration of 2D images. In general, these methods are extremely versatile with respect to the type of restorations they can cover; robust and fast, thus applicable on large and complex imagery; and, last but not least, simple and generic to implement. However, such methods operate by excellence locally, following the analogy with classical inpainting, where the damaged zone is reconstructed based on information present in a small neighborhood of the defect. While this allows enforcing many types of local desirable properties on the reconstructed areas, such as smoothness and texture, other equally important desirable properties have a global nature, *e.g.*, shape topology, thickness, and symmetry. Enforcing such global properties during restoration is not trivial. Separately, classical inpainting methods decouple the detection of damaged zones from their restoration. While the restoration proper is generally well taken care of, the detection of damaged zones is typically application-dependent, and not covered by the inpainting proposal itself.

1.3 STRUCTURAL SHAPE DESCRIPTORS

The problem of combining local and global reasoning about shapes is crucial in many areas of shape analysis and processing, and as such, has received much attention. The problem is typically approached by extracting a number of so-called *shape descriptors* from the available 2D or 3D shapes. Such descriptors are designed to capture important shape properties, such as planarity, corners, edges, concave and convex boundary regions, local shape thickness, global symmetry relations, the shape's part-whole structure, and the shape's topology, in ways that allow one to use such information efficiently and effectively for the desired analysis and processing tasks.

One particular class of shape descriptors is formed by medial axes, also known as skeletons. Medial axes have been introduced in 1973 by Blum in the context of analyzing two-dimensional binary images [34]. One major property of skeletons is their ability to encode the topological information present in a shape, *i.e.*, the shape's genus but also its part-whole structure, compactly and onto a structure of lower dimensionality than the shape itself. For instance, classical skeletons of 2D binary images are sets of 1D curves, which can be easily analyzed to detect and reason about shape features such as part tips, number and size of parts, and how parts are joined to create the shape. If information on the distance of each skeletal point to its closest

shape-boundary point is added to the skeleton, one obtains the so-called medial axis transform (MAT), which allows reconstructing shapes from their skeletons, *i.e.*, provides a dual shape representation to the classical boundary representation [135, 203]. Additional information linking skeletal points to their closest boundary points can be added by the so-called feature transform [203], which allows reasoning about the shape boundary's convexity and also detect features such as edges and corners. Finally, skeletal representations can also be modeled as multiscale objects, which allows progressive simplification, denoising, classification, and other level-of-detail analyses for shapes [80, 168, 186, 187, 228].

Overall, skeletons capture both geometrical and topological information about a shape, and allow accessing this information in a multiscale fashion. As such, skeletons form the basis of many types of shape analysis and processing applications, such as shape matching and retrieval [152], shape denoising [224, 225], shape segmentation [15, 184, 185], and shape metrology [79, 109]. Recent methods allow the robust and efficient computation of 2D skeletons [80, 228] and also 3D skeletons [102, 109] for large and complex real-world 2D and 3D images. As such, skeletons emerge as a powerful descriptor that supports shape processing applications which need to combine local and global information about the shape.

1.4 RESEARCH QUESTION

Summarizing the above discussion, we note two main aspects. First, a main challenge for 2D and 3D shape reconstruction resides in the difficulty of designing good detectors for damaged regions that capture a wide spectrum of local and global shape properties. Secondly, a main advantage of skeletal shape descriptors resides in precisely providing joint information on local and global shape properties. The above two aspects lead us to the formulation of this thesis' main research question:

Can skeletal descriptors be used to design efficient and effective shape restoration methods, based on inpainting techniques, that produce shapes having a wide set of both local and global desirable properties?

To construct a path towards answering the above research question, several aspects have to be considered. Globally put, all these aspects regard the concrete ways in which the above-stated research question is further *elaborated* to a more concrete level where actual experiments and algorithm designs can be proposed. The most important such aspects are outlined below.

1. *Detection vs restoration:* The effect of the aforementioned desirable shape properties has to be studied for both the process of detecting damaged regions, and also for the inpainting process itself that is used to reconstruct these regions. As already mentioned, the above steps of detection and inpainting proper are typically separated in classical shape reconstruction pipelines. However, when both local and global properties are considered, it may be of added value to treat the two steps jointly. This aspect is further elaborated in Chapters 3, 4, 5, and 9.

2. *Skeletonization choice:* The success of shape reconstruction based on skeletal descriptors strongly relies on the quality of these descriptors. This quality involves first and foremost 'functional' properties thereof, such as accuracy, centeredness, thinness, and detail control, which are well known in the skeletonization literature [59]. However, 'non-functional' properties of skeletonization methods, such as computational scalability, robustness to noise, ease of use, and availability are equally important [191]. While all above properties are well-known in skeletonization research and practice, it is far from clear which methods optimally comply with them – or, in simple words, which skeletonization method should one use as the building brick of future shape-processing applications. This aspect is further elaborated in Chapters 6 and 7.

3. *Shape representation:* Shape analysis and processing knows two major types of representations. Boundary representations (*b-reps*) describe only the interface separating the interior of a solid shape from the embedding space, *e.g.* as a point cloud or mesh. Volumetric representations describe every single (sample) point of the embedding space in terms of it being exterior, on the boundary, or interior to the shape. Both representations have their specific advantages and limitations, and these aspects naturally propagate to the construction and usage of shape skeletons. Choosing one of the above representations has far-reaching consequences in terms of the types of methods that can be further used to analyze and/or process shapes. This aspect is further discussed in Chapters 2, 6, and 7.

1.5 STRUCTURE OF THE THESIS

The above-mentioned research question and related challenges are treated in the remainder of this thesis as follows. At a high level, Chapter 2 covers related work concerning both 2D and 3D shapes. Chapters 3-5 cover restoration techniques for 2D shapes. Chapters 6-9 cover restoration techniques for 3D shapes. A detailed overview of each chapter is given next.

Chapter 2 presents related work in the two main domains covered by this thesis – shape reconstruction (with a focus on inpainting methods) and shape skeletonization. From this overview, it can be seen next that few inpainting methods jointly use local and global information for defect detection and reconstruction, leaving thus an open area for research. Separately, our survey of skeletonization methods shows that their practical application, *e.g.* in our context of shape reconstruction, is far from trivial, since most such methods have many complex assumptions and limitations.

Chapter 3 starts exploring the possibilities of global techniques for image restoration using a relatively simple example – the elimination of artifacts from low-resolution 2D facial images. The global aspect of the proposed method resides in the usage of statistical properties, computed over an entire collection of images, to both detect the damaged regions and restore these. The proposed method is shown to deliver good results for artifact elimination for a wide range of facial images when compared with three other well-known inpainting techniques, both in qualitative terms and also in

terms of structural image quality metrics.

Chapter 4 continues the exploration of 2D image restoration within a more specialized context. Specifically, we study here the joint detection and elimination of so-called *structural gaps* present in binary 2D images, *i.e.* thin and elongated cracks that (nearly) fragment the input image into various components. We show how such gaps can be robustly and efficiently detected and classified as being different from small-scale detail present on the shape boundary, and also eliminated by reconstruction. Key to detection, classification, and recostruction is the usage of 2D shape skeletons, which allow an elegant way to reason about the type of gaps with respect to the shape structure, and also a simple way to fill the desired gaps in a controlled manner. The added-value of our gap-detection-and-filling technique is demonstrated on several use-cases including robust 2D image segmentation, image denoising, and automatic hair removal from dermoscopy images.

Chapter 5 zooms in on a specific application of image restoration: digital hair removal (DHR). This application is a very good use-case of our overall proposal of skeleton-based inpainting, for several reasons. First, DHR is well known to be a hard and important problem in dermoscopy, specifically the preparation of acquired images for further automated diagnosis. Secondly, the characteristics of the artifacts to be detected and removed (hairs) are complex, spanning a mix of structural, geometrical, and color-related attributes. Finally, many images and techniques are available for testing and comparison. We propose here a new method for DHR that jointly detects and eliminates hairs by extending our earlier gap-detection technique (Chapter 4) to treat 2D skeletons obtained from a full threshold-set decomposition of an input grayscale image. Qualitative and quantitative comparison of our method with five known DHR techniques shows the advantages of our proposal.

Chapter 6 moves our exploration of skeleton-based shape restoration to the 3D domain. As outlined earlier in Sec. 1.4, many skeletonization methods exist in the literature, but a detailed quantitative and qualitative comparison of their performance with respect to several desirable quality criteria is still missing. On the other hand, such a comparison is critical for the selection of an optimal skeletonization method to be used next in our shape restoration context. Chapter 6 presents the first part of such a comparison, where we consider curve skeletons computed by mesh-based methods, which are a relatively new development in the skeletonization arena, as compared to classical voxel-based skeletons. We compare six such mesh-based methods and one classical voxel-based method from the perspective of six accepted quality criteria. The obtained results show that mesh-based methods are not always superior to voxel-based methods from all considered viewpoints, and thus likely not the desired candidate for us to use next.

Chapter 7 extends the comparison started in Chapter 6 to consider both 3D surface and curve skeletons. We restrict our study to voxel-based methods, as mesh-based methods did not show a strong superioirity in our earlier exploration (Chapter 6). Moreover, voxel-based methods naturally fit the context of volumetric shape restoration using inpainting methods. In this chapter, we compare four surface and six curve

skeletonization methods that operate on voxel shapes, based on the same set of criteria used in our earlier comparison. Additionally, we consider computational scalability as an important criterion. Finally, we propose several techniques for quantitative comparison of curve and surface skeletons. Overall, this chapter presents, to our knowledge, the broadest recent comparison of 3D curve and surface skeletonization methods, with added value beyond our specific shape restoration context.

Chapter 8 follows from the finding of the joint work in Chapters 6 and 7: There is no method for computing 3D curve and surface skeletons that optimally satisfies our desirable criteria required for practical usage. Given this, we propose here a new voxel-based skeletonization method that aims to unify design principles present in the most successful skeletonization methods covered by our study. The resulting method, dubbed an 'unified framework' for planar, curve, and surface skeletonization, combines the principles of contraction, boundary collapse, and advection present in many existing methods in a single advection-based model. Thereby, the presented framework allows computing all known types of medial skeletons by a single formulation, and also in a multiscale fashion. Presented results show that skeletons computed by our method comply well with the desirable quality criteria outlined in Chapters 6 and 7, yield very similar results to several well-known skeletonization methods, and are efficient to compute for large 3D volumes.

Chapter 9 covers our last part of exploring the use of skeletons for shape restoration by considering the elimination of gaps and cracks in 3D volumetric shapes. Two methods for this use-case are presented. The first method adapts the 2D gap-filling approach presented in Chapter 4 to efficiently treat a 3D grayscale volume in a per-slice manner. The method is applied to the task of wire artifact removal from Cone Beam Computer Tomography (CBCT) volumetric images, where we show that its application delivers significantly less noisy 3D reconstructions of the scanned shapes. The second method extends the 2D gap-filling approach in Chapter 4 to use 3D surface skeletons to remove more complex cracks and indentations present in generic 3D shapes. The method is compared with classical morphological techniques for gap elimination (closing), showing a better capability to both preserve details and remove the undesired gaps.

Chapter 10 concludes the thesis by revisiting the original research question in the light of our obtained results. Both strong aspects and challenging aspects of the methods proposed in this thesis are reviewed, and directions for future research in the field are outlined.

2 RELATED WORK

As outlined in Chapter 1, Sec. 1.4, our main research question concerns the potential of using skeletal descriptors to perform defect restoration on 2D and 3D shapes. As such, related work naturally segments into work on digital image restoration and inpainting (Sec. 2.1) and shape skeletonization (Sec. 2.2). Both these aspects are detailed next.

2.1 DIGITAL IMAGE RESTORATION AND INPAINTING

The restoration of works of art such as painting, drawing and geometric patterns has always been a fascinating occupation of many professionals who helped recover the fragments of the past to know, understand and appreciate the history of the world. Manual restoration requires a specialized formation and time [131]. Fortunately, with advances in technology and science, several techniques have emerged that help users restore digital images.

One important class of such techniques is globally known under the name *inpainting* [23, 26]. The technique inherits its name from the manual process of restoring small-scale damaged areas in oil or canvas paintings by artists who incrementally close these areas by 'painting inwards' from their boundaries, by reproducing the colors of undamaged image areas close to these boundaries. In the last 15 years, several so-called *digital inpainting* techniques have emerged. According to Tauber et al [222], *digital image inpainting refers to any methods that fill-in holes of arbitrary topology in images so that they seem to be part of the original image*. These techniques, also called image completion or image fill-in [26], aim to simulate the process of inpainting performed by manual artists by computerized methods, so as to automatically recognize and restore defect areas. Accordingly, digital inpainting can be defined as follows: Given an image $I : \Omega \subset \mathbb{Z}^2 \rightarrow \mathbb{R}^d$, having either grayscale ($d = 1$) or color ($d = 3$) pixels, define a damaged area $D \subset \Omega$ where restoration needs to take place. Next, inpainting can be seen as an operator $\mathscr{I}$ which takes (I, D) as input and outputs a restored image $I^r : \Omega \subset \mathbb{Z}^2 \rightarrow \mathbb{R}^d$, so that

a) $I^r = I$ over $\Omega \setminus D$;

b) I^r preserves, over D, the salient features of I at points in $\Omega \setminus D$ close to ∂D.

Digital inpainting has many applications, such as creating restoration previews of digital scans of physical paintings, prior to the actual physical restoration [190]; creating digital restorations of works of art which do not admit physical restoration [23]; removing undesired artifacts like text and logos from digital content [229]; and general-purpose editing of digital content to create artistic effects [64]. Following the formal definition of inpainting outlined earlier, practical inpainting tools work in two steps. First, the damaged region D of the input image I is selected, using automatic defect

detection or manual selection. Secondly, the desired inpainting operator $\mathscr{I}$ is applied on (I,D) to yield the restored image $I^r = \mathscr{I}(I,D)$.

Given the above, a survey of inpainting methods should cover two separate aspects: (a) the detection of the inpainting domain D; and (b) the definition of the inpainting operator $\mathscr{I}$. However, the largest part of inpainting methods in existence focus only on aspect (b) above, *i.e.*, assume that the inpainting domain D is given. As a recent survey on inpainting [26] puts it: "The region [D] is always given by the user, so the localization of [D] is not part of the inpainting problem". The same survey notes that the vast majority of inpainting methods treat D as a hard constraint, *i.e.* preserve the input image I over $\Omega \setminus D$. As such, our next discussion of inpainting methods focuses solely on the definition of $\mathscr{I}$.

Following Bertalmio *et al.* [26], inpainting methods can be classified in three groups: patch-based methods, sparse methods, and differential/variational methods. These method classes are discussed next.

2.1.1 *Patch-based inpainting methods*

Patch-based methods are characterized by the fact that they synthesize the values of image pixels in D in *blocks*, rather than individually. Also, for the synthesis process, blocks of pixels outside D are used. This allows replicating various types of characteristics of the undamaged image $\Omega \setminus D$ inside D. One of the earliest inpainting methods in this class proceeds precisely like this [76]: The area D is filled in increasing distance from ∂D. For each pixel $\mathbf{x} \in \partial D$, let $P(\mathbf{x})$ be a small square block of pixels in $\Omega \setminus D$ centered at $\mathbf{x}$. The color $I(\mathbf{x})$ is set to the color of the pixel $\mathbf{y} \in \Omega \setminus D$ whose block $P(\mathbf{y})$ is most similar to $P(\mathbf{x})$.

The patch-based inpainting idea is extended by the so-called exemplar-based inpainting [64]. The key changes proposed pertain to prioritizing the inpainting of pixels $\mathbf{x} \in \partial D$ that correspond to high-gradient regions, such as edges, in $\Omega \setminus D$ ending at ∂D, so that such edges are better preserved inside the region to inpaint; and inpainting entire blocks of pixels rather than individual pixels, which makes the method preserve texture patterns better and also computationally faster. A further speed-up of patch-based inpainting was proposed by Ashikhmin [12], by reducing the search space for the most-similar block $P(\mathbf{y})$ to get the color to inpaint for a pixel $\mathbf{x} \in \partial D$ to pixels $\mathbf{y}$ close to $\mathbf{x}$. This essentially makes the method linear in the number of pixels $\|D\|$ to inpaint.

More recent exemplar-based methods use a search-and-replace, rather than a texture-synthesis, strategy. In other words, the information to derive the inpainting from is searched in a (large) set of similar images, rather than outside the damaged region D of the image to inpaint. For instance, Hays and Efros [99] proposed a method for so-called 'scene completion' where the user first (manually) segments areas D to be inpainted in an image, and next the method searches images similar to areas surrounding D in a large databases containing millions of photographs taken from the Internet. The selected region D is next synthesized by mixing information from the most similar images found in the database. Whyte *et al.* [239] propose a similar method which uses a more advanced set of transforms to map information from similar images in the database to the region D to inpaint, including geometric

and photometric registration. These additional transforms ensure a better fusion of the information retrieved from the database with the local context of the image to inpaint. A similar exemplar-based method to [99] is proposed by Li *et al.* [144]. Like [99], the most similar image to the context $\Omega \setminus D$ of the image to inpaint is sought in a database, yielding a raw inpainting result. This result is next adapted to the local context $\Omega \setminus D$ by filtering and color-transfer functions.

In recent years, many other types of exemplar-based inpainting methods have been proposed, aiming to provide a better propagation of salient image structures from the undamaged area $\Omega \setminus D$ to the damaged area D. For example, Du *et al.* [74] extract such information using a hierarchical segmentation of the undamaged area, and propagate it inside the damaged area using the variational Mumford-Shah-Euler model for image-attribute propagation [78], discussed further in Sec. 2.1.3. Interestingly, the same method has also been used for image segmentation. This method gives better results than other inpainting methods for images having a large number of detail edges. Separately, Zhao and Li [145] separate inpainting in a pass that detects salient structures using the image wavelet transform, propagate these inside the damaged region using curve fitting, and fill gaps between such structures using classical local texture synthesis. Other exemplar-based methods propose different heuristics for the priority of filling image patches in the damaged region, as well as searching for the best-matching patch in the undamaged region to gather information from [32, 53, 118].

2.1.2 *Sparse inpainting methods*

Following the success of the first patch-based inpainting methods such as [12, 64, 76], several researchers have considered the problem of inpainting as an optimization problem – or in other words how to search the best-matching patches to fill the damaged region in a 'dictionary' of patches that describe the undamaged image portion. The sparse aspect of such methods relates to the definition of a compact dictionary whose patches can efficiently describe the damaged area in visually plausible ways.

In the above context, Elad *et al.* [77] proposed an image decomposition model into geometric and texture components, similar in spirit to the way images can be segmented in so-called superpixels, or zones having strong internal coherence [147]. As such, their model can be used for both image segmentation and image inpainting. The above model has been extended to image denoising [4], small-hole inpainting [153], and video-sequence inpainting [154].

Overall, sparse methods can achieve excellent results, as shown *e.g.* in [154]. However, they are more complex to implement than basic patch-based methods. Also, their increased quality comes with the cost of additional parameters that one has to tune to obtain the desired optimal results.

2.1.3 *Differential/variational inpainting methods*

In contrast to patch-based and sparse inpainting methods, differential and variational inpainting methods take a very different approach: Instead of iteratively searching for a *local* way to inpaint a small image portion (a pixel or a patch), they look for *global* formulations that restore all damaged pixels in D based on solving a partial differential

equation (PDE) or a variational problem. In both cases, the equation to solve captures both the fact that the restored image I^r should match the original image I outside the damaged area (boundary conditions), and the fact that the restored image should obey various local smoothness and continuity properties.

Arguably the simplest way to restore small damaged areas of an image is by using local filtering, which essentially blur or smooth out information from the undamaged area inwards in the damaged area. An early example hereof is presented by Ogden *et al.* [167] who use Gaussian filter pyramids for this purpose. However, given the isotropic and smoothing nature of such filters, this type of restoration can only handle (very) small-scale damaged areas, like holes. A similar approach is taken in [169] where a simple 3×3 filter is used for speed.

More advanced approaches aim to capture higher-order image features present in the undamaged area and extrapolate these smoothly within the damaged area. For instance, Mason and Morel propose to extrapolate the isophotes (isolines of constant luminance) of a grayscale image as elastic curves [157]. The model can also be used for color images by applying it independently on the three channels of the image represented in *e.g.* Lab color space.

The first use of the term 'digital image inpainting' was proposed by Bertalmio *et al.* [23]. In this approach, the image smoothness information, estimated by the image Laplacian, is propagated along the isophotes directions, estimated by the image gradient rotated 90 degrees. A separate study shows that the above type of inpainting is closely related to an incompressible Navier-Stokes flow that would 'move' grayscale values along isophotes [24]. A related approach, called the *total variation* (TV) model, uses an Euler-Lagrange equation coupled with anisotropic grayscale diffusion to better maintain the directions of the isophotes inside the inpainted area [46]. The TV method is further enhanced by the Curvature-Driven Diffusion (CCD) model [47], by driving diffusion along the isophotes directions and thus allowing one to inpaint thicker regions with high quality. While all above methods produce high-quality inpainting results, implementing the respective PDE solvers is relatively complicated and also can be computationally expensive. Aiming to alleviate such issues, Telea [229] proposed an inpainting method based on the fast marching method (FMM) [198]. In this approach, pixels in D are inpainted in increasing distance order to ∂D, similar to how a painter would do. The color of an inpainted pixel $\mathbf{x} \in \partial D$ is synthesized by an anisotropic weighting of pixels in $\Omega \setminus D$ close to $\mathbf{x}$, so as to preserve the isophotes' directions as well as possible. This method is fast ($O(\|D\| \log \|D\|)$, simple to implement, and gives good results for complex-shaped, crossing, but relatively thin damaged regions, and has been as such included in the popular OpenCV image-processing toolkit [170]. However, for thicker damaged regions, the method produces significantly more blurring than more advanced differential/variational methods, and than most exemplar-based methods.

Variational methods for image inpainting work by defining an integral cost function that tries to both preserve I^r close to I on the undamaged image area and also ensure various desirable smoothness properties for I^r over the damaged area. For example, Esedoglu *et al.* combine the well-known Mumford-Shah segmentation model for images with Euler's elastica model for (isophote) curves to cast inpainting in a variational setting [78]. However, this approach leads to complex high-order differential equations whose robust and efficient numerical solving is delicate. Another approach

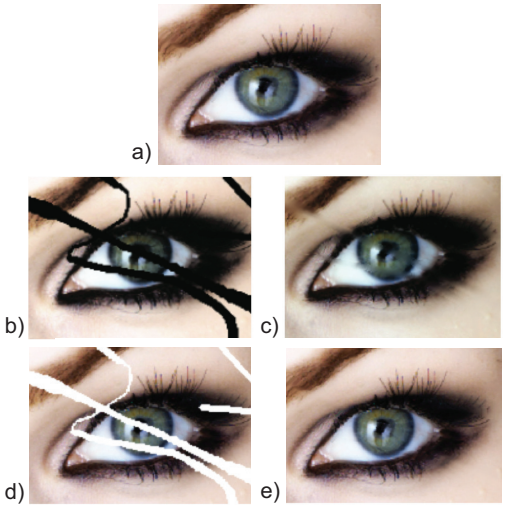


Figure 2.1: Comparison of inpainting methods. (a) Original image. (b,c) Damaged image and restoration by the method in [229]. (d,e) Damaged image and restoration by the method in [35].

is presented by Tschumperlé [233], where the damaged area is first roughly filled in by simple advection of colors on ∂D, and next refined by anisotropic diffusion that incrementally propagates the structure (texture) of image outside D into the damaged area. Separately, Bornemann and März [35] model inpainting by advecting image colors from the undamaged area into the damaged area D along the integral curves of a so-called coherence vector field that extrapolates the directions of the image gradient into D. This method yields very high quality results for damaged regions of moderate thickness at a very low computational cost, roughly in line with much simpler (and lower-quality) methods such as [229] (see Fig. 2.1).

Overall, differential/variational methods excel in inpainting regions with guaranteed smoothness and continuity properties for various relevant image aspects, such as isophotes, and color and luminance gradients. Compared to patch-based and sparse methods, they also come with solid mathematical underpinning which allows one to reason about the properties of the reconstructed image. However, globally speaking, differential/variational methods are far less able to reconstruct texture-rich areas of significant thickness, as compared to patch-based and sparse methods. As such, more recent methods attempt to combine the strengths of the above method classes by using various heuristics [25, 42, 73, 130, 200].

Summarizing the discussion on image inpainting, we can say with reasonable confidence that the current state-of-the-art offers a rich palette of methods that suit most application contexts in terms of the desired characteristics of the restored region, priors of the input image and/or the damaged areas, and speed *vs* implementation complexity of the method. However, as mentioned in the beginning of Sec. 2.1, most such methods focus purely on the restoration problem, and less on the automatic detection of damaged areas to restore next.

2.1.4 *3D inpainting*

Three-dimensional (3D) inpainting aims, in essence, to cover similar goals to classical 2D image inpainting: Given a shape $\Omega \subset \mathbb{R}^3$, 3D digital inpainting aims at restoring parts of the shape's surface $\partial\Omega$ or shape volume Ω so that the modified (restored) portions appear to seamlessly fit with the surrounding parts of the restoration domain.

The vast majority of 3D inpainting methods are confined to the first category outlined above, *i.e.*, aim to restore parts of a shape's surface $\partial\Omega$. Such methods work analogously to their 2D image-inpainting counterparts described in Sec. 2.1: Given the shape surface $\partial\Omega$, a so-called inpainting domain $D \subset \partial\Omega$ is first determined, so that it includes the areas to be restored (inpainted). Next, the inpainting method of choice is executed so as to replace D by surface elements which are similar with regions of $\partial\Omega$ close to D.

One of the most typical applications for such methods is to repair 3D surfaces by filling holes. In this setting, the inpainting domain D cna be seen as carving a number of holes on the 3D surface $\partial\Omega$. As such, these methods are also known under the name of *hole filling* methods. Applications of hole filling are numerous, ranging from the generation of watertight 3D meshes from meshes containing cracks, removal of meshing defects (mesh fairing) [54, 55] and 3D surface reconstruction from mesh pieces, such as generated during the reconstruction of surfaces from unstructured point clouds [134]. Besides 3D surface hole filling, 3D inpainting is also used in the context of view interpolation in image-based rendering, multi-view construction of stereo models of scenes, and filling cracks and repairing imperfections in volumetric 3D shapes.

3D surface hole-filling methods typically work on mesh-based discretizations of $\partial\Omega$ [21, 22, 244]. Such methods perform inpainting using either implicit surface representations [65, 82, 114] or explicit surfaces [75]. Other methods perform the 3D surface repairing by reconstructing the surface stitching a sequence of 2D images (views) of the damaged surface [96]. Kawai *et al.* [121–123] present some methods for surface and texture completion on 3D surfaces. Such methods work by minimizing various energy functions for shape and texture similarity [122, 123] or analyzing the principal curvature of the 3D shape to inpaint [121]. There are many further methods for surface completion in 3D mesh datasets [43, 115, 237]. However, for our goal of *volumetric* restoration, *i.e.* the identification and removal of hole-like defects that affect a 3D densely-sampled volume, much fewer methods exist. In this specific context, we have found only one such method proposed by Janaszewski *et al.* [111], which treats the detection and filling of tunnels in 3D shapes. However, besides tunnels, many more types of hole-like defects can exist in 3D volumes, *e.g.*, cracks, cuts, indentations, and all such defects can have a wide spectrum of distributions of size and noisiness. As such, the area of 3D inpainting is clearly open for additional research.

2.2 SHAPE SKELETONIZATION

Skeletons, also called medial axes or medial descriptors, have been introduced more than 50 years ago by Blum as compact and simple descriptors able to capture the geometry and topology of a 2D shape [33, 34]. Given these properties, skeletons have

subsequently emerged as being efficient and effective tools for a wide range of shape analysis and processing operations, including shape recognition, matching, and retrieval, path planning, metrology, and computer animation [203]. Skeletons are computed from 2D and 3D shapes by various methods known globally under the name of *skeletonization*. In the following, we first overview the main definitions and properties related to skeletons (Sec. 2.2.1), followed by a survey on skeletonization methods (Sec. 2.2.2). Given our focus on 3D shape processing, the following discussion will be naturally focused on definitions, properties, and algorithms related to skeletons of 3D shapes.

2.2.1 *Definitions*

Given a 2D or 3D binary shape $\Omega \subset \mathbb{R}^{\{2,3\}}$ with boundary $\partial\Omega$, a convenient way to introduce skeletons is to first define the shape's distance transform $DT_{\partial\Omega} : \mathbb{R}^{\{2,3\}} \rightarrow \mathbb{R}^{+}$

$$DT_{\partial\Omega}(\boldsymbol{x} \in \Omega) = \min_{\boldsymbol{y} \in \partial\Omega} \|\boldsymbol{x} - \boldsymbol{y}\|. \tag{2.1}$$

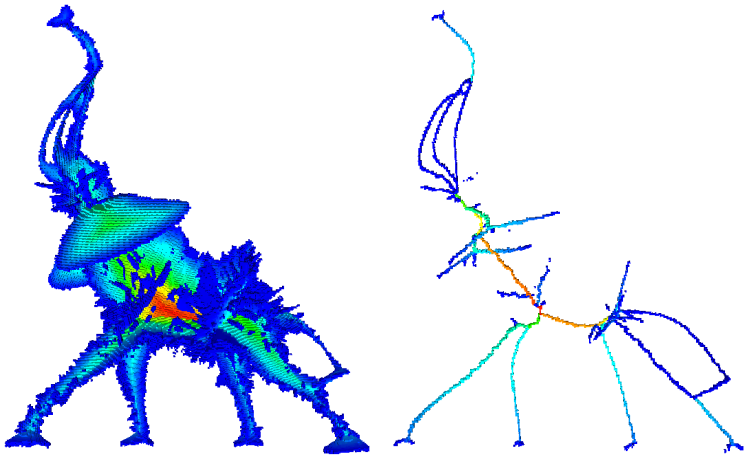
The so-called *surface skeleton* of Ω is next defined as

$$S(\Omega) = \{\boldsymbol{x} \in \Omega \,|\, \exists\, \boldsymbol{f}_1, \boldsymbol{f}_2 \in \partial\Omega, \boldsymbol{f}_1 \neq \boldsymbol{f}_2, \|\boldsymbol{x} - \boldsymbol{f}_1\| = \|\boldsymbol{x} - \boldsymbol{f}_2\| = DT_{\partial\Omega}(\boldsymbol{x})\} \tag{2.2}$$

where $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are the contact points with $\partial\Omega$ of the maximally-inscribed ball in Ω centered at $\boldsymbol{x}$ [93, 187], also called *feature transform* (FT) points [216] [102] [109] or *image points* [203]. In the case that the shape is two-dimensional, *i.e.* $\Omega \subset \mathbb{R}^2$, definition 2.2 yields the classical 2D skeleton, also called *medial axis* [61, 203]. The tuple $(S(\Omega), DT_{\partial\Omega})$ is known as the medial axis transform (MAT). If $S(\Omega)$ and $DT_{\partial\Omega}$ are exactly computed, the MAT allows an exact reconstruction of the shape Ω from it, which makes the MAT a *dual* representation, as opposed to the classical boundary representation of shapes.

Surface skeletons consist of several 3D manifolds with boundaries which meet along a set of Y-intersection curves [48, 66, 141]. Figure 2.2a shows an example of surface skeleton computed for an elephant shape. Surface skeletons can be computed by voxel-based or mesh-based methods [18, 36, 102, 171, 217]. Analogously, 2D medial axes consist of several 2D curves which meet along a set of so-called junction points or bifurcations. Consequently, the structure of 2D medial axes is considerably simpler than the structure of 3D surface skeletons. This has led to the emergence of tens of methods that are able to compute 2D medial axes in real-time for complex 2D shapes represented as polygonal contours [168] or, alternatively, as binary images [61, 228]. In contrast, until recently, computing 3D surface skeletons has been considered a complex and computationally expensive problem. However, as we shall also discuss later on, recent research has made it possible to compute 3D surface skeletons with comparable degrees of accuracy and computational scalability as earlier methods for computing 2D skeletons [109].

Besides surface skeletons, 3D shapes admit a second type of medial structure, typically called the *curve skeleton*. In contrast to surface skeletons, curve skeletons are loosely defined as 1D structures "locally centered" within the input shape Ω. Fig-



(a) (b)

Figure 2.2: Skeletons of a 3D shape: (a) surface skeleton; (b) curve skeleton. Both skeletons are computed by the method presented further in this thesis in Chapter 8.

ure2.2b shows the curve skeleton corresponding to the same elephant shape from which the surface skeleton in Fig. 2.2a has been extracted. Given the above looseness in defining what local centeredness precisely means, curve skeletons admit a large variety of both implicit and explicit definitions [58]. As a consequence, besides them being 1D piecewise-curve structures and being homotopic to the input shape, there are no other formal properties that emerge from their definition and that are, thus, respected by all existing curve-skeletonization methods out there. The lack of a unanimously accepted formal definition has also led to many methods which compute curve skeletons following not necessarily identical definitions. This makes it hard to analytically compare, and reason about, the properties of the produced curve skeletons. For instance, a formal relationship between the surface and curve skeletons of a given 3D shape is still not unanimously accepted. For instance, although it is commonly accepted that the curve skeleton should be centered within the surface skeleton of the same shape, only few skeletonization methods use and/or enforce this property [187, 227].

In the above context of providing a formal curve-skeleton definition, Dey and Sun proposed one of the first analytic definitions of curve skeletons based on the medial geodesic function (MGF), where the curve skeleton is defined as the locus of points having at least two equal-length shortest geodesics on $\partial\Omega$ between their feature points [70, 178]. Reniers *et al.* extended the MGF to regularize curve skeletons by assigning each skeleton point an importance equal to the area bounded by such geodesics [187], inspired by the so-called 2D collapse metric [168, 228]. A GPU implementation of the above metric was presented in [109].

2.2.2 *Overview of Skeletonization Methods*

Surface and curve skeletons can be computed by thinning, field, and mesh-based methods (see *e.g.* [58, 191]). Such methods can be further classified based on whether they use a volumetric (voxel-based) sampling of the input shape Ω or a boundary (typically mesh-based) sampling of the shape's surface $\partial\Omega$. A brief review of skeletonization methods follows next.

Thinning: Thinning methods remove $\partial\Omega$ voxels (or pixels in 2D) while preserving connectivity until a minimal structure –the skeleton – is left [18, 62, 171, 179]. Although simple and fast, thinning can be sensitive to Euclidean transformations. Tools from mathematical morphology [197] were among the first used to compute curve skeletons by thinning. The residue of openings, based on Lantuéjoul's formula [136], usually leads to disconnected skeleton branches, whereas methods based on homotopic thinning transformations [29, 136, 158, 171] yield connected skeletons. Constraining thinning by distance-to-boundary order [11, 179, 219] or flux-order [175] further enforces centeredness. Recent distance-ordered thinning methods have shown their ability to compute high-quality surface and curve skeleton of 3D voxel-based shapes representations [11]. However, just as the field methods (discussed next), thinning methods are challenged by the need to maintain an expensive dense volumetric sampling of the input shape. This inherently limits their scalability and computational efficiency for large, high-resolution, 3D shapes.

Field methods find $S(\Omega)$ along singularities of $DT_{\partial\Omega}$ or related more general fields [91, 102, 125, 142, 189, 228, 236] and can be efficiently done on GPUs [41, 216, 217]. General-field methods use fields smoother (with fewer singularities) than distance transforms [5, 13, 58, 98], and thus are more robust for noisy shapes. Siddiqi *et al.* find the skeleton as the non-zero divergence locus of $\nabla DT_{\partial\Omega}$ [204]. However, $\nabla\cdot(\nabla DT_{\partial\Omega})$, with $\nabla\cdot$ the divergence operator, can be non-zero also at non-skeletal points. Torsello and Hancock correct this for a more accurate 2D skeleton detection by a momentum conservation principle $\nabla\cdot(\rho\nabla DT_{\partial\Omega})=0$, where ρ is the mass density on the evolving boundary $\partial\Omega$ [14]. Rossi and Torsello extend this idea to compute 3D surface skeletons [188]. However, this method does not compute curve skeletons and does not model the curve-surface skeleton relationship. Field methods can also compute 3D curve skeletons by backprojecting 2D skeletons of 2D projections [150] or axis-aligned slices [228] of the shape back into 3D. Field methods have been proposed for both voxel and mesh shapes.

Mesh-based methods have emerged as a consequence of the high costs implied by voxel-based methods. As mentioned earlier, voxel-based methods typically require significant resources to store and process the large voxel volumes required to capture the fine details of complex 3D shapes. To be used on 3D meshes, such methods require a costly voxelization step. Mesh-based methods address these cost issues by working directly on a mesh representation of $\partial\Omega$.

Mesh-based methods include Voronoi diagrams [72] and subsets thereof [8], mesh contraction in normal direction [15, 41, 146, 220], mean-shift-like clustering [105], and union-of-balls approaches [109, 151, 159]. Such methods use meshed shape rep-

resentations and thus scale well to handle high-resolution models [109, 151]. Early mesh methods used Voronoi diagrams to compute polygonal skeletons [71]. Amenta *et al.* compute the Power Crust, an approximation of a surface and its medial axis by a subset of Voronoi points [8]. Other methods use edge collapses [146], starting from a mesh segmentation [120]. Surface skeletons can be extracted from oriented point clouds [109, 151] or polygon meshes [141, 161] by searching for maximally inscribed balls tangent at at least two shape points. Curve skeletons can be extracted from point clouds as centers of cloud projections on a cut plane which optimizes for circularity [220].

Contraction techniques are a separate, and more recent, subclass of mesh-based skeletonization methods. Like field techniques, they evolve $\partial\Omega$ under various types of normal flows, effectively collapsing it onto the surface-or-curve skeleton. Methods using a (constrained) Laplacian contraction by mean curvature flow deliver high-quality curve skeletons [15, 40, 52], or even 'meso skeletons' mixes of surface and curve skeletons [221]. Au *et al.* shrink the mesh via Laplacian smoothing until its volume gets close to zero, followed by an edge-collapse (to extract the 1D curve skeleton) and a re-centering step (to correct shrinking errors) [15]. Cao *et al.* extend this idea to extract curve skeletons from incomplete point clouds [41]. The ROSA method defines, and extracts, curve skeletons using rotational, rather than positional, symmetry: $\partial\Omega$ is cut with planes, and curve-skeleton points are found as the centers of planes which minimize the variance between the plane's normal and $\partial\Omega$ normals along the cut curve [220]. Sharf *et al.* reverse the contraction direction: They find the curve skeleton as the centers of a set of competing fronts which evolve to approximate the input surface [201]. A similar method is presented by Hassouna and Farag [98]. Telea and Jalba define, and extract, curve-skeletons by contracting the surface skeleton $S(\Omega)$ (computed as in [151]) inwards, along the gradient of the 2D distance transform of $\partial S(\Omega)$, *i.e.* define the curve-skeleton as the result of a two-step skeletonization [227].

Given their computational scalability and high-quality results, mesh contraction techniques have emerged as likely candidates for the best 3D surface-and-curve skeletonization methods in existence [15, 109, 159, 221]. However, in the same time, other authors present strong arguments that vote in favor of recent voxel-based skeletonization methods [11, 191]. To the present moment, a very limited number of comparison studies exist that pitch voxel-based methods against mesh-based methods to see which method-class is able to produce better skeletons from the perspective of a number of desirable criteria. As such, the question of which (class of) methods produces better skeletons is still an open one.

Multiscale skeletons: Apart from the method itself used to extract curve and/or surface skeleton, a separate aspect is of crucial importance in skeletonization. This aspect is related to the fact that the skeletonization operation, seen as a function that acts on the space of 3D shapes with results in the space of 3D (surface or curve) skeletons, is not Cauchy or Lipschitz-continuous with respect to changes of its argument. In simple terms: even very small changes to a shape Ω, with respect to *e.g.* a Hausdorff distance metric, can yield very large changes of $S(\Omega)$, with respect to the same distance metric. This phenomenon is known under several names, such as the instability of skeletonization, the emergence of so-called 'spurious' branches, and the lack of robustness of skeletons. In practice, this issue is of high importance: Small changes can

easily occur on 3D shape surfaces, caused *e.g.* by discretization, shape acquisition, or numerical issues. As even tiny such changes can cause massive modifications to the emerging skeletons, ways are needed to control the skeletonization process.

The process of removing the effect of the above-mentioned small-scale shape changes in the resulting skeletons is known under the generic name of *regularization*. Formally speaking, regularization can be seen as an operator R that takes a skeleton $S(\Omega)$ as input and delivers a 'clean' skeleton $\tilde{S}(\Omega)$, so that $R(S(\Omega))$ is continuous with respect to small changes in Ω in the sense mentioned above. Regularization is typically done by first defining a so-called *importance measure* $\rho : \Omega \rightarrow \mathbb{R}^+$ which is large for important skeletal branches and small for branches caused by noise or small-scale details on $\partial\Omega$. Hence, upper-thresholding ρ delivers the desired regularized skeleton $\tilde{S}(\Omega)$ [66, 199].

Following several authors [109, 159, 187], one can distinguish between local and global importance measures. *Local measures* cannot separate locally-identical, yet globally-different, contexts (see *e.g.* [187], Fig. 1). Thresholding local measures can disconnect skeletons. Reconnection needs extra work [155, 176, 204, 217], and makes pruning less intuitive [199]. Local measures include the angle between the feature points and distance-to-boundary [8, 86, 217], divergence-based [36, 204], first-order moments [189], and points where $\nabla DT_{\partial\Omega}$ is multi-valued [214, 215]. Leymarie and Kimia topologically simplify point-cloud skeletons to capture Y-intersection curves and skeleton sheet boundaries in medial scaffolds [141]. A good survey of such methods is given in [203].

Global measures, in contrast, monotonically increase from the skeleton boundary ∂S_Ω inwards. Thresholding them yields connected skeletons which capture skeleton details at a user-given scale. Miklos *et al.* approximate shapes by unions of balls (UoB) and use UoB medial properties [94] to simplify skeletons [159]. Dey and Sun introduce the medial geodesic function (MGF), equal to the shortest-geodesic length between feature points [70, 178]. Reniers *et al.* [187] extend the MGF for surface and curve skeletons using geodesic lengths and surface areas between geodesics, respectively, inspired by the so-called collapse metric used to extract multiscale 2D skeletons [79, 168, 228]. A fast GPU implementation of this extended MGF is given in [109].

The MGF and its 2D collapse metric counterpart have an intuitive geometric meaning: They assign to a skeleton point $\mathbf{p}$ the amount of shape boundary that corresponds, or 'collapses' to, $\mathbf{p}$ by some kind of boundary-to-skeleton mass transport. Skeleton points $\mathbf{p}$ with low metric values correspond to small-scale shape details or noise; points $\mathbf{p}$ with large metric values correspond to large-scale shape details. This allows an easy simplification of the skeleton: Thresholding by a value τ eliminates all skeleton points which encode less than τ boundary length or area units. If the collapse metric monotonically increases from the skeleton boundary to its center, thresholding delivers a set of connected and nested skeleton approximations, also called a multiscale skeleton [70, 79, 187, 228]. As such, global importance measures do not serve only the purpose of skeleton *regularization* (by elimination of spurious noise-induced skeleton branches), but also the more complex purpose of skeleton *simplification*.

While global importance measures do, thus, serve the joint and important purposes of regularization and multiscale simplification, they also come with several challenges. Firstly, computing such measures for 3D surface and curve skeletons is quite involved

and computationally expensive, even when using recent GPU methods [109]. More importantly, from a theoretical perspective, is the fact that there is no *unified* way to define multiscale skeletal importance for 2D medial axes, 3D surface skeletons, and 3D curve skeletons. Indeed, while several methods allow computing such multiscale importance metrics, as discussed above, and while all such methods essentially use an intuition based on the collapse of mass transported from $\partial\Omega$ to the skeleton by some sort of advection process, there is no formal definition and computational model for simulating such a collapse process. As such, the various existing global importance metrics used for skeleton regularization share the same property mentioned earlier for curve skeletons – the lack of an unified formulation able to be used to derive properties from it.

2.2.3 *Properties*

For the vast majority of shapes, skeletons cannot be computed analytically, for a number of reasons. First and foremost, typical shapes available in application domains where skeletonization is needed do not come in an analytic form, *e.g.*, described in terms of explicit or implicit functions, but as sampled representations. Examples hereof are 2D binary images and 3D binary (voxel) volumes, 2D polygonal contours and 3D polyhedral meshes, and 3D unstructured point clouds. Sampling introduces, by definition, a certain amount of errors or accuracy loss. As such, one has to practically work with the skeletons that are computed from such (approximative) shapes. Consequently, such skeletons will also suffer from various degrees of inaccuracy, as opposed to the 'true' skeletons implied by the original shapes that have been sampled. Hence, being able to reason about inaccuracies present in computed skeletons is an important task.

Skeleton inaccuracies have two main causes. The first one relates to the above-mentioned discretization (sampling) approximations that inherently exist in the input shapes. The second cause relates to various simplifying assumptions, approximations, or heuristics used by the different existing skeletonization methods. Since it is, often, hard to distinguish inaccuracies in skeletons based on their cause, the typical way to reason about the quality of skeletons is by defining a number of *desirable properties* that the computed skeletons and/or skeletonization methods used to compute them should have.

Existing work in skeletonization over the last decade has converged to a number of desirable properties which are well-accepted by most researchers, and considered to be important to be satisfied by any skeletonization method [58, 191, 203]. These properties should equally hold for both 2D medial axes and 3D surface skeletons. For 3D curve skeletons, only a subset of the properties is relevant. Several such properties emerge directly from the skeleton definition (Eqn. 2.2), while other properties capture desirable features implied by many practical applications that use skeletons. The aforementioned properties are briefly described below.

Homotopy: Skeletons should have the same homotopy as the shapes they are extracted from, *i.e.*, they have the same number of connected components, holes (in 2D), tunnels and cavities (in 3D). If skeletons computed by practical methods respect this

property, then they can be further easily used to reason about the shape's topology, which is in turn useful in many shape matching and shape retrieval applications [218].

Invariance: Skeletons should be invariant to isometric transformations. That is, given a shape Ω and an isometric transformation T, $S(T(\Omega)) = T(S(\Omega))$. This property directly follows from the skeleton definition (Eqn. 2.2) as this definition says that the skeleton depends only on the shape Ω and not on the shape's position, size, or orientation in the embedding space. In general, mesh-based methods, that represent both Ω and $S(\Omega)$ as a polygonal mesh, comply relatively well with this property. In contrast, pixel-based and voxel-based methods, that represent both Ω and $S(\Omega)$ as a pixel, respectively voxel, grid, cannot be *fully* invariant to such transformations, since the sample positions are constrained to a fixed grid.

Thinness: Skeletons following the definition in Eqn. 2.2 should be infinitesimally thin, *i.e.*, composed of curve and surface fragments. In practice, this means that skeletons computed by the various skeletonization methods out there should be as thin as the spatial representation used by the respective methods allows it. Mesh-based methods typically achieve the desired zero thickness by construction. In contrast, the thickness of voxel-based skeletons is lower bounded by the grid resolution – meaning, in practice, that pixel-based and voxel-based skeletons cannot be thinner than the size of one pixel, respectively voxel.

Centeredness: By definition, each skeleton point should be at equal distance from at least two different points of the shape boundary $\partial\Omega$ (Eqn. 2.2). Several issues exist for practical skeletons with respect to centeredness. First and foremost, centeredness is con- strained by the spatial sampling used to represent the skeleton. Specifically, there exist shapes where pixel-based and voxel-based skeletons cannot be perfectly centered due to the fixed grid resolution. Consider the simple example of the 2D axis-aligned rectangle Ω whose formal skeleton is shown in Fig. 2.3 (left). In a pixel-based spatial sampling where the rectangle width is even, the skeleton will either completely miss the central branch (marked gray in Fig. 2.3 (right)), or, if constraints are used to ensure homotopy, it will have a two-pixel thick branch covering all gray pixels, or a one-pixel-thick branch which is not perfectly centered with respect to $\partial\Omega$.

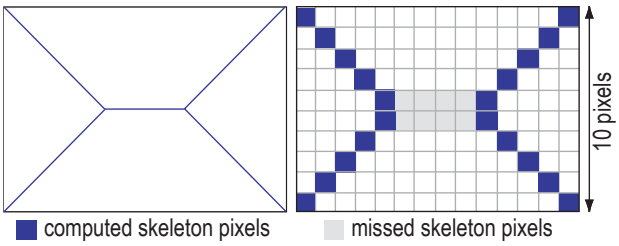


Figure 2.3: Thickness and centeredness issues. (left) Formal skele- ton and (right) its counterpart computed on a fixed pixel grid. Image taken from [226].

A separate centeredness issue arises when considering curve skeletons. Since there is no universally accepted definition of what curve skeletons are, it is also hard to formally talk about their centeredness. In most applications, the centeredness of curve skeletons is assessed only qualitatively, by visually examining that the produced curves appear to be locally far away from the boundary of the input shape. In practice, having well-centered skeletons is of crucial importance when further using such skeletons to reconstruct shapes by the medial axis transform (MAT) (see Sec. 2.2.1) or to plan paths that need to be close to the center of complex shapes [236].

Smoothness: Surface skeleton manifolds are known to be at least piecewise C^2 continuous, regardless of the noisiness of the input shape surface [176, 203]. As such, skeletonization methods should produce piecewise-smooth skeletons. Several problems exist related to smoothness for practical skeletons. First, it is hard to quantitatively assess when a skeleton is sufficiently smooth, since computing a ground-truth skeleton on which smoothness can be assessed is hard or even impossible, as discussed earlier in this section. Secondly, smoothness can be limited by the sampling resolution and/or sampling model used for the skeleton (*i.e.*, mesh- or voxel-based). The smoothness of mesh-based skeletons strongly depends on the local point density used for sampling. Many such methods have difficulties in densely sampling the so-called *ligature branches* [152], *i.e.*, branches which emerge from the main skeleton rump towards small-scale convex bumps on the input surface $\partial\Omega$ [109]. Smoothness can be increased by low-pass filtering the computed skeletal points [15, 98, 105, 227]. However, unconstrained smoothing can adversely affect centeredness.

Regularization: As discussed earlier in this section, skeletons are sensitive to small-scale perturbations on the surface $\partial\Omega$ of the input shape Ω. Such perturbations can occur due to multiple causes, such as inaccurate data acquisition or insufficient sampling resolution. As we have discussed earlier, skeletons can be regularized by applying various types of local and global importance metrics. Often, regularization is automatically performed as the last step of a skeletonization method, so that the method is guaranteed to deliver skeletons free from spurious branches. As such, regularization is said to enforce the desirable property of *robustness* (of skeletons to small-scale perturbations). This property is also known under the name of insensitivity to noise.

Reconstructibility: As outlined in Sec. 2.2.1, the medial axis transform (MAT) provides a full (dual) encoding and is also fully invertible. In theory, this allows reconstructing Ω from its MAT, a process also known under the name of *garbing* [68] or shape reconstruction from its skeleton [11, 109]. However, as we have seen, practical skeletons are affected by various approximations, such as imperfect centeredness and the absence of noise-induced branches. As such, exact reconstructibility of a shape from its skeleton is, in practice, rarely possible. High-resolution and accurately-centered mesh-based surface skeletons can efficiently reconstruct shapes up to high detail [109] Voxel-based surface skeletons also offer good reconstruction accuracy [11], albeit with lower quality as compared to mesh methods, due to the discussed limitations of fixed grids.

Scalability: To be useful and usable in practical applications, skeletonization methods should be scalable with respect to the size of the input shape Ω, typically measured in terms of number of sample elements (pixels, voxels, or mesh points) used to represent it. As modern data acquisition devices are able to acquire increasingly large 2D and 3D digital shapes in increasingly lower time-frames, skeletonization methods need to be increasingly more scalable. To achieve this goal, several approaches have been proposed. Voronoi-based methods typically achieve a complexity of $O(n \log n)$ for n sample points on $\partial\Omega$ [8, 70, 168]. Distance-field-based methods using voxel sampling achieve a complexity of $O(T \log \|\partial\Omega\|)$, where T is the average shape thickness [80, 228], which is comparable with the complexity of Voronoi methods. Other methods achieve a linear complexity in the number of input voxels $\|Omega\|$ [102, 183], which is slightly higher than [80, 168, 228]. Mesh contraction methods have a complexity of $O(n\,s)$, where n is the number of surface samples $\|\partial\Omega\|$ and s is the number of contraction iterations, typically a constant [15, 109, 151]; linear- complexity $O(n)$ methods also exist [221]. Overall, all such methods are quite similar in theoretical computational costs. However, much can be gained in terms of practical costs, by parallelizing on the graphics processing unit (GPU) the skeleton-detection operations, *e.g.*, distance transform computation [41] or mesh shrinking method [109, 151]. Such methods can compute surface skeletons of mesh models having up to one million vertices in subsecond time on a modern PC computer.

While the above properties are well-known and often mentioned in the skeletonization literature, comparatively little work has been done in *checking* them on the vast body of existing skeletonization methods. This makes it further hard to compare different methods among themselves and, thus, to select the optimal skeletonization method for a given application context having specific requirements and constraints.

2.3 CONCLUSIONS

In this chapter, we have reviewed the core two topics related to the work in this thesis – research related to the restoration of 2D and 3D shapes by using inpainting methods, and research related to the computation of skeletal descriptions of 3D shapes. Three main conclusions can be drawn from the above review, as follows. First, the area of volumetric restoration of 3D shapes by inpainting and hole-filling methods is far less populated than the restoration of 3D surfaces and 2D images. By itself, this indicates a promising direction for future research. Secondly, we have not identified any significant class of methods that uses 3D skeletons for the volumetric restoration of 3D shapes. In addition to the first point mentioned above, this strengthens our conviction that exploring the use of skeletons for 3D shape inpainting is a novel research domain. Thirdly, we have seen that a large number of methods exist for computing both surface and curve skeletons of 3D shapes. However, picking an optimal method from this large set (for further use in our volumetric inpainting context) is very challenging, due to the lack of extensive comparisons of existing methods, and also due to the very different heuristics and models used by different skeletonization methods. The three points mentioned above set the research agenda for the work further described in this thesis.

3 INPAINTING IN FACIAL IMAGES

As outlined in the previous chapters, the main goal of this thesis is to explore the possibilities of using skeletal descriptors to design efficient and effective image and shape restoration methods that guarantee a number of global properties of the restored shape. As outlined in Chapter 2, a key set of techniques helping such restoration tasks are formed by inpainting techniques. Furthermore, the overall restoration success is affected by two independent elements: (1) the *detection* of damaged regions to repair and (2) the *repairing* itself.

To get a better understanding of the challenges affecting shape and image restoration, we describe in this chapter an application where restoration plays a key role: The automatic restoration of low-quality facial images. In this application, both damaged-region detection and inpainting-based repairing come into play. We present novel methods for both above tasks, and compare our solution with restoration methods based on well-known existing inpainting techniques.

3.1 INTRODUCTION

Facial recognition is the process of obtaining the identity of a person based on information obtained from facial appearance [16]. Compared to other person identification methods such as biometric fingerprints, retina scans, and voice recognition, facial recognition has the advantage that it can be used in contexts where the collaboration of the person to be identified is not possible [103, 245]. One such context is the identification of missing people based on existing photographs thereof.

In the last decades, automatic face recognition has received considerable attention from the scientific and commercial communities. However, several open issues still remain in this area. One such issue is that facial recognition tools are in general not efficient for poor quality facial images, *e.g.* in the presence of shadows, artifacts, and blurring [44, 243, 245].

In some contexts, facial photographs are the only key to person identification. For example, the sites of Australian Federal Police [3], Federal Brazilian Government [83], and UK Missing People Centre [162] publish facial photographs of missing people. Often, such photographs are old, poorly digitized, and have artifacts such as folds, scratches, irregular luminance, molds, stamps, and written text, all of various sizes, shapes, texture, and color. Since the effectiveness of facial recognition methods depends on the quality of input images, it is of high importance that such images present the discriminant facial features (eyes, mouth, and nose) with minimal artifacts. Moreover, it is desirable that they all have a standard look, *e.g.* avoid outlier elements such as glasses, hair locks, highlights, and smiles.

We present here a computational framework for segmentation and automatic restoration of poor-quality facial images, based on a statistical model built from frontal facial images and inpainting techniques. The location of facial features is obtained by a mean image generated from a sample population of conforming facial

images that provides privileged information about spatial location of the features. Salient outliers are found by statistically comparing the input image with this mean image. These outliers are next eliminated by using a modified inpainting technique which uses information from both the input image itself and the closest high-quality image in our image database. This produces images of sufficient quality for typical face recognition tasks. In contrast to most existing digital restoration methods, we require no user input to mark regions in the image which has to be restored. Our method is simple to implement and can be run in real-time on low-cost PCs, which makes it attractive for utilization by government agencies in least developed countries.

This chapter is structured as follows. Section 3.2 describes related work on image inpainting. Then, section 3.3 presents our method for facial artifact segmentation and removal by inpainting. Next, section 3.4 shows and discusses our results on images from public photograph databases and presents comparisons with existing inpainting methods. Finally, Section 3.5 concludes the chapter.

3.2 RELATED WORK

Many image inpainting techniques have been proposed in the last decade. Oliveira *et al.* [169] present a simple inpainting method which repeatedly convolves a 3×3 filter over the regions to inpaint. Although fast, this method yields significant blurring. Bertalmio *et al.* [23, 24] estimate the local image smoothness, using the image Laplacian, and propagate this along isophote directions, estimated by the image gradient rotated 90 degrees and by Navier-Stokes equation. The Total Variational (TV) model [46] uses an Euler-Lagrange equation coupled with anisotropic diffusion to keep the isophotes directions. Telea [229] inpaints an image by propagating the image information from the boundary towards the interior of the damaged area following a fast marching approach and a linear extrapolation of the image field outside the missing region. These methods give good results when the regions to restore are small.

To inpaint thicker regions, the Curvature-Driven Diffusion (CCD) method [47] enhances the TV method to drive diffusion along isophote directions. Diffusion-based texture synthesis techniques have been proposed to achieve higher quality inpainting results [37, 38]. Closer to our proposal, Li *et al.* [144] present semantic inpainting, where the damaged image is restored based on texture synthesis using a most similar image from a given database. Perez *et al.* [173] and Jeschke *et al.* [112] present seamless image cloning that uses a Poisson process to compute the seamless filling as well as a guidance vector field that incorporates prior knowledge of the damaged domain.

Image cloning gives very good results, but is relatively expensive in CPU and memory terms. Joshi *et al.* use example images, taken from a person's photo collection, to improve the luminance, blur, contrast, and shadows of that person's photograph [116]. Chou *et al.* also use the seamless image cloning of Perez *et al.* for the editing of facial features in digital photographs [50]. However, in contrast to our goal of automatic removal of facial outlier artifacts, their goal was to assist users in previewing the results of explicitly chosen image modifications.

To achieve our goal of facial image preprocessing for supporting automatic missing persons identification in low-cost contexts, we need a method which is able to

- automatically detect the most salient artifacts in a low-quality facial photograph;
- remove these artifacts in a plausible way;
- work (nearly) automatically;
- require very low computational costs.

Next, we present our proposal in order to address these requirements.

3.3 PROPOSED METHOD

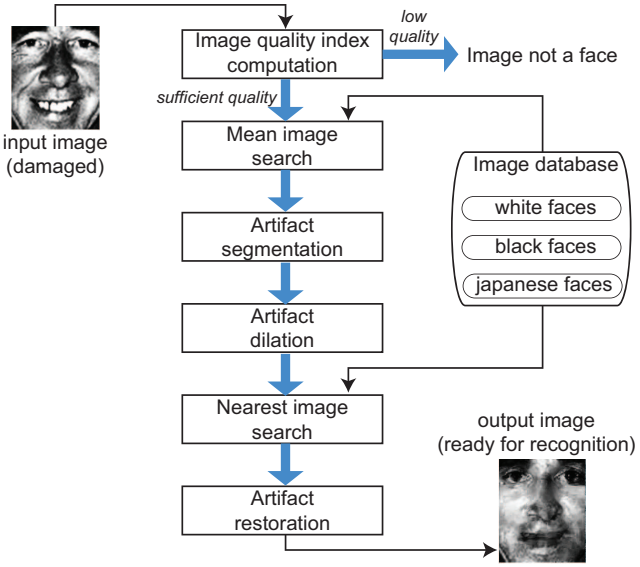


Figure 3.1: Pipeline of proposed method

Figure 3.1 illustrates our computational pipeline for facial image restoration. We start by computing an image quality index, which classifies the input image as potentially benefiting from artifact removal or not (Sec. 3.3.1). If the image can be improved, we next identify, or segment, its artifacts using a statistical decision method where the image is compared to an existing database of facial images of various ethnicities (Sec. 3.3.2). The detected artifacts are next slightly enlarged using morphological dilation [10] to remove small-scale spatial noise such as artifact boundary jaggies. Finally, we eliminate the artifacts by using a semantic inpainting, a variation of the method presented in [144], which combines information from the input image and the image database (Sec. 3.3.3).

All our images (input and database) are normalized and equalized by the framework proposed in [7]. This is needed since existing facial image databases around the world contain images with several sizes, resolutions, and contrasts. In our work, we

use grayscale images because many face photographs from missing people, in typical databases, are quite old, and therefore available only in this format. If color photographs are available, we can handle these easily by first converting them to grayscale, by *e.g.* considering only the luminance of their HSV representation. The steps of our pipeline are explained next.

3.3.1 *Image Quality Index*

Objective image quality measures have an important role in image processing applications, such as compression, analysis, registration, restoration and enhancement. One such simple measure is the image quality index [238], which compares two images $\mathbf{x}$ and $\mathbf{y}$ based on luminance $l(\mathbf{x},\mathbf{y})$, contrast $c(\mathbf{x},\mathbf{y})$ and structure $s(\mathbf{x},\mathbf{y})$ comparison measures

$$l(\mathbf{x},\mathbf{y}) = \frac{2\mu_x\mu_y}{\mu_x^2+\mu_y^2}, \tag{3.1}$$

$$c(\mathbf{x},\mathbf{y}) = \frac{2\sigma_x\sigma_y}{\sigma_x^2+\sigma_y^2}, \tag{3.2}$$

$$s(\mathbf{x},\mathbf{y}) = \frac{\sigma_{xy}}{\sigma_x\sigma_y}. \tag{3.3}$$

where $\mathbf{x} = \{x_i | 1 \leq i \leq n\}$ and $\mathbf{y} = \{y_i | 1 \leq i \leq n\}$ are the two n-pixel images to compare. Here, μ_x, μ_y, σ_x^2, σ_y^2 and σ_{xy} are the mean, variance, and covariance of $\mathbf{x}$ and $\mathbf{y}$, respectively [196], *i.e.*

$$\mu_x = \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad \mu_y = \frac{1}{n}\sum_{i=1}^{n} y_i$$

$$\sigma_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu_x)^2, \qquad \sigma_y^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i-\mu_y)^2$$

$$\sigma_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu_x)(y_i-\mu_y)$$

The quantities $l(\mathbf{x},\mathbf{y})$ and $c(\mathbf{x},\mathbf{y})$ range between 0 and 1, while $s(\mathbf{x},\mathbf{y})$ is between -1 and 1. For normalized and equalized (NEQ) images, as in our case, $l(\mathbf{x},\mathbf{y})$ and $c(\mathbf{x},\mathbf{y})$ have a maximum of 1 since NEQ images have very similar standard deviation and mean values. In contrast, covariance values σ_{xy} are distinct in NEQ images. Hence, $s(\mathbf{x},\mathbf{y})$ (Eqn. 3.3) is a good candidate for assessing the similarity of two NEQ images. Note that $s(\mathbf{x},\mathbf{y})$ does not give a direct descriptive representation of the image structures: It reflects the similarity between two image structures, where $s(\mathbf{x},\mathbf{y})$ equals one if and only if the structures of the two images $\mathbf{x}$ and $\mathbf{y}$ are exactly the same.

We use the image quality index s for two purposes. First, given an input image $\mathbf{x}$, we compare it with the average image $\overline{\mathbf{x}}$ of our pre-computed image database (see Sec. 3.3.2), using $s(\mathbf{x},\overline{\mathbf{x}})$. If $\mathbf{x}$ is too far away from $\overline{\mathbf{x}}$, then $\mathbf{x}$ is either not a face image, or an image we cannot improve; and so, we stop our pipeline. This is further detailed in the quality index discussion in Sec. 3.4. Otherwise, we determine the so-called outlier artifacts which differentiate $\mathbf{x}$ from $\overline{\mathbf{x}}$, and suppress them, as shown next.

3.3.2 *Statistical Artifact Segmentation*

Once we have chosen to improve the quality of an input image, we aim to *segment* those image parts, or artifacts, which deviate significantly from typical average images. These will be the target of our inpainting (Sec. 3.3.3).

For artifact segmentation, we use statistical decision methods based on inference theory [39, 213], where samples in a database can generate privileged information. This makes it possible to discriminate artifacts which we next want to remove from regular image pixels. Given a database of N of NEQ-normalized facial frontal images $\mathbf{x}_i$, the mean image is given by:

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x_i}. \tag{3.4}$$

We compute mean images $\overline{\mathbf{x}}^{DS}$, $\overline{\mathbf{x}}^{LS}$, and $\overline{\mathbf{x}}^{JP}$ separately for images of people of three ethnicities (dark skin (*DS*), light skin (*LS*), and japanese (*JP*)). This avoids mixing up too different facial traits. Figure 3.2 shows the mean and standard deviation images for these three groups. For dark skin people, we use the FERRET database, (65 images) [177] and 14 images taken from the FEI database [232]. For light skin people, we consider 100 images from the FEI database. For japanese people, we use the JAFFE database (100 images) [160]. These databases contain a variety of images for people of different races, ages, and appearances. Although all images are equalized, facial structure differences exist, *e.g.* larger nose and lips (*DS* images), shadows around the eyes (*LS* images), and significant mouth position variations (*JP* images).

We next determine the ethnicity of the input image $\mathbf{x}$ by finding its closest mean image $\overline{\mathbf{x}}_{min} \in \{\overline{\mathbf{x}}^{DS},\overline{\mathbf{x}}^{LS},\overline{\mathbf{x}}^{JP}\}$, in terms of Euclidean distance.

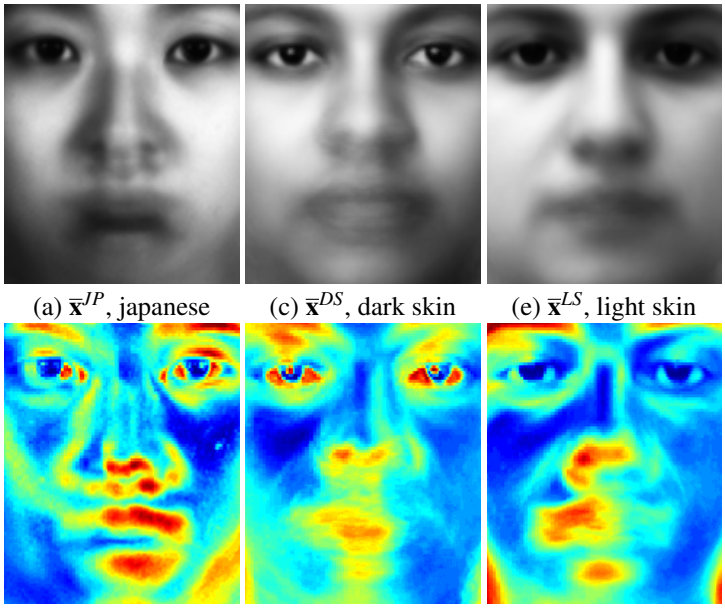
Finally, we use statistical segmentation to find the artifact regions to be removed. To illustrate this, let us suppose that the face samples are represented by a random field $\mathbf{x}$ that follows a normal distribution with mean $\overline{\mathbf{x}}_{min}$ and standard deviation σ.

Thus the distribution of the standardized variable is given by:

$$\mathbf{z} = \frac{\mathbf{x}-\overline{\mathbf{x}}_{min}}{\sigma}. \tag{3.5}$$

Here, σ is the standardized normal distribution of $\mathbf{x}$ with mean 0 and variance 1, computed similariy to $\overline{\mathbf{x}}$, *i.e.* separately for the white skin, dark skin, and japanese face databases.

Table 1 shows the values of per-pixel z scores and significance levels α. The set of z scores outside the range $[-2.58, 2.58]$ is called the critical region of the hypothesis



(a) $\bar{\mathbf{x}}^{JP}$, japanese (c) $\bar{\mathbf{x}}^{DS}$, dark skin (e) $\bar{\mathbf{x}}^{LS}$, light skin

(b) σ^{JP}, japanese (d) σ^{DS}, dark skin (f) σ^{LS}, light skin

Figure 3.2: Mean images $\bar{\mathbf{x}}$ and color-coded standard deviation σ images for japanese, black skin, and light skin ethnicities.

or region of significance. This is the region of rejection of the hypothesis. The set of z scores inside the range $[-2.58, 2.58]$ is called the region of hypothesis acceptance or the non-significance region.

Table 1: Statistical significance

Level of significance α	Values of z
10%	- 1.645 .. 1.645
5%	-1.96 .. 1.96
1%	- 2.58 .. 2.58
0.1%	-3.291 .. 3.291

In practice, we observed that a value of $\alpha \simeq 10\%$ gives a good selection of atypical face features. In other words, for each pixel z of the image $\mathbf{z}$, computed by Eqn. (3.5), we decide if it is an artifact pixel or not, based on the value of z and our threshold α (Tab. 1). The set of artifact pixels Ω are removed as described next.

3.3.3 *Semantic Inpainting*

Now that we know which pixels of our input image are likely to be artifacts, we show a way to remove them by using inpainting. As our approach combines the fast

marching based inpainting [229] and Poisson image editing [173] techniques, we first briefly outline relevant aspects of these techniques.

3.3.3.1 *Inpainting using Fast Marching Method*

Inpainting technique based on fast marching method (FMM) considers a first order approximation $I_q(p)$ of the image at a point p situated on the boundary $\partial\Omega$ of the region to inpaint Ω

$$I_q(p) = I(q) + \nabla I(q) \cdot (p - q), \tag{3.6}$$

where q is a neighbor pixel of p located within a ball $B_\varepsilon(p)$ of small radius ε centered at p and $I(q)$ is the image value at q. Each point p is inpainted as function of all points $q \in B_\varepsilon(p)$, by summing the estimates of all points q, weighted by a weighting function $w(p,q)$:

$$I(p) = \frac{\sum_{q \in B_\varepsilon(p)} w(p,q) I_q(p)}{\sum_{q \in B_\varepsilon(p)} w(p,q)} \tag{3.7}$$

where the application-dependent weights $w(p,q)$ are normalized, *i.e.* $\sum_{q \in B} w(p,q) = 1$.

The boundary $\partial\Omega$ is advanced towards the interior of Ω using the FMM [198]. While doing this, FMM implicitly computes the so-called distance transform $DT : \Omega \to \mathbb{R}^+$

$$DT(p \in \Omega) = \arg\min_{q \in \partial\Omega} \|p - q\|. \tag{3.8}$$

As the FMM algorithm visits all pixels of the damaged region the method gradually propagates gray-value information from outside Ω inwards by computing expression (3.7).

This inpainting method is fast and simple. However, for regions thicker than roughly 10..25 pixels, it creates an increased amount of blurring as we go farther from $\partial\Omega$ into Ω, given the smoothing nature of Eqn. 3.7. Since our facial artifacts are not guaranteed to be thin, this method is not optimal.

3.3.3.2 *Poisson image editing*

This image restoration method allows to achieve seamless filling of the damaged domain by using a source image data. It renders remarkably good results, even for thick target regions. The method is based on solving the equation:

$$\Delta I = div\ \mathbf{v} \quad \text{on } \Omega \tag{3.9}$$

with I known over the boundary $\partial\Omega$ and constrained to the values of $\mathbf{v}$, the so-called guidance field, that is derived from a source image. For instance, if we desires to seamlessly clone a source image I_{src} onto Ω, we can set $\mathbf{v} = \nabla I_{src}$, which effectively reduces Eqn. 3.9 to $\Delta I = \Delta I_{src}$ on Ω (Eqn. 10 in [173]). Solving the Poisson problem

(Eqn. 3.9), however, is expensive in terms of memory requirements, which is critical in our low-cost scenario [81].

3.3.3.3 *Proposed inpainting method*

Both the FMM inpainting method and Poisson image editing basically use *only* the non-artifact areas of the input and near image to restore its artifacts.

However, we have additional information coming from our high-quality image database. Hence, our proposal is to combine the fast-and-simple inpainting method of [229] with the Poisson image editing [173] augmented with information extracted from our image database (Fig. 3.3).

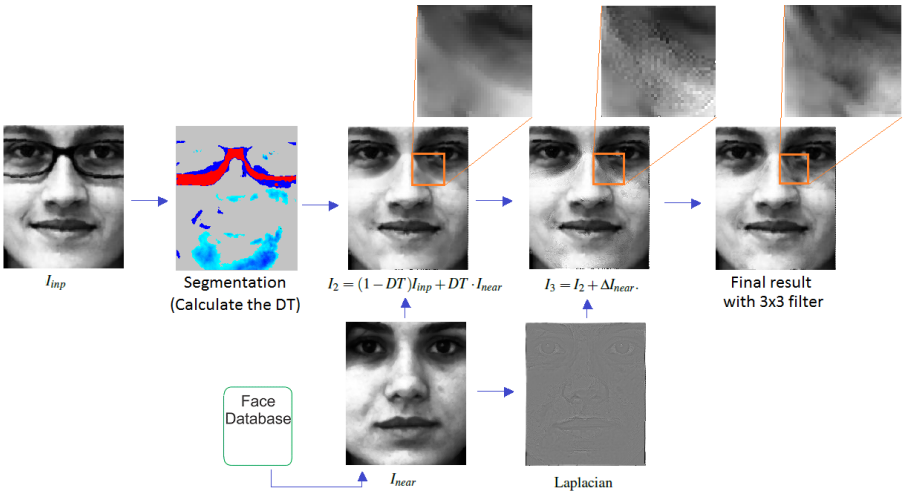


Figure 3.3: Diagram of the proposed inpainting method (see Sec. 3.3.3).

We start by applying FMM inpainting (Sec. 3.3.3.1) on the artifact region $\Omega \subset \mathbf{x}$ of our input image $\mathbf{x}$. We denote the result of this step, *i.e.* the image $\mathbf{x}$ where Ω has been inpainted, by I_{inp}. Next, we search in our face image database DB for the *closest* facial image I_{near}, *i.e.*

$$I_{near} = \arg\min_{\mathbf{y} \in DB} \|\mathbf{x} - \mathbf{y}\|. \tag{3.10}$$

As outlined in Sec. 3.3.3.1, when Ω is thick, FMM inpainting produces a blurred result I_{inp}. We correct this by mixing I_{inp} and I_{near} to yield:

$$I_2 = (1 - DT)I_{inp} + DT \cdot I_{near} \tag{3.11}$$

where DT is the distance transform of $\partial\Omega$ (Eqn. 3.8). This progressively mixes the inpainting result I_{inp} with the nearest image I_{near} in a progressive way. At the border $\partial\Omega$, we see the inpainted image, which smoothly extrapolates the non-artifact area out of the damaged region in the input image. In the middle of Ω the blurred effects observed in I_{inp} are attenuated by the high-quality image I_{near}.

Next, we calculate Laplacian $\Delta I_{near} = \frac{\partial^2 I_{near}}{\partial x^2} + \frac{\partial^2 I_{near}}{\partial y^2}$ of the closest image I_{near}, using central differences, and add this component to our restoration, *i.e.* compute

$$I_3 = I_2 + \Delta I_{near}. \tag{3.12}$$

This is a simplified form of the 'seamless cloning' presented in [173] which just adds high-frequency features to I_2 through the Laplacian operation ΔI_{near}." In contrast to [173], our approach is much cheaper, as it does not require solving a Poisson equation on Ω.

As a final step, we apply a 3-by-3 median filter on I_3 to yield the final reconstruction result. This further removes small-scale noise elements created by using the finite-difference Laplacian (Eqn. 3.12).

3.4 RESULTS

Our method can produce good artifact removal even in large and thick regions, due to the mix of inpainting (which preserves information specific to the input image) and prior information (which introduces information from a similar high-quality image from our image database). A key element is that our image database is composed by high quality images. Therefore, they are considered to be *good* for facial recognition purposes. Besides this information is suitable for both to extract the artifacts (Sec. 3.3.2) and to restore the corresponding damaged areas.

We tested our method on 79 frontal facial images provided by several public organizations: the Federal Brazilian Government [83], the Australian Federal Police [3], and the UK Missing People Organization [162]. These images contain various levels of artifacts, *e.g.* annotations, scratches, glasses, obscuring facial hair, folds, and exposure problems. All input images were normalized and equalized as outlined in Sec. 3.3.

Segmentation: Figure 3.4 shows the results when applying our segmentation approach for the images of collum (a) using the light-skin mean image (Fig. 3.4.(b)), dark-skin mean image (Fig. 3.4.(c)) and japanese mean image (Fig. 3.4.(d)). We observe that if significance level $\alpha \leq 1.0\%$ we can extract the artifacts (glasses) when using the light and dark-skin mean image images. Segmentation of artifacts works best and has similar perceived quality results for dark-skin and light-skin images (see Fig. 3.4). For several images, the mean of japanese people presents very good results. Although segmentation cannot select *all* details where an input image differs from its ethnic group's statistical average, it does a good job in selecting details deemed to be unsuitable for official person recognition photographs.

Significance level: Figure 3.5 shows results when changing the significance level α (Sec. 3.3.2). If we choose high values for α the method extracts the artifacts as well as other regions, as we can see on Figure 3.5 . Conversely, the choice of low α values, like in Figure 3.5, prevent such problem but is not efficient to segment the region of interest. The optimum value for α is application dependent and it depends on manual fine-tuning in general. For the test performed, the value $\alpha = 10\%$ has given good results in all cases. This fact can be verified in Figure 3.6 which shows the results using some values of α.

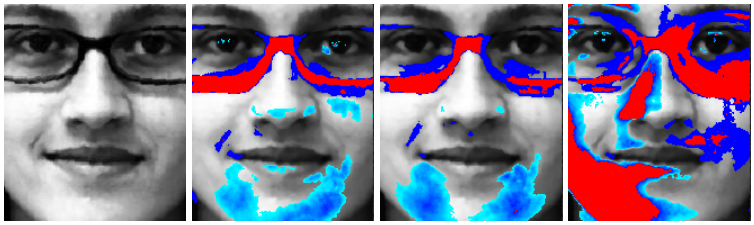
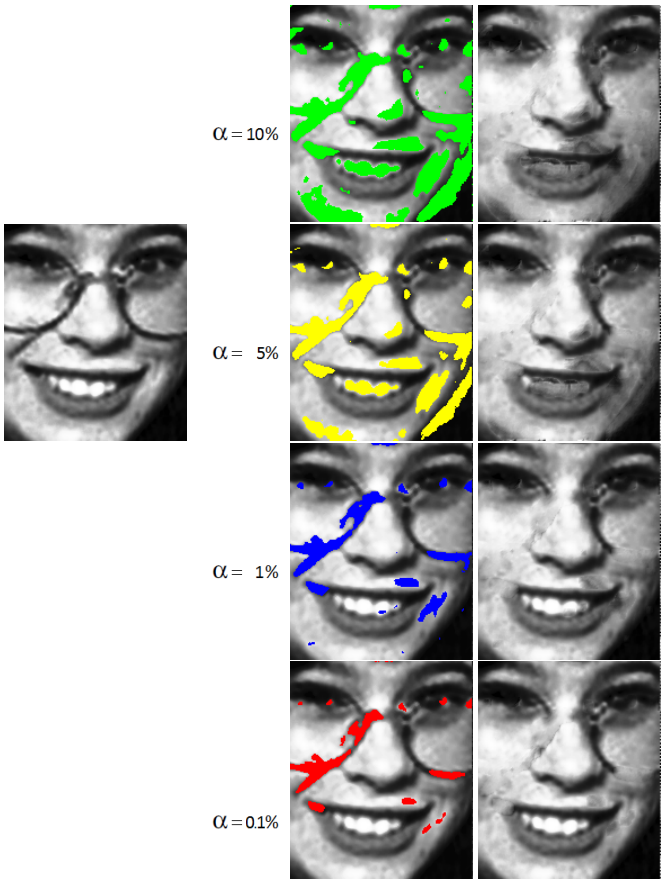


Figure 3.4: Segmentation comparison, with $\alpha = 0.1\%$ (red), $\alpha = 1\%$ (dark blue), and $\alpha = 10\%$ (light blue). (a) original image; (b) light skin mean image $\bar{\mathbf{x}}^{LS}$; (c) dark skin mean image $\bar{\mathbf{x}}^{DS}$; (d) japanese mean image $\bar{\mathbf{x}}^{JP}$



(a) (b) (c)

Figure 3.5: Segmentation results: (a) original image; (b) segmentation; (c) reconstruction.

Artifacts: *Smile* is considered an artifact, as it does not comply with official standards for facial images concerning expression, illumination, and background [9, 106, 230].

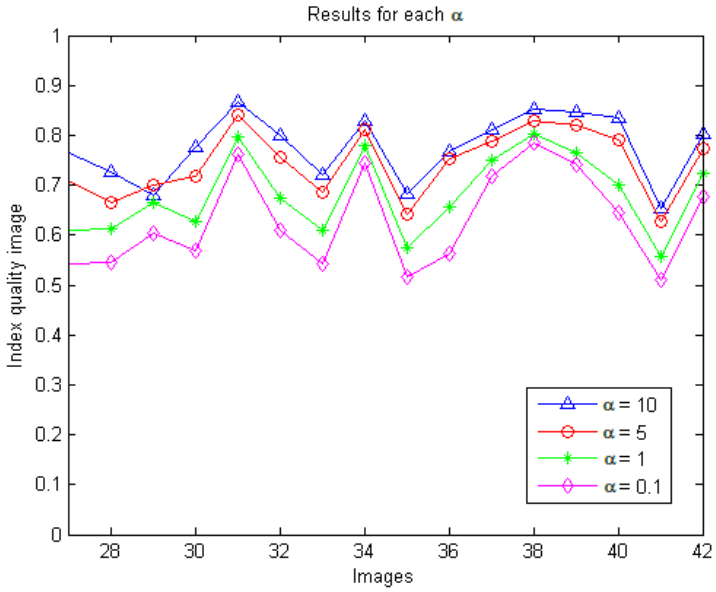


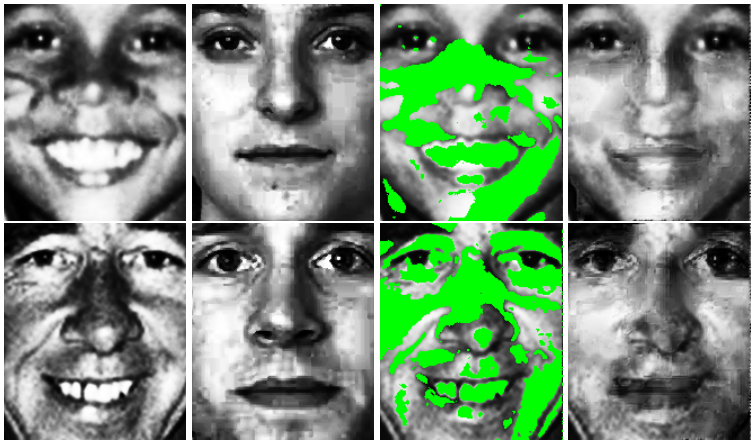
Figure 3.6: Results at each value of α.

Figure 3.7 presents an example of smile suppression. The bottom row shows a case where the smile has been well segmented and suppressed. The top row shows a less successful case: Part of the large smile (see Fig. 3.7 a, top) has not been captured by the segmentation, and as such, it leaked in the final reconstruction (Fig. 3.7 d, top). Also, note that our method can produce changes in facial expressions. For example, Fig. 3.7 (bottom row, d) shows a face where the eyes are a mix between the input image and nearest image, whereas the nose, and cheeks follow more closely the input image.

Figure 3.8 shows further results, and also compares them with three known inpainting techniques. In the top row, we see an image with a hair lock artifact. Segmenting this artifact is easy, as it is much darker than the mean light-skin image $\overline{\mathbf{x}}^{LS}$. Inpainting this artifact is also relatively easy, as it is not thick. The second row shows a face image which has a highlight, as well as a thin horizontal crease (most probably due to a photograph damage) on the cheeks. This artifact is more complicated to capture since its grayvalue field is similar with the mean image pixels nearby. However, the result (column *g*) shows that this artifact is also largely eliminated from the input image.

The third row of Figure 3.8 shows an image with glasses as an outlier artifact. Existing inpainting methods (columns d-f) succeed in eliminating this artifact less than our method (column g), because the eyes are reconstructed. The fifth row of Figure 3.8 shows a similar case. Here, the artifact region is thin, so all four tested inpainting methods produce eliminate the glasses equally well.

In the fourth row Figure 3.8 it is shown an image where a large shadow artifact appears under the nose, on the cheeks, and the mouth. As expected, standard inpainting cannot easily remove this problem, since the shadow slightly extends outside of the segmented region, on the base of the nose (dark area under the nose, Fig. 3.8 c,



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

Figure 3.7: Smile elimination: (a) input image; (b) nearest image; (c) segmentation; (d) proposed inpainting.

fourth row). By using the nearest image our method can be more efficient to remove the shadow while generating less spurious details. A similar effect is observed in Figure 3.8 (bottom row), where our method performs better for both to remove the cheek highlights and to reconstruct the opened eyes better than existing methods.

Quality index: Figure 3.9 shows the average structural image quality index (Equation 3.3) for the three ethnicities present in our face image database, the original input images for inpainting, and the results of the studied inpainting methods. Several observations follow. First, as expected, the face database images have a relatively high quality index (average $s \in [0.6, 0.8]$). Secondly, input images which are faces have a lower quality (average $s \sim 0.49$). This motivates our proposal to adjust such images to bring them in line with the quality level of the face database. In contrast, input images which are not faces have a much lower quality index (average $s \sim 0$). This allows us to decide whether an input image should be improved or deemed not improvable (because it is not a face): We choose a suitable threshold τ, in this case, $\tau = 0.1$. Input images with $s > \tau$ are likely faces, so they are further improved; images with $s < \tau$ are likely non-faces, and thus skipped from the process. Thirdly, we notice that the nearest image I_{near} used in our inpainting (Sec. 3.3.3.3) has an average high quality index, which justifies its explicit usage during inpainting. Finally, we see that our method yields, on average, results with a higher structural quality than the others studied inpainting methods.

Limitations: Despite of its capabilities, our method cannot handle facial images which deviate too much from the information provided by the predefined image database. Figure 3.10 shows such an example, which is the worst case we found in our tests: The input image (a) is too far away from *both* the average black-skin image (Fig. 3.2 c) and the nearest image (Fig. 3.10 b) to successfully remove the facial hair,

Figure 3.8: Method results: (a) original image; (b) nearest image (not the same person, just similar); (c) segmentation, where green color indicates significance level $\alpha = 10\%$, yellow is 5%, blue is 1% and red is 0.1%; (d) inpainting [169]; (e) inpainting [24]; (f) inpainting [229]; (g) our method.

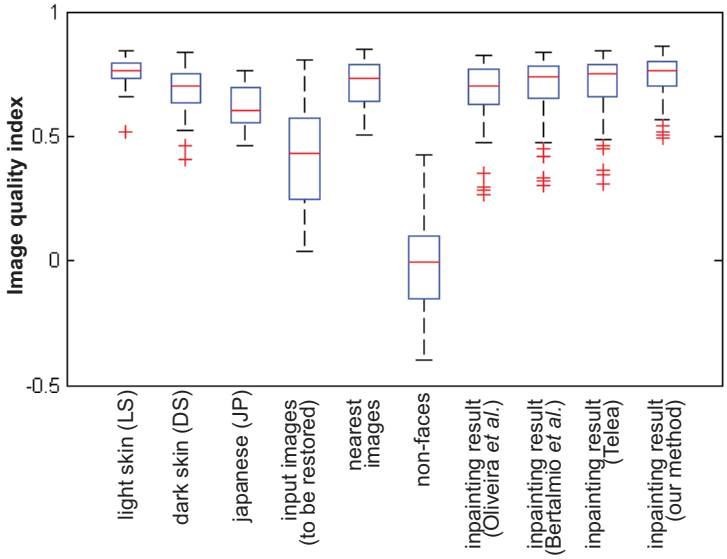


Figure 3.9: Structural image quality for the input images, face database, and result images.

nose pin, and large opaque glasses. However, we note that this type of outlier can be easily detectable by our approach: When the input image is too far away from the mean image, as mentioned above, our method reports that it cannot likely improve this image and stops.

Database: Selecting images that has good quality for the face database serves two purposes. First, this allows specifying what one considers to be an acceptable facial image in a given context (*e.g.* open eyes, no smiles, shadows, adorns, or imprints). Statistical characteristics of this collection are used to implicitly determine what are outliers, thus what has to be suppressed in a given input image. Secondly, features of the nearest image in this database are used in the restoration. Hence, if one wants to allow certain specific feature to persist in the restored images, this can be done by inserting images containing such features in the database.

Computational aspects: The proposed method is simple to implement. For an image of n pixels, its memory and computational complexities are $O(n\ log\sqrt{n})$ [198] respectively, in contrast to more sophisticated inpainting algorithms [24, 46, 47, 116, 173]. This makes our method suitable for computers that have low capacity of memory.

Validation: Although we have not validated the added value of our artifact suppresion method in a full end-to-end face recognition pipeline, which would be used in concrete cases by *e.g.* law enforcement officers, we have evidence that such a pipeline (having the properties of ease-of-use, speed, and simple implementation that our proposal has) would be very useful. For instance, a huge database of face photographs from missing people is available in Brazil[83], containing photographs that are strongly affected by artifacts such as the ones our method aims to remove. The law enforcement organiza-

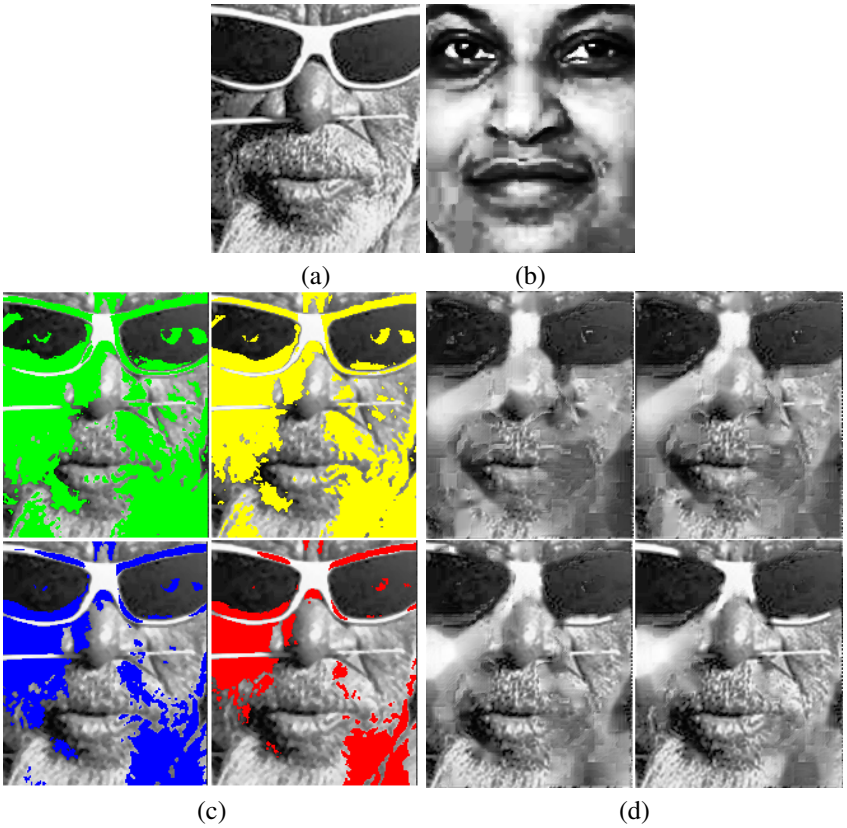


Figure 3.10: Challenging case for our method: (a) input image; (b) nearest image; (c) segmentation; (d) inpainting.

tion of São Paulo believes that removing such artifacts is a crucial step required prior to the application of existing face recognition algorithms to their database. However, as stated earlier, additional future work is needed to quantify the end-to-end improvement that our artifact removal technique adds to face recognition tasks.

3.5 CONCLUSIONS

We have presented a computational framework for automatic inpainting of facial images. Given a database of facial images which are deemed to be of good quality for recognition tasks, our method automatically identifies outlier (artifact) regions, and reconstructs these by using a mix of information present in the input image and information from the provided image database. The proposed method is simple to implement and has low computational requirements, which makes it attractive for low-cost usage contexts such as government agencies in least developed countries. We have tested our method using three real-world public image databases of missing people,

and compared our restoration results with three popular methods used in image inpainting.

Potential improvements lie in the areas of more robust segmentation using artifact-specific quality metrics and using the k nearest images ($k \geq 1$) for inpainting and actual evaluation of inpainting in a real-world face recognition set-up.

At a higher level, the lessons learned from the facial restoration application presented in this paper are twofold. First, we observe that the overall restoration quality depends mainly on the accuracy of detection of the damaged regions in the images under study. The choice of inpainting method used to restore these regions, among the four studied methods ([24, 169, 229] and our own method), has a far less pronounced influence on the final quality. As such, we will next focus mainly on researching improved methods for detecting damaged areas in our subsequent applications, and we will use an out-of-the-box inpainting method rather than focusing on improving the inpainting itself. A good candidate for such an inpainting method is [229], which is very fast, simple to implement, predictable, and gives good results for damaged regions of relatively small thickness. Secondly, as inpainting works best for relatively thin (but possibly elongated) regions, we will next focus on defect types which have this nature, rather than considering all possible defect configurations and morphologies.

This chapter is based on:

André Sobiecki, Alexandru C. Telea, Gilson Giraldi, Luiz A.P. Neves, and Carlos E. Thomaz. Low-Cost Automatic Inpainting for Artifact Suppression in Facial Images. *In 8th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISAPP*, 2013.

4 SKELETON-BASED GAP DETECTION AND FILLING IN BINARY IMAGES

The image restoration work presented in Chapter 3 has outlined that defects that are thin and elongated are relatively easy to restore by standard inpainting methods, producing overall good results, as long as such defects can be reliably detected. Additionally, we have seen that detecting such defects can be challenging. As we shall see next, such thin-and-elongated defects occur in many contexts, so their automatic detection and removal serves a variety of use-cases. In this chapter, we show how medial axes can be exploited to both detect and remove such defects from binary images, leading to applications in image segmentation, digital hair removal, and general-purpose crack filling in 2D shapes.

4.1 INTRODUCTION

Reconstruction of shapes missing internal information serves a wide range of applications, such as repairing scans of deteriorated images by closing holes, improving shape recognition and shape matching, and connecting shapes that are broken into pieces [19, 54, 64, 113, 148, 235]. Digital shapes missing internal information can often be repaired by using morphological operators as well as automatically or manually with inpainting techniques. However, morphological operations cannot discriminate between locally identical, but globally different, details, such as gaps close or on the shape boundary (which should not be filled, if we want to preserve boundary detail), and gaps deeper in the shape (which may need to be filled). Separately, inpainting requires extra work to select the areas to be inpainted, which requires manual effort or more involved, and thus more sensitive, image-analysis algorithms [64, 138, 229].

We propose a technique to detect and reconstruct digital shapes that lack some internal information, which we generically call ‘gaps’, while guaranteeing that detail information present on the apparent shape boundary is kept. For this, we first classify gaps into *detail* (that should be kept) and *errors* (that should be filled) using a global approach, based on the gap position with respect to the skeleton of a blurred version of the shape. Next, we fill error gaps using the medial axis transform associated to this skeleton. The method generically works using curve-skeletons, and is fast, simple to implement, and easy to use. We present applications for robust detail-preserving image segmentation and hair removal for dermatological images, and compare our method with several segmentation and restoration methods in the same field.

This chapter is organized as follows. In Sec. 4.2, we review related work. Section 4.3 presents our proposal. Section 4.4 presents shape restoration examples. Section 4.5 presents an application of our method to the field of dermato-imaging. Section 4.6 discusses our method. Section 4.7 concludes the chapter.

4.2 RELATED WORK

Many algorithms to segment and reconstruct digital shapes have been proposed. While an exhaustive review of the huge body of work on digital shape restoration is beyond our scope, we review three well-known approaches on segmentation and restoration of digital shapes which relate to our goals.

Filters: Filtering techniques like the median, mean, Laplacian [95], and morphological operators like erode, dilate, open, and close [100] can restore digital shapes by eliminating small-scale gaps, and are fast and simple to implement. However, most such filters work *locally*, so they cannot discriminate between gaps deep inside the shape (which we may want to eliminate) and gaps close or on the apparent shape boundary (which we want to keep, as they are part of the border structure).

Image segmentation: A key part of medical imaging is the segmentation of shapes from grayscale or color images. For example, in dermatology, one wants to segment tumors from surrounding healthy skin. Preserving all details on the segmentation border, and in the same time removing small-scale gaps and cracks inside the tumor, is essential for further automated analyses of the segmented image [69, 89, 172]. Several such segmentation methods exist [57, 80, 172, 246]. However, as we shall see later in Sec. 4.5, none of these methods can fully comply with the above two requirements.

Inpainting: Digital inpainting helps in restoring damaged parts of an image, such as reconstructing scans of deteriorated images by removing scratches or stains, or creating artistic effects [19, 64, 229]. Inpainting works well for reconstructing shapes that miss inside information, giving better results than the filter techniques mentioned above, but requires prior segmentation of the damaged region that in turn requires manual effort or more complex algorithms. For example, the DullRazor technique uses inpainting to digitally remove hairs from dermatological skin images, to make these images suitable for automatic analysis for diagnosis [138]. Although this technique works automatically, it requires a quite complex algorithm to robustly detect the hairs to be inpainted. Improvements of this technique are presented in [124], in terms of segmenting hairs of different colors, [241], for hair detection using morphological operations, and [1], for comparing different inpainting schemes applied to the segmented hairs.

Salience skeletons: Skeletons, or medial axes, are descriptors that encode both the geometry and topology of digital shapes [80, 228]. Together with their distance field to the shape boundary, they yield the so-called medial axis transform (MAT), which can be used to reconstruct shapes [203]. Simplifying the skeleton prior to reconstruction by using various importance metrics, such as the salience metric [224, 225], allows a multiscale reconstruction of shapes which removes small-scale (noise) details but keeps important details. However, eliminating gaps using such methods is difficult, as these cause complex topological changes in the skeleton.

4.3 OUR PROPOSAL

Summarizing, our major goals are to (a) eliminate thin and long gaps that (nearly) disconnect a shape into several parts. We call these *error* gaps. In the same time, we want to (b) keep all *details*, including concave indentations or gaps, present on the shape's apparent boundary.

For this, we propose a three-step process (see Fig. 4.1 top). Given a shape $\Omega \subset \mathbb{R}^{n=\{2,3\}}$ with boundary $\partial\Omega$, stored as a binary image (black=foreground, white=background), we first close *all* gaps of Ω, using morphological operations (Sec. 4.3.1). Next, we use the resulting image Ω_{oc} to classify gaps into errors and details, using a topological analysis of Ω_{oc} (Sec. 4.3.2). Finally, we use related topological mechanisms to fill gaps identified as errors in the previous step (Sec. 4.3.3). These three steps are detailed next.

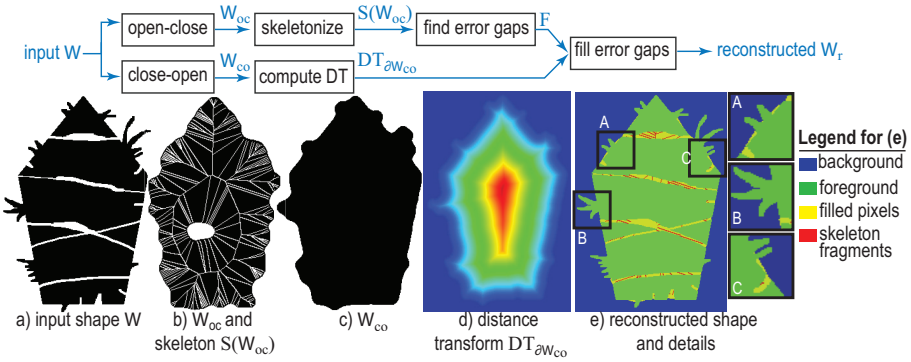


Figure 4.1: Overview of proposed method. Top: algorithm pipeline. Bottom (a-e): Example on a test image. See Sec. 4.3.

4.3.1 *Gap Closing*

To close all gaps present in our input image Ω, we use classical morphological operations. In detail, given a so-called structuring element H, we consider the dilation of Ω by H, *i.e.*, the union of copies of H_x, the element H centered at all pixels $x \in \Omega$, *i.e.*

$$\Omega \oplus H = \bigcup_{x \in \Omega} H_p. \tag{4.1}$$

Similarly, we define the erosion of Ω by H, which keeps only pixels $x \in \Omega$ where H_x fits inside Ω, *i.e.*

$$\Omega \ominus H = \{x \in \Omega | H_x \subseteq \Omega\}. \tag{4.2}$$

Next, we define the opening of Ω as erosion followed by dilation, *i.e.*

$$\Omega \circ H = (\Omega \ominus H) \oplus H, \tag{4.3}$$

and, analogously, the closing of Ω as dilation followed by erosion, *i.e.*

$$\Omega \bullet H = (\Omega \oplus H) \ominus H. \tag{4.4}$$

If we use a disk structuring element H of radius ρ, the result of applying opening and closing, denoted $\Omega_{oc} = (\Omega \circ H) \bullet H$, will close *all* holes in Ω whose thickness is smaller than ρ. Additionally, we denote the result of applying closing and opening, by $\Omega_{co} = (\Omega \bullet H) \circ H$. Both Ω_{oc} and Ω_{co} will be used next for our error-hole removal, see Secs. 4.3.2 and 4.3.3.

4.3.2 *Gap Classification*

We now use the image Ω_{oc} to classify holes into errors and detail. For this, we first compute the skeleton $S(\Omega_{oc})$. For this, we define the distance transform $DT_{\partial\Omega} : \Omega \rightarrow \mathbb{R}^n_+$ of any shape Ω as

$$DT_{\partial\Omega}(x \in \Omega) = \min_{y \in \partial\Omega} \|x - y\|. \tag{4.5}$$

The skeleton $S(\Omega)$, or medial axis, of Ω is next defined as

$$\begin{aligned} S(\Omega) = \{x \in \Omega \,|\, \exists f_1, f_2 \in \partial\Omega, f_1 \neq f_2, \\ \|x - f_1\| = \|x - f_2\| = DT_{\partial\Omega}(x)\}. \end{aligned} \tag{4.6}$$

Figure 4.1 b shows the shape Ω_{oc} (in black) for our test image in Fig. 4.1 a, and the skeleton $S(\Omega_{oc})$ (in white) for the same shape.

We now compute the fragments F of the skeleton $S(\Omega_{oc})$ that fall outside our input shape Ω, *i.e.*

$$F = \{x \in S(\Omega_{oc}) | x \notin \Omega\}. \tag{4.7}$$

We now observe that points in F are inside the error gaps, but outside the detail gaps, of Ω. Let us explain this. As noted in Sec. 4.3.1, Ω_{oc} closes both error and detail gaps of Ω, by construction. Additionally, Ω_{oc} has a boundary that is smoother than Ω (see Fig. 4.1 b). More precisely, all details on $\partial\Omega$ whose curvature is larger than $1/\rho$ are replaced by the close operation (Eqn. 4.4) by circle arcs in 2D of radius ρ. We know that branches in $S(\Omega_{oc})$ correspond to curvature maxima on $\partial\Omega_{oc}$ [203]. Since $\partial\Omega_{oc}$ is smoother than Ω, it follows that branches of $S(\Omega_{oc})$, thus also points in F, will never be located inside *boundary* gaps, or details, of $\partial\Omega$, since (1) these correspond to curvature minima along $\partial\Omega$, and (2) Ω_{oc} has an absolute curvature smaller than $\partial\Omega$. On the other hand, since the branches of $S(\Omega_{oc})$ are centered in the middle of the salient features of Ω_{oc} (by the definition of the skeleton, Eqn. 4.6), they will also be centered in the middle of the corresponding salient features of Ω (compare Figs. 4.1 b and a). This is so because the open-close operation that constructs Ω_{oc} from Ω uses a *circular* disk H, so it does not modify the local shape symmetry. Overall, it follows that points in F will be located in gaps of Ω which protrude *deep* inside this shape.

4.3.3 *Error Gap Restoration*

To close the error gaps identified by the skeleton subset F, we next proceed as follows. For each point $p \in F$, we find its closest skeleton point being in the input shape Ω

$$C(p) = \underset{q \in S(\Omega_{oc}) \cap \Omega}{\operatorname{argmin}} \|p - q\| \tag{4.8}$$

and then draw a foreground-disk with radius $DT_{\partial\Omega_{co}}(C(p))$ centered at p. This effectively fills the error gap containing p using the local shape thickness, which is equal to $DT_{\partial\Omega_{co}}(C(p))$. Let us explain this. First, we use here the distance transform of the shape Ω_{co} (see Fig. 4.1 d) obtained by the close-open operation, rather than the distance transform of Ω_{oc}, since the former first dilates, then erodes, the input shape. As such, Ω_{co} closes gaps better than Ω_{oc} (compare Fig. 4.1 c *vs* b). Thus, using $DT_{\partial\Omega_{co}}$ gives a better estimate of the apparent (filled) shape boundary within gaps than $DT_{\partial\Omega_{oc}}$. On the other hand, we use the skeleton of Ω_{oc} to detect error gaps, and initiate reconstruction from, rather than the skeleton of Ω_{co}, since Ω_{oc} does *not* close detail gaps (on the input boundary). If, in contrast, we used the skeleton of Ω_{co}, this skeleton would have branches that protrude outside Ω in boundary areas, and thus using F defined by Eqn. 4.7 would fill both error and detail gaps, which is undesired. Figure 4.2 details the above decision for a simple rectangle shape cut half-way by a vertical gap (Fig. 4.2 a). Images (b) and (c) show the results of the open-close and close-open operations, respectively. As visible, the close-open operation better fills the gap. Image (d) shows the reconstruction result if we used $DT_{\partial\Omega_{oc}}$. As visible, the gap is not well filled, since Ω_{oc} does not fill well the gap (image (b)). Image (e) shows our chosen reconstruction, where we use $DT_{\partial\Omega_{co}}$. The skeleton $S(\Omega_{oc})$, drawn in red, is of course identical. However, the disks drawn atop of the skeleton fragments F are now larger, since Ω_{co} is larger than Ω_{oc}, and thus, correspondingly, $DT_{\partial\Omega_{co}}(x) \geq DT_{\partial\Omega_{oc}}(x), \forall x \in F$.

Secondly, we note that $DT_{\partial\Omega_{co}}(p)$ is typically smaller than $DT_{\partial\Omega_{co}}(C(p))$, due to the effect of the close-open operation sequence. Hence, the gap filling done by this operation tends to 'shrink' the filled shape towards the middle of the gap. Hence, using $DT_{\partial\Omega_{co}}(C(p))$ instead of $DT_{\partial\Omega_{co}}(p)$, fills the gap by using a value which is much closer to the real shape thickness, and thus leads to a smoother reconstruction of the boundary of the filled shape across the error gap.

By the above procedure, error gaps which intersect the skeleton $S_{\Omega_{oc}}$ are thus eliminated. Figure 4.1 e shows the reconstructed shape $\Omega_r = \Omega \cup D$, where D is the set of pixels filled by the disk-drawing procedure outlined above. For clarity, we marked background pixels of Ω as blue, foreground Ω pixels as green, pixels in D as yellow, and pixels in F as red. Intuitively, our reconstruction procedures implies that gaps which cut deep inside Ω, to be precise more than half of the local thickness, get filled. In particular, gaps which completely disconnect (cut) Ω, but which are removed in Ω_{oc} by the close operation, are guaranteed to be filled. In contrast, small superficial concavities or indentations of $\partial\Omega$ that do not intersect $S_{\Omega_{oc}}$, *i.e.* are less deep than half the local thickness of Ω, are never eliminated. This way, concave boundary details of Ω are kept (see insets in Fig. 4.1 e). Separately, note that the removal of *convex* details in Ω_{oc} (as compared to Ω, see Fig. 4.1 b *vs* a), due to the open operation (Eqn. 4.3),

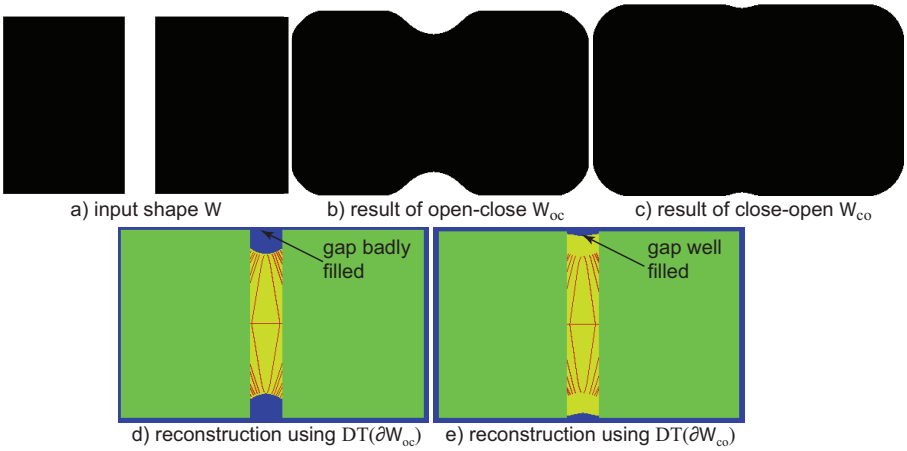


Figure 4.2: Effect of using distance transform of close-open shape Ω_{co} *vs* distance transform of open-close shape Ω_{oc}. See Sec. 4.3.3.

does not adversely affect our final result. Indeed, our gap filling only adds foreground pixels to Ω, but never eliminates pixels from it (see again insets in Fig. 4.1 e).

4.3.4 *Implementation*

For 2D skeleton extracting, we use the AFMM method [228], which computes robust, centered, pixel-wide, and topologically correct skeletons for 2D shapes of up to 1024^2 pixels in subsecond time on a modern PC. The AFMM implicitly compute, besides skeletons, the exact Euclidean distance transform $DT_{\partial\Omega}$ of the input shape. This allows us to efficiently implement accurate dilation and erosion (Eqns. 4.1 and 4.2) by simply thresholding $DT_{\partial\Omega}$ with the desired radius of the disk structuring element H. Finally, we efficiently implement the disk-drawing filling in Sec. 4.3.3 by computing the distance transform DT_F of the set F and lower-thresholding it by the values $DT_{\partial\Omega_{co}}(C(p))$ for all points $p \in F$. Both AFMM and IMA methods are implemented in C++, and do not use parallelization. Overall, on a commodity 3.5 GHz PC, our entire pipeline takes subsecond time for 2D images up to 3000^2 pixels.

4.4 RESULTS

Figure 4.3 shows several 2D restoration examples for a set of synthetic shapes, on which gaps were created manually (a,d,j) or by luminance thresholding (g,m). As visible, our gap filling eliminates the complex internal gaps, but keeps the fine boundary details, including all boundary indentations. In contrast, if we were to use a naive gap-filling by executing only an open-close operation sequence, the result would indeed fill most of the internal gaps, but also erase much of the (convex) boundary detail (images (b,e,h,k,n)). Images (m-r) show the effect of varying the structuring-element radius ρ (Sec. 4.3.1) for the input shape in image (m). Images (n) and (o) show, for illustration purposes, the open-closed shape Ω_{oc} and its skeleton $S(\Omega_{oc})$ respectively for the input image and the *rho* value for the result shown in image (q). As we increase

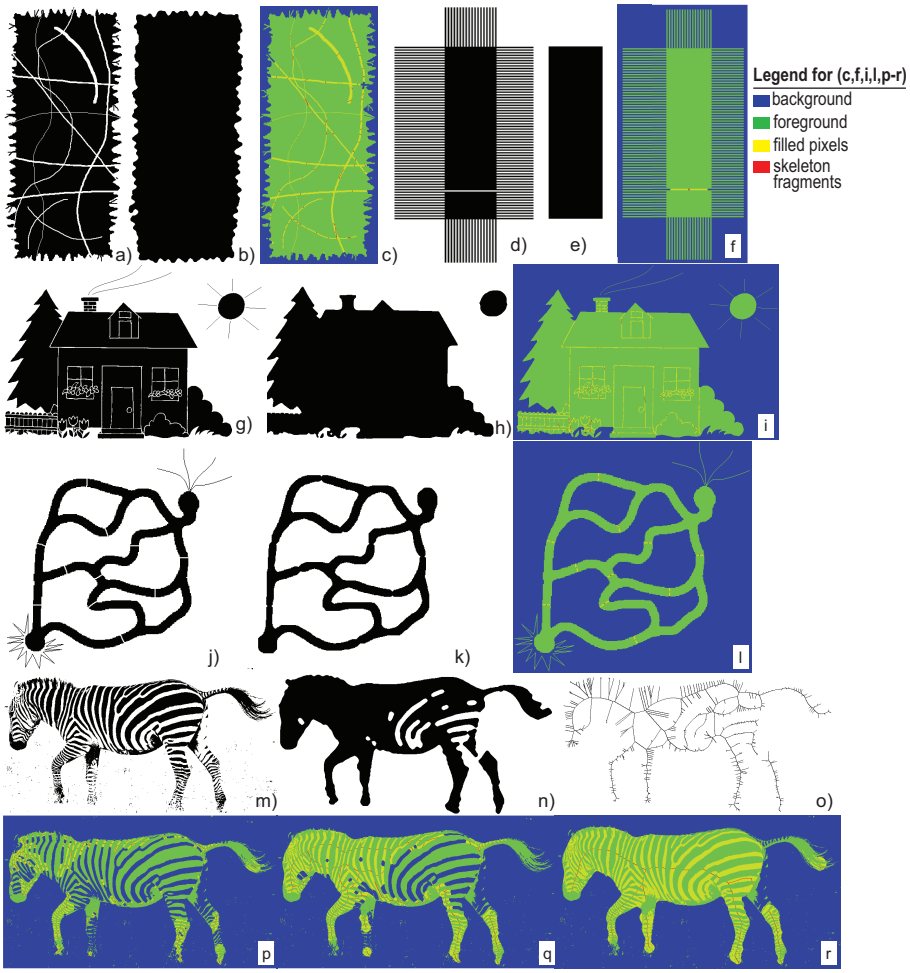


Figure 4.3: Gap filling for a set of simple shapes. (a,d,g,j,m) Input shapes Ω. (b,e,h,k,n) Result of an open-close operation. (c,f,i,l,p-r) Gap-filling results, with blue=background pixels, green=foreground pixels, yellow=filled pixels, and red=skeleton pixels. See Sec. 4.4.

ρ, larger internal gaps get progressively filled. However, fine-scale details on the apparent boundary of the input image stay preserved. As such, ρ can be effectively used to control the thickness of the internal shape gaps to be filled.

Figure 4.4 shows results for a set of binary shapes obtained from natural grayscale and color images via luminance thresholding. As expected, thresholding creates many disconnected components and/or holes and cracks within the perceived overall shapes. As for the synthetic images discussed earlier, open-close can fill most such gaps, but inherently destroys the boundary detail. In contrast, out method successfully removes gaps inside the apparent shape, but keeps most boundary detail.

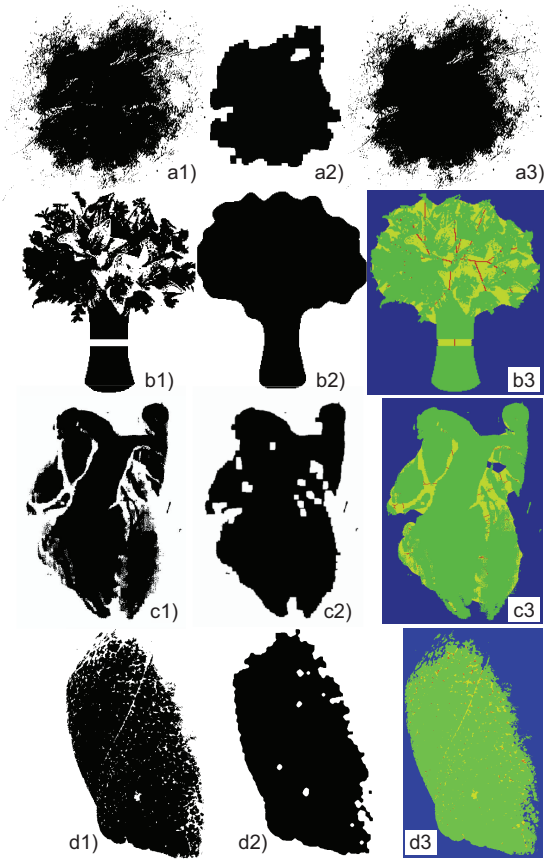


Figure 4.4: Segmentation of natural 2D images: (1) Input shapes; (2) open-close operation; (3) proposed method. See Sec. 4.4.

Figure 4.5 shows a variation of our gap-filling technique. We start like in the previous cases, *i.e.*, we produce a binary segmentation (b) by luminance-thresholding of a grayscale CT brain image (a), taken from [225]. The segmentation result shows significant noise and gaps that disconnect the apparent (black)foreground shape. Images (c-e) show the result of our gap-filling method. In contrast to the earlier examples (Figs. 4.3, 4.4), we use now the skeleton $S(\Omega_{co})$ instead of $S(\Omega_{oc})$. The effect is that

more, and larger, gaps get filled, as we increase the structuring-element radius ρ. Additionally, instead of using the full skeleton $S(\Omega_{co})$, we now threshold it by eliminating branch end-fragments that correspond to fragments of the boundary $\partial\Omega_{co}$ shorter than τ pixels, using the skeleton-importance metric proposed in [228], to which we refer for implementation details. This further smooths the boundary of the reconstructed shape (yellow pixels in Fig. 4.5). Overall, by increasing ρ and τ, we thus obtain a set of progressively simpler segmentations where larger holes are filled by smoother segments. However, as visible in images (c-e), small-scale convex boundary detail are still well preserved.

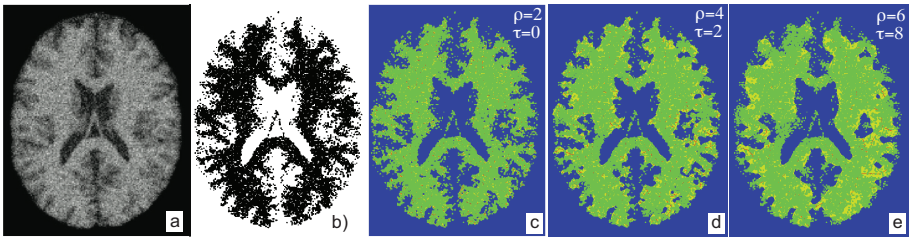


Figure 4.5: Progressively smoother segmentations (c-e) of noisy brain image segmentation (b). See Sec. 4.4.

4.5 APPLICATIONS FOR SKIN IMAGING

We next present several applications of our gap reconstruction technique in the field of skin imaging. Input images are dermatoscopic color images, of resolutions ranging from 500^2 to over 2500^2 pixels, showing skin tumors which can be either *naevi* (benign) or melanoma (malignant). Several techniques exist for the automatic prediagnosis of such tumors, based *e.g.* on the ABCD criteria [69, 89, 172]. However, to automatically evaluate such criteria, an accurate segmentation of the tumor from the surrounding skin is required. This is hard to do, as shown in Fig. 4.6, where we show the result of six known image segmentation methods on a typical dermatoscopic image(mean shift (MS) [57], gradient vector flow (GVF) [172], graph cuts (GC) [202], image foresting transform (IFT) [80], level sets (LS) [143], and dense skeletons (DS) [246]). Three types of problems occur. First, fuzzy tumor areas create strong irregularities in the segmentation boundary (GC, MS). Methods with an in-built boundary smoothness remove such problems, but create too smooth boundaries missing image details (GVF). Both these issues create problems in evaluating the ABCD criteria [172]. Secondly, occluding hairs generate boundary artifacts (MS, LS). Finally, several methods are prone to oversegmentation (MS, GC, DS). All in all, this proves that segmentation of such images is a challenging task.

Figure 4.6 j shows the result of our method, applied to a luminance-based thresholding of the input skin image (Fig. 4.6 e). As visible, thresholding generates many holes, due to both inherent color variation in the tumor, and to occluding hairs. As visible, our method generates smooth (but also detail-rich) boundaries, does not oversegment the image, and is not sensitive to occluding hairs. The gap-filling effectively removes the latter two issues, but does not remove the fine-scale detail present on the tumor

boundary. Figure 4.6 b shows, for comparison, a manual segmentation performed by a dermatologist. While this segmentation is unavoidably not identical to ours, we notice that our result is, among the set of automatic techniques considered, the closest, both in shape and extent, to the manual segmentation.

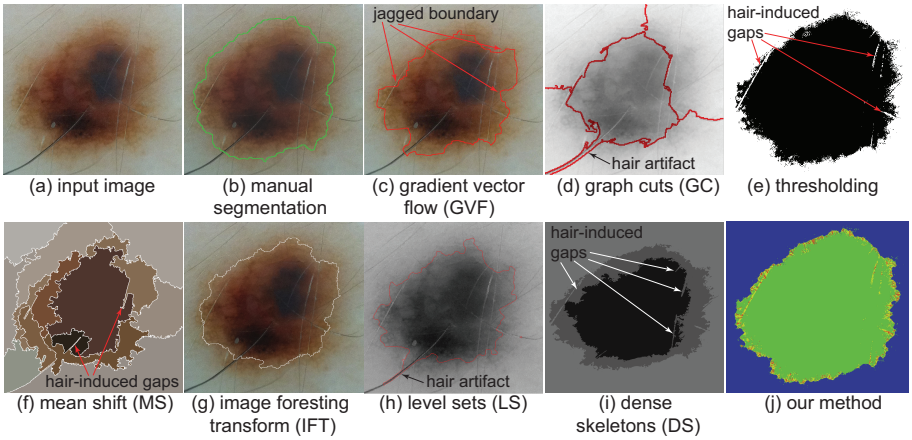


Figure 4.6: Comparison of skin image segmentation – our method (j) *vs* seven other methods (c,d,e,f,g,h,i).

Figure 4.7 shows the result of our method on five other skin-tumor images taken with different acquisition devices, of various resolutions, and showing widely different patterns that correspond to different types of skin diseases. As visible, our method improves the binary thresholding results by closing gaps inside the apparent tumor shape but keeping the tumor boundary details.

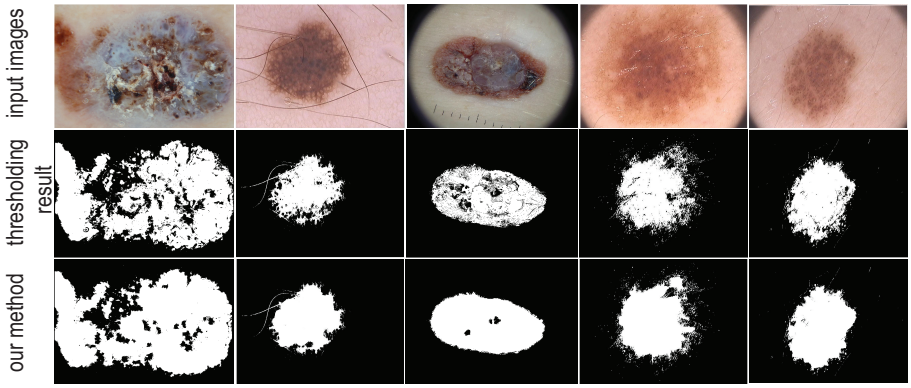


Figure 4.7: Application to clean segmentation of five dermatoscopy skin images. See Sec. 4.5.

Finally, Figure 4.8 shows the use of our method for automatic hair removal in dermatoscopic images. The input image (a) show a very complex tumor shape, which is also covered by dense hair. Applying our technique on a luminance-thresholded image (b) yields the segmentation in (c). To remove hairs, we use $\Omega_r \setminus \Omega$ (see image

(d)), *i.e.*, the difference between our result Ω_r (c) and the thresholded image Ω (b) as a mask for inpainting the input image using the method in [229]. The result (f) shows that all internal hairs have been successfully removed while preserving the tumor texture. Note that, for diagnostic image analysis, accurately segmenting the tumor *and* removing hairs inside the tumor only, is sufficient: Diagnostic analysis will next only run on the portion of the image inside the tumor, so all hairs (as well as healthy skin) outside the tumor are irrelevant. In contrast, the DullRazor method [138], with the software provided by the authors, one of the best-known hair-removal techniques in dermato-imaging, fails to remove most hairs (e), as it cannot robustly detect these, and is also considerably slower (16 seconds *vs* 0.6 seconds for our method on the platform mentioned in Sec. 4.3.2). Upon closer analysis, this is not surprising: DullRazor detects hairs using a contrast-based edge detector that works well for relatively separated constant-color hairs covering a low-contrast tumor of highly different luminance than the hairs. In our image, however, we have dense, variable-luminance, hairs that overlap a highly textured tumor, so this method fails.

Figure 4.8: Automatic hair removal in complex skin tumor images. See Sec. 4.5.

For validation, we showed our skin-image segmentation and restoration results to a dermatology specialist, with over 6 working-experience years in dermato-oncology. We posed a set of questions pertaining qualitative aspects of our results, such as perceived correctness, relative quality with respect to other similar automatic methods, and relative quality with respect to manual segmentation. The test-set included over 30 images (not all present in this chapter). The specialist responded very positively, pointing our that our segmentations are, in nearly all cases, superior in terms of boundary smoothness, detail preservation, and ease-of-use, to any known automatic method (though she indicated that manual segmentation can sometimes perform better in some fuzzy image areas). Additionally, our hair-removal method was found qualitatively better than manual alternatives and much better than DullRazor, for all complex images being tested, and equally good to these methods for the simple (low-hair-occlusion) test images. In particular, it was noted that our method has only two user-parameters (the luminance threshold level and gap-filling radius ρ), so it is much simpler to learn and use than other methods which expose more, and more complex, parameters.

4.6 DISCUSSION

Below we discuss several aspects of our method.

Strengths: The main strength of our method is the ability to close gaps which appear *inside* the input binary shape, and in the same time keep both convex and concave detail present on the apparent boundary of the same shape. The method can handle well gaps of variable position, orientation, and thickness, as demonstrated by the examples shown in this chapter. The single user-parameter to control is ρ, the maximal thickness of gaps to be closed, which has an intuitive meaning. Experiments done showed that our method can yield good-quality segmentations and restorations of dermatoscopic images, which are perceived to be better, and more useful, by domain specialists.

Limitations: The key heuristic of our method is the classification of error gaps (to be filled) as being those which intersect the skeleton of a simplified (open-close) version Ω_{oc} of the input shape Ω. The main rationale behind this heuristic is that (a) open-close simplifies Ω by removing details but keeping its main structure, and its parameter ρ allows specifying the maximal thickness of gaps to be filled (*e.g.*, allows users to specify that large gaps are important details, so should not be filled); (b) hence, the skeleton of the simplified shape Ω_{oc} captures the main part-whole structure of the original Ω; (c) gaps in the original Ω that cut this skeleton affect thus more critically the 'structure' of Ω, and thus should be removed, than gaps far from the skeleton, which can be safely regarded as details of Ω. Clearly, there can exist application contexts where step (c) of our heuristic would fail. In such cases, our method would fill less gaps than desired. However, in the over 120 examples tested so far, we have noticed that our heuristic works as expected, *i.e.* discriminates between relevant gaps (far from the shape skeleton, and thus should be kept) and error gaps (which locally cut the shape more than half, and thus should be removed) in the desired way. However, we fully agree that our heuristic needs more testing before being able to state its value in a strong sense.

Comparison: In our presented examples, we make a number of simplifications. First, we only use basic luminance thresholding for creating the input binary images for our gap-filling and restoration process. Clearly, more advanced techniques can be used. However, we chose a simple technique precisely to be able to demonstrate the added-value of our method on poor-quality input images. Secondly, the comparison against the six segmentation methods in Sec. 4.5 is surely limited, as more such methods exist. However, as stated, it is noteworthy that our (simpler) method performs qualitatively better than this range of very different segmentation methods. Thirdly, our inpainting examples only use a simple technique [228]. We do this to clearly separate the inpainting effects from the added-value of our method. End applications can immediately swap our choice for more complex, and qualitatively better, inpainting, *e.g.* [19, 64]. Finally, we note that the choice of the AFMM [228] and IMA [102] skeletons is important. One can use other 2D skeletonization methods, provided that these can produce correct multiscale, pixel/voxel-thin, centered, and connected, skeletons from any complex, noisy, disconnected shape. Unless skeletons have all above

properties, the error-gap detection (Sec. 4.3.2) and gap filling (Sec. 4.3.3) would not correctly work.

4.7 CONCLUSION

We have presented a method for reconstruction of binary 2D shapes that miss internal information in the form of holes, disconnections, and cracks. In contrast to local filtering methods, which can remove such artifacts, but also smooth our relevant details on the shape boundary, our method can successfully remove these artifacts but fully preserve the shape boundary. To achieve this, we propose a heuristic to classify gaps in terms of their position to the shape skeleton, and next remove deep gaps which intersect this skeleton. We efficiently implement our method by means of distance transform and skeletonization algorithms. Finally, we present a concrete application of our technique for robust image segmentation and hair removal in dermatological applications, and compare our results with a number of known segmentation and one restoration technique in this field.

Future work can target a number of directions. Technique-wise, our method could be extended to the area of hole and crack filling in 3D surface meshes [148, 235]. Application-wise, we can adapt our method for the reconstruction of 3D scalar and/or vector fields, such as CT and MRI scans, by removal and restoration of low-quality and low-certainty areas [67]

This chapter is based on:

A. Sobiecki, A. Jalba, D. Boda, A. Diaconeasa, and A. Telea. Gap-Sensitive Segmentation and Restoration of Digital Images. *In Proc. Theory and Practice of Computer Graphics (TPCG). Eurographics, pages 1–8,*, 2014.

5 SKELETON-BASED DIGITAL HAIR REMOVAL

Chapter 4 has presented an efficient and effective method for detecting and repairing thin-and-elongated gaps present in 2D binary images. As discussed at that point, one potential application for our method is the automatic removal of hairs from skin images, a process also known as digital hair removal (DHR), with many applications in dermatology. However, the DHR method proposed in Chapter 4 is relatively limited, as it relies on a potentially delicate manual thresholding of the input grayscale dermatoscopic image to capture the hairs, which are next removed by standard inpainting. To compete with existing state-of-the-art DHR methods, more sophisticated approaches are needed. In this chapter, we build atop of the basic gap-removal technique introduced in Chapter 4 to design a fully fledged automatic DHR method. In the design of this method, 2D medial axes will play multiple roles for the robust detection of hairs. Comparing our method with a wide range of existing DHR methods and on a large collection of real-world dermatoscopic images shows the added-value of our skeleton-based approach.

5.1 INTRODUCTION

Automatic analysis of pigmented skin lesions occluded by hair is a challenging task [51, 108]. Several *digital hair removal* (DHR) methods aim to address this by finding hairs and replacing them by plausible colors based on surrounding skin. However, DHR methods are challenged by thin, entangled, low-contrast, or thick-and-short (stubble) hairs [2, 84, 104, 124, 138, 241].

To address the above problems, we regard DHR in the context of a threshold-set representation. For this, we represent the input skin image as a set of binary images by thresholding its luminance component. Next, we adapt our gap-detection technique introduced in Chapter 4 to find potential hairs in each threshold layer. Found gaps are merged into a single hair mask, where we find actual hairs by using 2D medial axes or skeletons. Separately, to robustly detect and remove stubble hair, we propose a morphological filter geared to detecting these structures while keeping remaining image details sharp. Finally, we remove detected hairs by standard image inpainting. To implement our approach, we propose a CPU-GPU pipeline that makes the usage of complex image analysis tools such as threshold sets and medial axes practical and computationally efficient.

Compared to existing DHR methods, our main contributions are as follows:

1. We show how both elongated hairs and stubble hair can be effectively and efficiently removed, by the combined action of a morphological filter and an adapted gap-detection algorithm, respectively;

2. We demonstrate the added value of DHR for the task of robust skin-tumor *segmentation* for images containing occluding hairs;

3. We present a detailed analysis of the *scalability* of the proposed method, showing how its performance depends linearly on image size and number of threshold values, and not on the hair complexity.

We propose a method for digital hair removal from dermoscopy images, based on a threshold-set model. For every threshold, we adapt a recent gap-detection algorithm to find hairs, and merge results in a single mask image. We find hairs in this mask by combining morphological filters and medial descriptors. We derive robust parameter values for our method from over 300 skin images. We detail a GPU implementation of our method and show how it compares favorably with five existing hair removal methods, in terms of removing both long and stubble hair of various colors, contrasts, and curvature. We also discuss qualitative and quantitative validations of the produced hair-free images, and show how our method effectively addresses the task of automatic skin-tumor segmentation for hair-occluded images.

Dermatoscopy, also called dermoscopy, is the process of examination of skin lesions with a device called a dermatoscope. In usual scenarios, the dermatoscope is used to acquire a high-resolution image of a limited skin area, typically one to a few square centimeters, which corresponds to the diameter of most skin tumors. For this, the dermatoscope uses a high-resolution camera (typically over one megapixel), and special lighting consisting of polarized and/or non-polarized light sources surrounding the camera aperture. To reduce the effect of ambient lighting, a shade screen is mounted around the camera and lights, and the entire device is applied in direct contact with the skin area to be imaged. Certain models also use a special gel between the skin and camera lens to ensure better contact. The acquired images are next studied for diagnostic purposes, either manually by dermatologists, or using various automated image analysis tools.

The structure of this chapter is as follows. Section 5.2 reviews related work on digital hair removal. Section 5.3 details our method. Section 5.4 presents its implementation. Section 5.5 compares our results with five DHR methods. Section 5.6 discusses our method. Section 5.7 concludes the chapter.

5.2 RELATED WORK

In the past decade, many DHR methods have been proposed. The most known ones are outlined next. DullRazor, the first and arguably most famous, finds dark hairs on light skin by morphological closing using three structuring elements that model three line orientations [138]. Different morphological operators were similarly used in [165, 192]. Prewitt edge detection [124] and top-hat filtering [241] help finding low-contrast or thin-and-curled hairs. Once detected, hairs can be removed by bilinear [138] or PDE-based inpainting [240]. Huang *et al.* find hairs by multiscale matched filtering and hysteresis thresholding and remove these by PDE-based inpainting [104]. However, this method is quite slow (minutes for a typical dermoscopy image). Abbas *et al.* find hairs by a derivatives-of-Gaussian (DoG) filter [1, 2]. However, this method has many parameters whose setting is complex.

While filters such as the above ones succeed in finding *locally* linear high-contrast structures, assessing that such structures form together a long-and-thin object requires *global* analyses. Unless this is done, many false-positives will be found, *e.g.* very

short disconnected hair-like fragments of various orientations. Their removal affects the skin texture, which may next adversely affect the use of such texture for image analysis and classification. To address this, VirtualShave finds hairs by top-hat filtering, like [241], and uses three density, sphericity, and convex-hull-sphericity metrics to separate true positives (hairs) from other high-contrast details (false positives) [84]. Finding other elongated objects such as arterial vessels and fibers is also addressed by path opening methods [56] and grayscale skeletons [63]. The last method also permits filling thin gaps similar to our hairs. However, such approaches have not been yet demonstrated for DHR aims.

Table 2 captures several aspects of the above DHR methods. As visible, there is little comparison across methods. One salient aspect in this overview is that existing methods are validated on relatively small image sets and/or do not have public implementations on which other researchers could test them (except [104, 138]). As such, exhaustive comparison of existing DHR methods is hard. For our proposed DHR method described next, extensive comparison with other methods and on large image sets will be a main goal.

5.3 PROPOSED METHOD

Most DHR methods find hairs by local luminance analysis (see Tab. 2, column 2). Such methods often cannot to find hairs that have *variable* color, contrast, thickness, or crispness across an image. Hence, our main idea is to perform a conservative hair detection at all possible luminance values. For this, the following pipeline is proposed. First, we convert the input image into a luminance threshold-set representation (Sec. 5.3.1). For each threshold layer, we find thin hair-like structures using a morphological gap-detection algorithm (Sec. 5.3.2). Potential hairs found in all layers are merged in a mask image, which we next analyze to remove false-positives (Sec. 5.3.3). True-positive hairs are next removed by using a classical image inpainting algorithm (Sec. 5.3.4). Finally, we detect short and relatively thick hairs (stubble) by morphological analysis and remove these by the same image inpainting method used for long hairs (Sec. 5.3.5). These steps are discussed next.

5.3.1 *Threshold-set Decomposition*

We reduce color images first to their luminance component in HSV space. Next, we compute a threshold-set model of the image [246]: Given a luminance image $I : \mathbb{R}^2 \rightarrow \mathbb{R}_+$ and a value $v \in \mathbb{R}_+$, the threshold-set $T(v)$ for v is defined as

$$T(v) = \{\mathbf{x} \in \mathbb{R}^2 | I(\mathbf{x}) \geq v\}. \tag{5.1}$$

For n-bits-per-pixel images, Eqn. 5.1 yields 2^n layers $T_i = T(i), 0 \leq i < 2^n$. We use $n = 8$ (256 luminances), in line with the color resolution of typical dermoscopic images. Note that $T_j \subset T_i, \forall j > i$, *i.e.* brighter layers are 'nested' in darker ones. If $I(\mathbf{x}) \neq i, \forall \mathbf{x} \in \mathbb{R}^2$, we find that $T_i = T_{i+1}$. In such cases, we simply skip T_i from our threshold-set decomposition, as it does not add any information. Our decomposition $\{T_i\}$ will thus have at most 2^n layers.

Table 2: Comparison of existing digital hair removal methods.

Method	Hair detector	Inpainting by	Compared with	# test images	Implementation
DullRazor [138]	generalized morphological closing	bilinear interpolation	–	5	available (binary)
Huang *et al.* [104]	multiscale matched filters	median filtering	DullRazor	20	available (binary)
Fiorese *et al.* [84]	top-hat operator	PDE-based [23]	DullRazor	20	not available
Xie *et al.* [241]	top-hat operator	anisotropic diffusion [174]	DullRazor	40	not available
E-shaver [124]	Prewitt edge detector	color averaging	DullRazor	5	not available
Abbas *et al.* [2]	derivative of Gaussian	coherence transport [35]	DullRazor, Xie *et al.* [241]	100	not available
Our method	long hair detection by multiscale skeletons; stubble detection by morphological operators	fast marching method [229]	DullRazor, Xie *et al.* [241], Huang *et al.* [104], Fiorese *et al.* [84] Abbas *et al.* [2]	**over 300**	**available** (source code, binary)

5.3.2 *Potential Long Hair Detection*

To find typical (long) hairs, we detect thin-and-long shapes in each layer T_i by adapting the gap-detection method introduced in Chapter 4, as follows.

Original gap-detection: Given a binary shape $\Omega \subset \mathbb{R}^2$ with boundary $\partial\Omega$, we compute the *open-close* image $\Omega_{oc} = (\Omega \circ H) \bullet H$ and *close-open* image $\Omega_{co} = (\Omega \bullet H) \circ H$. In detail, given a so-called structuring element H, which is a disk in our case, we consider the dilation of Ω by H, *i.e.*, the union of copies of H_x (H centered at all pixels $x \in \Omega$), *i.e.*

$$\Omega \oplus H = \bigcup_{x \in \Omega} H_x. \tag{5.2}$$

Similarly, we define the erosion of Ω by H, which keeps only pixels $x \in \Omega$ where H_x fits inside Ω, as

$$\Omega \ominus H = \{x \in \Omega | H_x \subseteq \Omega\}. \tag{5.3}$$

Next, we define the opening of Ω as erosion followed by dilation, *i.e.*

$$\Omega \circ H = (\Omega \ominus H) \oplus H, \tag{5.4}$$

and, analogously, the closing of Ω as dilation followed by erosion, *i.e.*

$$\Omega \bullet H = (\Omega \oplus H) \ominus H. \tag{5.5}$$

In both Ω_{oc} and Ω_{co}, small gaps of the input image Ω get filled; yet, Ω_{co} has more gaps filled than Ω_{oc}, but also fills shallow concavities (dents) along $\partial\Omega$.

Next, we compute the skeleton or medial axis $S_{\Omega_{oc}}$ of the shape Ω_{oc}. Considering the distance transform $DT_{\partial\Omega} : \mathbb{R}^2 \to \mathbb{R}_+$ given by

$$DT_{\partial\Omega}(\mathbf{x} \in \Omega) = \min_{\mathbf{y} \in \partial\Omega} \|\mathbf{x} - \mathbf{y}\|, \tag{5.6}$$

the skeleton S_Ω of Ω is next defined as

$$S_\Omega = \{\mathbf{x} \in \Omega | \exists \mathbf{f}_1, \mathbf{f}_2 \in \partial\Omega, \mathbf{f}_1 \neq \mathbf{f}_2, \|\mathbf{x} - \mathbf{f}_1\| = \|\mathbf{x} - \mathbf{f}_2\| = DT_{\partial\Omega}(\mathbf{x})\} \tag{5.7}$$

where $\mathbf{f}_1$ and $\mathbf{f}_2$ are the contact points with $\partial\Omega$ of the maximally inscribed disc in Ω centered at $\mathbf{x}$. From $S_{\Omega_{oc}}$, the algorithm removes branch fragments that overlap with Ω, yielding a set $F = S_{\Omega_{oc}} \setminus \Omega$ that contains skeleton-fragments located in thin *gaps* that cut *deeply* inside Ω. To find all pixels in the gaps, the proposed method convolve the pixels $\mathbf{x} \in F$ with disk kernels centered at the respective pixels and of radius equal to $DT_{co}(\mathbf{x})$. As shown in Chapter 4, this produces an accurate identification of deep-and-thin indentations, or *gaps*, in Ω, while ignoring pixels in shallow dents along $\partial\Omega$.

Hair-detection: We observe that, in a binary image with hairs in foreground, hairs are gaps of surrounding background. We next aim to find robustly hairs in all layers T_i. For this, several changes to our gap-detection method from Chapter 4 are needed.

First, we note that the original gap-detection uses $DT_{\Omega_{co}}$ as disk-radius values for gap-filling as they argue that Ω_{co} closes more gaps than Ω_{oc}, supported by the observation that $DT_{\Omega_{co}}(\mathbf{x}) \geq DT_{\Omega_{oc}}(\mathbf{x}), \forall \mathbf{x} \in F$. Yet, for our hair-removal context, using $DT_{\partial\Omega_{co}}$ on every layer T_i, and next merging gaps into a single hair-mask, results in too many areas being marked as hair. The resulting mask proves to be too dense – thus, creates too many false-positive hairs for our next filtering step (Sec. 5.3.3). Using the smaller $DT_{\partial\Omega_{oc}}$ as disk radius prevents this problem, but fails to find many hair fragments – thus, creates too many false-negatives. To overcome these issues, we propose to use a linear combination of $DT_{\partial\Omega_{oc}}$ and $DT_{\partial\Omega_{co}}$. For this, we define a set of pairs disk-centers $\mathbf{x}$ and corresponding disk-radii ρ as

$$D_\lambda = \{\big(\mathbf{x}, \rho = (1-\lambda)DT_{\partial\Omega_{co}}(\mathbf{x}) + \lambda DT_{\partial\Omega_{oc}}(\mathbf{x})\big) \,|\mathbf{x} \in F\} \tag{5.8}$$

where $\lambda \in [0,1]$ gives the influences on the disk radius of $DT_{\partial\Omega_{oc}}$ and $DT_{\partial\Omega_{co}}$ respectively. A value of $\lambda = 0.2$, found empirically (see Sec. 5.6), avoids finding too many gaps (false-positives), while also preventing missing too many hairs (false-negatives).

Let D be the union of pixels in all disks described by D_λ. We next find the gaps G that potentially describe hairs as the difference

$$G = D \setminus \Omega. \tag{5.9}$$

We apply Eqn. 5.9 to compute a gap G_i from every shape $\Omega_i := T_i$. Next, we merge all resulting gaps G_i together into a single hair-mask image $M = \bigcup_{i=0}^{n} G_i$.

Morphological closing finds only hairs darker than skin. To find hairs lighter than skin, we replace closing by morphological opening. Having the dark-hair and light-hair masks M^d and M^l, we can next either combine the two or select one mask to use further. We observed in virtually all our test images that dark and light hairs do not occur together. So, we use next the mask $M \in \{M^d, M^l\}$ that most likely contains hairs, *i.e.*, which maximizes the length of the longest skeleton-branch in $S_{\partial M}$. For example, for the image in Fig. 5.1 a, which has mainly dark hairs, our method will select to use the mask $M := M^d$ (Fig. 5.1 b).

5.3.3 *False Positive Elimination*

Since we search for gaps on *every* threshold-level, we find more gaps than traditional approaches, *e.g.* [104, 124, 138, 241]. Filtering out 'false positives' (gaps unlikely to be hairs), is thus necessary. We achieve this in four steps, outlined below.

Component detection: First, we extract from M all 8-connected foreground components $C_i \subset M$. We skip components less than 1% of the size of image M, as these cannot possibly be elongated hairs. Remaining components are analyzed next to see if they are hairs or not.

Hair skeletons: Hair fragments are long and thin. To measure such properties on our components C_i, we use their skeletons $S_{\partial C_i}$. Yet, components C_i may have jagged borders, due to input-image noise, shadows, or low resolution (Fig. 5.1 b), so $S_{\partial C_i}$ have many short spurious branches. We discard these and keep each component 'core' by pruning each $S_{\partial C_i}$ as in [228]: From $S_{\partial\Omega}$, we produce a skeleton $S^{\tau}_{\partial\Omega}$ which keeps only points in $S_{\partial\Omega}$ caused by details of $\partial\Omega$ longer than τ. By making τ proportional to the component's boundary length $\|\partial C_i\|$, we ensure that longer branches are pruned more

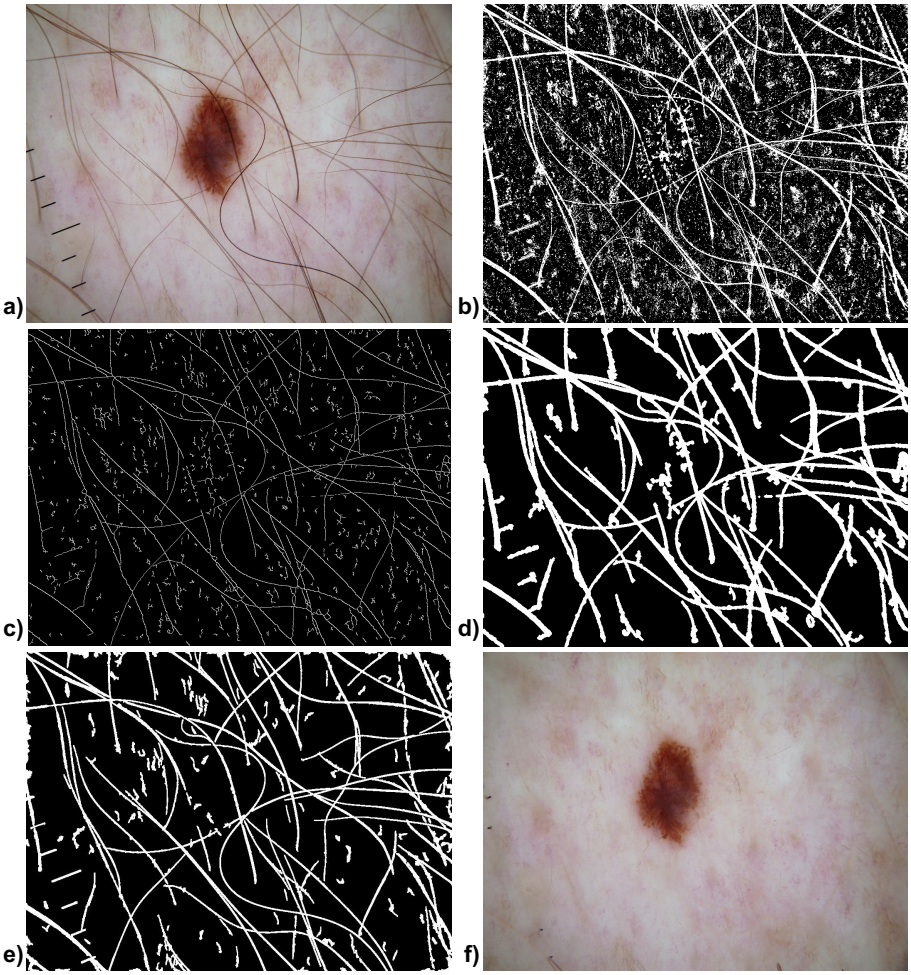


Figure 5.1: a) Input image. b) Full hair mask M. c) Simplified mask skeleton S_M^τ. d) Filtered mask M^f. e) Mask created by [104]. f) Inpainted hair using M^f.

than shorter ones. We also impose a minimum τ_{min} to discard tiny spurious fragments, and a maximum τ_{max} to preserve large branches. Hence, the pruning parameter τ for a component C_i is

$$\tau = \max(\tau_{min}, \min(\|\partial C_i\| \cdot \mu, \tau_{max})) \tag{5.10}$$

where $\mu \in [0,1]$ is used as a scaling parameter and $\|\partial C_i\|$ denotes the boundary length of C_i, in pixels. Figure 5.1 c shows the simplified skeleton $S^{\tau}_{\partial M}$ obtained from the mask M in Fig. 5.1 b.

Hair detection: In classical DHR, finding if a component is thin-and-long is done by *e.g.* (a) fitting lines in a finite number of orientations and checking the length of the longest such line [138]; (b) using principal component analysis to find if the major-to-minor eigenvalue ratio exceeds a threshold [137]; and (c) computing an elongation metric comparing a component's skeleton-length with its area [241]. Xie *et al.* argue that (a) and (b) are limited, as they favor mainly straight hairs and yield false-negatives for curled hairs [241]. They alleviate this by an elongation metric equal to the ratio of the area $\|C_i\|$ to the squared length of the 'central axis' of C_i. However, they give no details on how this central-axis (and its length) are computed. In particular, for crossing hairs, *i.e.*, when the skeleton of C_i has multiple similar-length branches, multiple interpretations of the notion of a 'central axis' are possible. We also found that (c) also yields many false-negatives, *i.e.*, marks as hair shapes which do not visually resemble a hair structure at all.

To address such issues, we propose a new metric to find if a thin-and-long shape is likely a hair. Let $J_i = \{\mathbf{x} \in S^{\tau}_{\partial C_i}\}$ be the set of junctions of $S^{\tau}_{\partial C_i}$, *i.e.*, pixels where at least three $S^{\tau}_{\partial C_i}$ branches meet. If the maximum distance $d_{max} = \max_{\mathbf{x}\in J_i, \mathbf{y}\in J_i, \mathbf{x}\neq\mathbf{y}} \|\mathbf{x}-\mathbf{y}\|$ between any two junctions is small, then C_i is too irregular to be a hair. We also consider the average branch-length between junctions $d_{avg} = \|S_{\partial C_i}\|/\|J_i\|$, *i.e.*, the number of skeleton-pixels divided by the junction count. If either $d_{max} < \delta_{max}$ or $d_{avg} < \delta_{avg}$, then C_i has too many branches to be a thin elongated hair (or a few crossing hairs), so we erase $S^{\tau}_{\partial C_i}$ from the skeleton image. Good preset values for δ_{max} and δ_{avg} are discussed in Sec. 5.6.

Mask construction: We construct the final mask M^f that captures hairs by convolving the filtered skeleton-image (in which false-positives have been removed) with disks centered at each skeleton-pixel $\mathbf{x}$ and of radius equal to $DT_{\partial M}(\mathbf{x})$. Figure 5.1 d shows the mask M^f corresponding to the skeleton image in Fig. 5.1 c. Comparing it with the hair-mask produced by [104] (Fig. 5.1 e), we see that our mask succeeds in capturing the same amount of elongated hairs, but contains fewer small isolated line-fragments (thus, has fewer false-positives).

5.3.4 *Long Hair Removal*

We remove the detected thin-and-long hairs by using classical inpainting [229] on the hair-mask M^f. To overcome penumbras (pixels just outside M^f are slightly darker due to hair shadows), which get smudged by inpainting into M^f, we first dilate M^f isotropically by a 3×3 square structuring element. This tells why hairs in M^f in

Fig. 5.1 d are slightly thicker than those in Fig. 5.1 b. Figure 5.1 f shows our final DHR result.

5.3.5 *Stubble Detection and Removal*

While the above four steps effectively find and remove thin-and-long hairs, they can easily miss thick-and-short hairs (stubble). Such hairs appear in dermoscopy images, *e.g.* in situations where the lesion area was shaved for a better image acquisition. To remove stubble, we propose a post-processing filter on the images generated by the inpainting step (Sec. 5.3.4), as follows.

Let I^{inp} be the output of the long-and-thin hair inpainting step (note that this is a color image). We compute I_{OC} and I_{CO} by applying open-close and close-open operators respectively to the red, green, and blue channels of I^{inp}. We next compute the absolute difference (grayscale) images $I^d_{OC,I}$ and $I^d_{CO,I}$ of I_{OC} and I_{CO} respectively with the input image I^{inp}. This is related, but not identical to, the top-hat and bottom-hat transforms, where the difference between an image and its opening, respectively closing, is taken. As shown in Chapter 4, using the open-close and close-open images instead of basic openings and closings yields better results for gap detection scenarios like our DHR context.

Similarly to the mask construction for long hair detection in the presence of hairs darker, respectively lighter, than skin, we next choose to use the difference image $I^d \in \{I^d_{OC,I}, I^d_{CO,I}\}$ which has the largest intensity value summed over its pixels. This selects $I^d := I^d_{OC,I}$ for images having predominantly dark stubble, and $I^d_{CO,I}$ for images having predominantly light-colored stubble.

We next threshold I^d, normalized to $[0,255]$, into a binary stubble mask M^s by using a threshold value defined as

$$t = \frac{\max_{\mathbf{x}\in I^d} I^d(\mathbf{x})}{\gamma}, \tag{5.11}$$

where γ is a scaling factor. Setting $\gamma = 2$ reliably selected stubble hair in all out test images. Note that the normalization applied to I^d prior to thresholding has the effect of a contrast enhancement operation, which makes low-contrast hairs more visible and thus selects them more reliably in the mask M^s. We finally dilate M^s isotropically by a 3×3 square structuring element, and remove stubble from I^{inp} by inpainting it over M^s, analogously to the long-and-thin hair removal (Sec. 5.3.4).

Figure 5.3 shows the effects of our stubble removal filter. As visible, stubble is still present in the output of the long-and-thin DHR algorithm pass (Figs. 5.3 a,c), while it is well detected and removed by our stubble removal filter (Figs. 5.3 b,d). The stubble filter also removes other small-scale line-like details, such as the ruler annotations introduced by the dermatoscope (Figs. 5.3 a,c top-row). The two filters (long-and-thin and stubble removal) assist each other, as follows. If an image contains only thin-and-long hair, or only stubble, only one of the filters will actively change the image, while the other one will act as a pass-through. If, however, an image contains both hair types, applying the thin-and-long hair filter *before* stubble removal has the desirable effect of making stubble detection much easier, as complicated structures of entangled hair are already removed.

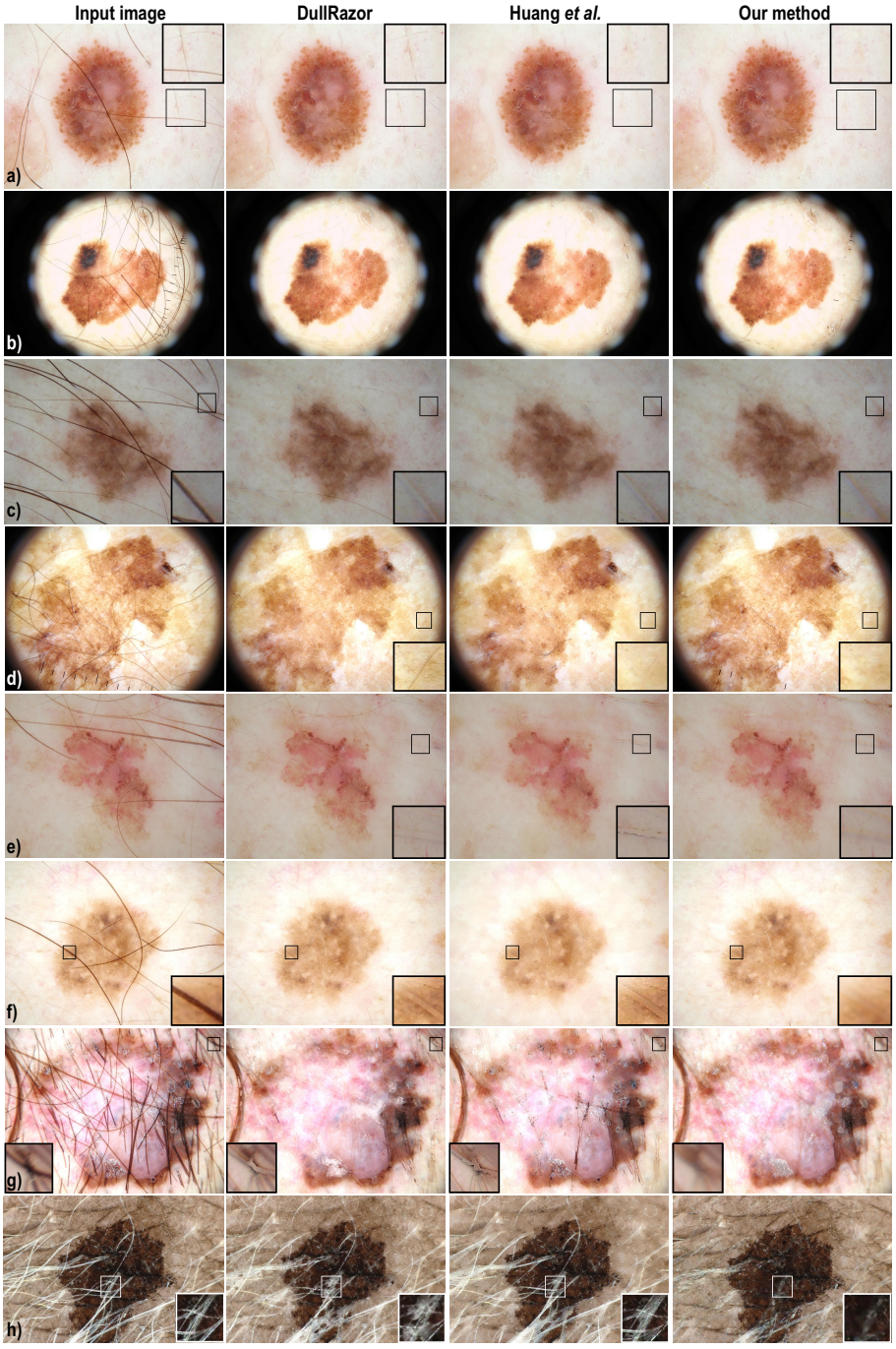


Figure 5.2: Comparison of our method with DullRazor [137] and Huang *et al.* [104]. Insets show details.

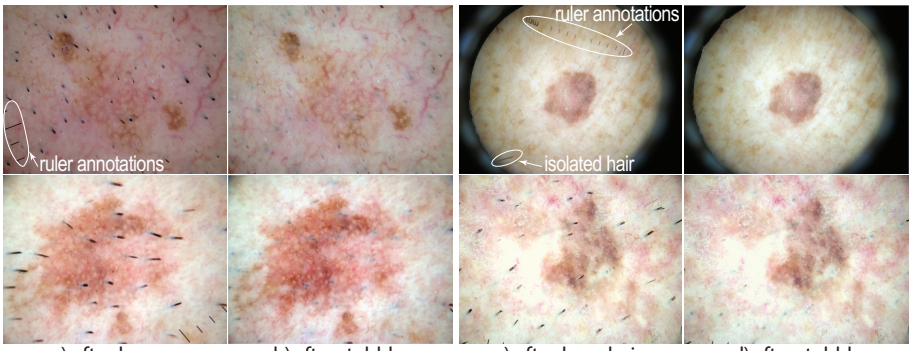


Figure 5.3: Stubble removal filter (b,d) filters out stubble from the output of the long-hair removal pass (a,b).

5.4 IMPLEMENTATION

The most expensive part of our method is computing M, which requires distance transforms and skeletons from up to 256 binary images (Sec. 5.3.2). As these images can be over 1024^2 pixels for modern dermoscopes [87], processing a single image must be done within milliseconds to yield an acceptable speed. For this, we use the Parallel Banding Algorithm (PBA) for exact Euclidean distance transforms (EDTs) in [41]. A simple modification of this method allows us to compute dilations and erosions (by thresholding the distance transform with the radius of the disk structuring element) and simplified skeletons (by implementing the boundary-collapse in [228]). Computing the skeleton of a shape Ω by [228] only requires the identity of the closest point of $\partial\Omega$ for any point in Ω, or the so-called feature transform of $\partial\Omega$. This information is directly provided by the PBA method, so computing skeletons has virtually no additional cost atop of the day computation.

Hair masks M (Sec. 5.3.2) are also computed on the GPU. First, the grayscale image is copied from CPU memory to VRAM, after which each threshold is processed sequentially on the GPU. For each threshold i, the open-close and close-open images are computed from the binary shape Ω_i. Erosions and dilations are computed by thresholding the distance transforms $DT_{\partial\Omega_i}$ and $DT_{\partial\bar{\Omega}_i}$ with the radius of the desired disk structuring element. Open-close images are computed by optimizing $\Omega \oplus H \ominus H \ominus H \oplus H$ into $\Omega \oplus H \ominus H' \oplus H$, where H' has double the radius of H. Similar optimizations are done for close-open. The distance transforms $DT_{\partial\Omega_i}$ and $DT_{\partial\bar{\Omega}_i}$ are subsequently used to compute the radii ρ of the disks D_λ (Eqn. 5.8). Next, for each skeleton pixel $\mathbf{x}$ located in a gap (set F in Sec. 5.3.2), we launch a thread to draw a disk of radius ρ centered at $\mathbf{x}$, which yields the image D. As F does not contain many pixels (hundreds at most), computing D by disk drawing is efficient. The final step in processing a layer is to compute the gap mask G_i by finding all disk pixels outside Ω_i and marking their locations directly in the hair mask M.

After all layers have been processed, the hair mask M is copied from VRAM back to CPU memory. The latter steps of the algorithm – connected component detection,

done with union-find [180]; skeleton-based filtering; stubble filtering; and hair inpainting [229] – are implemented in C++ on the CPU, as they are only performed once and thus not performance-critical as the per-layer computations are.

We also ran our method on multi-GPU machines by starting k MPI processes for k GPUs. Each process $p \in \{0,\ldots,k\}$ does gap-detection on a subset of the threshold-set by launching CUDA threads to parallelize gap-detection at image block level [41]. The k separate masks $M_p, 1 \leq p \leq k$ are merged by process 0 into a single mask M, after which the algorithm continues on the CPU like outlined above.

Memory-wise, our entire implementation requires only 12 floating-point buffers of the size of the input image I, seven by PBA [41] to compute EDTs and skeletons, and five for the remaining algorithm steps. This allows processing megapixel-size images on even the lowest-range CUDA-capable GPUs having 128 MB VRAM. Feature-wise, we only use CUDA 1.1 capabilities, which makes our implementation run on virtually all existing Nvidia cards, including low-end ones. C++ source code of our full method is available openly for download at [126]. Additional details regarding computational speed are given in Sec. 5.6.

5.5 RESULTS AND COMPARISON

Material: We have tested our method on over 300 skin images. These cover a wide range of skin lesions; hair thickness, color, length, and density; image resolution (between 400^2 and 2448×3264 pixels *i.e.* full Handyscope resolution [87]); and skin pigmentation. Images were acquired by several types of dermoscopes, by three unrelated research groups. Additionally, we tested our DHR method on the skin images reported in the papers of [2, 84, 104]. Some of our test images contain no hair (see *e.g.* Fig. 5.6c discussed further in this section); they let us see how well can we avoid false positives. This is important, as removing non-hair details may affect subsequent analyses [2, 104].

Methods: We compared our results with five DHR methods, as follows: Where an implementation of the method to compare with was available [105, 138], we ran our full image-set through it. For the other methods [2, 84, 241], we processed images from the respective papers by our method and compared our results with the ones in the respective papers.

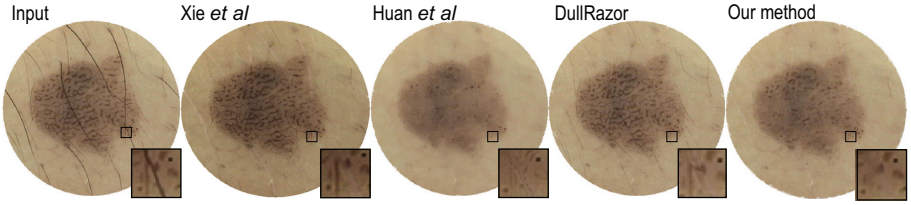


Figure 5.4: Comparison between Xie *et al.* [241], Huang *et al.* [104], DullRazor, and our method. Input image from Xie *et al.*

Results: Compared to DullRazor and Huang *et al.* [104] (Fig. 5.2), we see that DullRazor cannot remove low-contrast hairs (a,d); and both methods create undesired

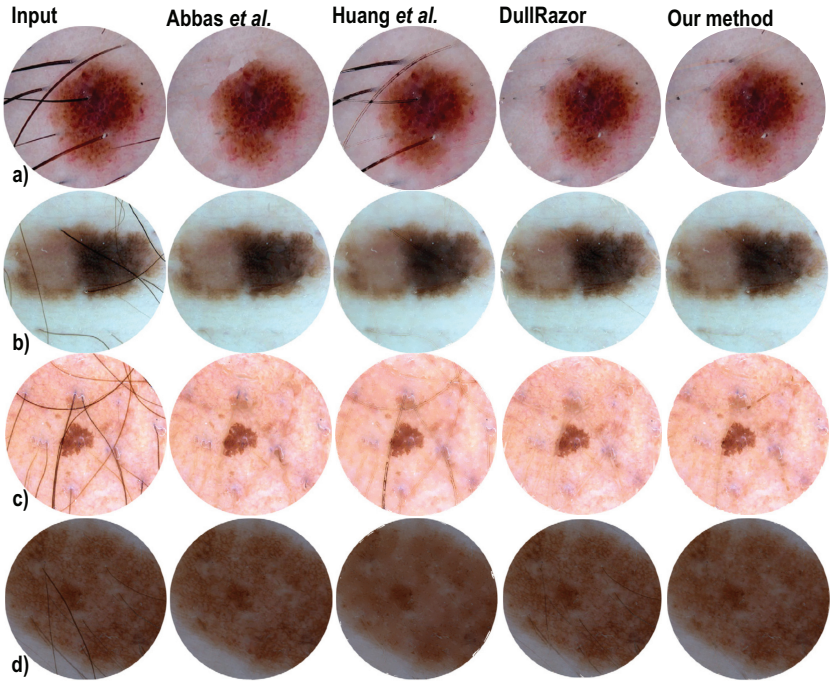


Figure 5.5: Comparison between Abbas *et al.* [2], Huang *et al.* [104], DullRazor, and our method. Input images from Abbas *et al.*

'halos' around removed hairs (c,f;e,f). Images (g,h) show two complex lesions, with hair of variable tints, opacity, thickness, and density. For (g), we create less halos around removed hairs than both DullRazor and Huang *et al.* For (h), our method removes considerably more hair than both methods. Figure 5.6 shows supplementary comparisons for four complex images. Image (a) contains several crossing and very low-contrast hairs. We see that DullRazor can remove several, but no all, such hairs. Also, both DullRazor and Huang *et al.* create high-contrast edges from small non-hair pigmentation details, such as the ones shown in the insets, an effect of their use of local edge-detection filters. In contrast, our method removes most such hairs and also correctly preserves pigmentation details. Image (b) contains a few hair-like details (dermoscope markers in top-left corner), but no hairs. The markers are successfully removed by all methods, including ours. Image (c) shows a few crossing very low-contrast hairs. DullRazor cannot remove these. Huang *et al.* remove them, but also significantly blurs the skin line-like pattern. Our method removes the hairs and keeps the skin pattern, since its line-like structures are not sufficiently long to be seen as hairs by our skeleton-based analysis (Sec. 5.3.3). Finally, image (d) contains no hairs, but a number of bubbles formed by contact gel placed between the dermoscope lens and the skin for better contact (see inset). Like hairs, such artifacts are not part of the tumor texture proper, and can confuse subsequent image analyses, and as such should be removed, if possible. We see that both DullRazor and Huang *et al.* cannot remove these structures. In contrast, our method detects the thin-and-curly bubble structure and removes most of it.

Figure 5.4 compares our results with Xie *et al.* [241] and Huang *et al.* We remove more hairs than Xie *et al.*, but also remove a small fraction of the skin. Huang *et al.* removes all hairs but also massively blurs out the skin. This is undesirable, since such patterns are key to lesion analysis.

Figure 5.5 compares our method with Abbas *et al.* [2], Huang *et al.*, and DullRazor, on a set of images from Abbas *et al.* These images cover a wide gamut of skin and lesion pigmentations and hair thicknesses and contrasts. Our method shows comparable results to Abbas *et al.* Huang *et al.* has issues with thick hairs (a); creates undesired hair halos (c); and also blurs the fine-grained typical network texture present in image (d). This last effect is highly undesired, since typical network texture is, among other image features, an important indicator for the malignancy assessment of skin tumors [132]. Separately, we see that DullRazor cannot remove most of the low-contrast hairs for the dark lesion (d).

Compared to Fiorese *et al.* [84], we show a similar ability in removing both stubble and elongated hairs (Fig. 5.7). For images (a,b), Fiorese *et al.* strikingly changes the hue of the input image, which is undesired, as this can affect both manual and automatic lesion assessment. Our method correctly preserves the hue of the image. For the same images, showing both stubble hair (Fig. 5.7 a) and long curly hair (Fig. 5.7 b,c), our method performs very similarly to DullRazor and Huang *et al.*, and also creates less halos around removed hairs (see insets).

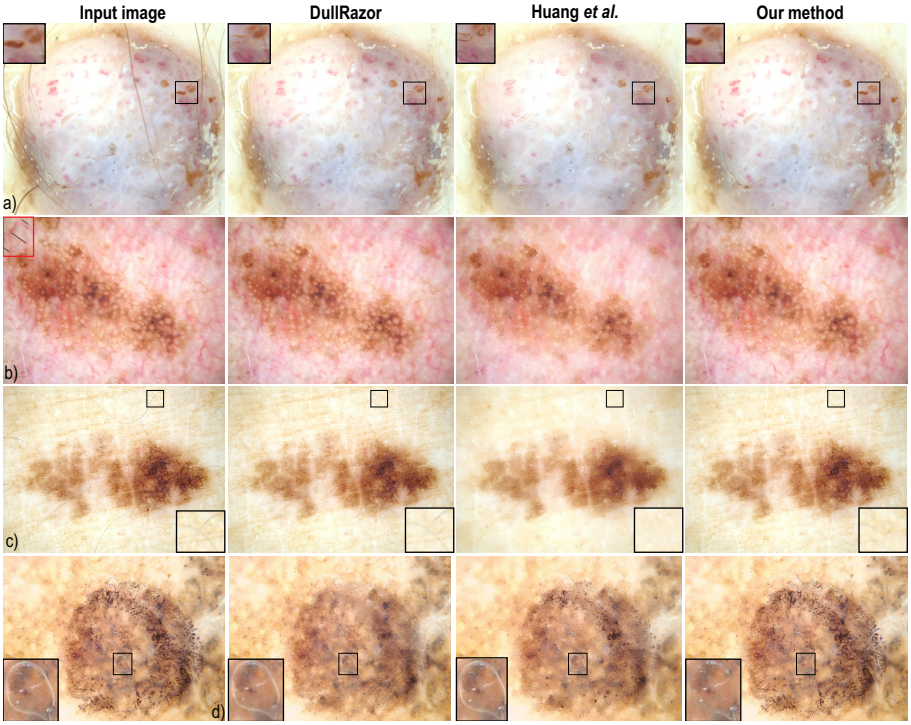


Figure 5.6: Comparison of our method with DullRazor [137] and Huang *et al.* [104] for several complex skin images.

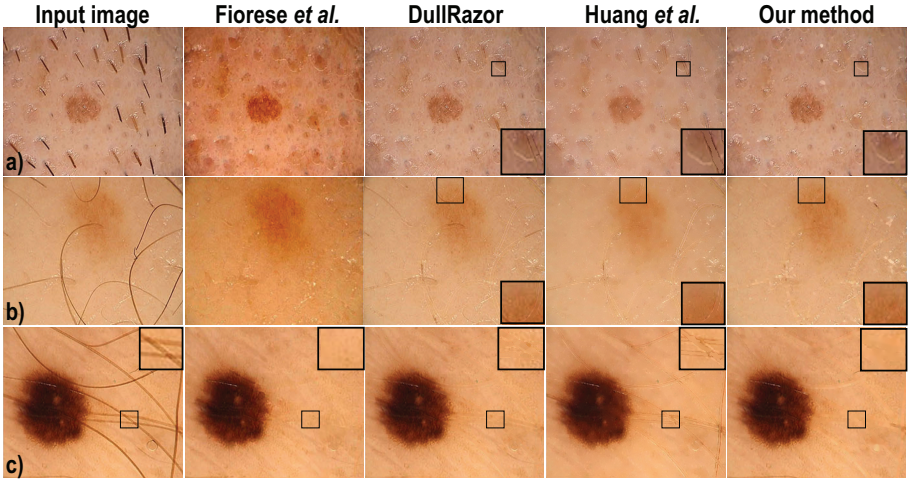


Figure 5.7: Comparison of Fiorese *et al.* [84], Huang *et al.* [104], DullRazor, and our method. Input images from Fiorese *et al.*

5.6 DISCUSSION

Parameters: To obtain full automation, we ran our method on several tens of skin images (at resolution 1024^2), varying all its parameters, and selected those values which visually yielded the best results (most true-positive and least false-positive hairs). Next, we computed final parameters by averaging, and tested that these values give good results on our full image test-set. Table 3 presents the final parameter values, used to produce all images in this chapter.

Table 3: Empirically established parameter values.

	Description	Definition	Value
H	Structuring element radius	Section 5.3.2	5.0 pixels
λ	Gap detection parameter	Equation 5.8	0.2
μ	Skeleton simplification parameter	Equation 5.10	0.05
τ_{min}	Minimum skeleton pruning	Equation 5.10	3.0 pixels
τ_{max}	Maximum skeleton pruning	Equation 5.10	40.0 pixels
δ_{max}	Hair detection parameter	Section 5.3.3	20.0 pixels
δ_{avg}	Hair detection parameter	Section 5.3.3	10.0 pixels

Robustness: We reliably remove hairs regardless of thickness, curvature, length, color, or underlying skin pattern. Very thin and low-contrast hairs may not get (fully) removed, as they are either not found in M^f or do not meet the elongation criteria (Sec. 5.3.3). Yet, such details do not influence further manual analysis tasks (since they are too small to occlude significant skin tumor patterns) or automatic analysis tasks (since they are too small and low-contrast to influence image descriptors such as color histograms, gradients, texture descriptors, or edge detectors).

Speed: We compute an open-close, a close-open, a skeletonization, and a skeleton-to-shape reconstruction step for all threshold layers T_i found in an image. For a 1024^2 pixel image densely populated by hairs, this takes 28 seconds on a MacBook Pro Core i7 with a GT 750M GPU, and 18 seconds on a comparable desktop PC with a GTX 690. For the same image and desktop PC, DullRazor needs 4 seconds, Fiorese *et al.* 7 seconds, Abbas *et al.* 40 seconds, Xie *et al.* 150 seconds, and Huang *et al.* about 10 minutes. As such, our method is the third-fastest from the set of methods we compared against.

The complexity of our method is $O(\|T\| \cdot \|I\|)$, *i.e.*, it is linear in the size of the input image I and the number of threshold layers that we decompose I into (Sec. 5.3.1). This is due to the fact that all core operations in our pipeline (morphological filters, inpainting, distance transforms, and skeletonization) are linear in the number of processed pixels, and we process $\|T\|$ such images, one for each threshold layer. We next analyze how our implementation scales with respect to the number of threshold layers $\|T\|$, as this is the parameter that dominates the processing time. For this, we fix the input image resolution at 1024×768, and run our DHR method on 100 images which have a wide variation of the remaining parameters (type and density of hairs and skin color). Figure 5.8 (left) shows the measured execution timings *vs* the number of different threshold images $\|T\|$ found in each input image. We notice a good linear

correlation of the execution time with number of thresholds. Note that a maximum of 512 threshold images are being processed, as we compute two masks M^l and M^d, and each mask is determined by maximally $2^8 = 256$ thresholds.

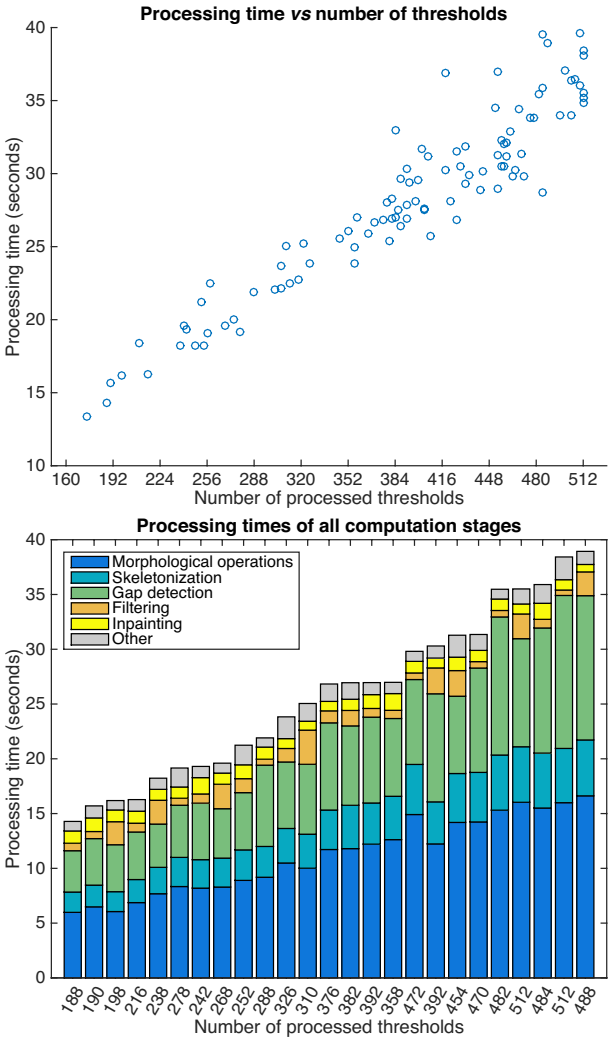


Figure 5.8: Top: Total processing time as a function of $\|T\|$. Bottom: Relative cost of our methods' different computation stages, sorted on total time.

Figure 5.8 (bottom) shows the distribution of relative costs of the various stages of our pipeline, for the same set of images as in Fig. 5.8 (top). Several points can be made here, as follows. First, we note that the *relative* costs of all stages are largely independent on the number of processed thresholds – or, in other words, that the total cost is indeed dominated by the number $\|T\|$ of processed threshold images. Secondly, we note that total cost is dominated by morphological operations (opening and closing) and the gap-detection (computation of image D by disk drawing, see Sec. 5.4).

Interestingly, computing exact Euclidean skeletons, which is often perceived as an expensive operation, accounts for only 10 up to 15% of the total processing time. Inpainting also has a very low cost, which justifies our implementation thereof on the CPU. Overall, this analysis tells that significant speed-ups can be obtained by optimizing our implementation of the morphological operations and disk-drawing used to detect the hair gaps.

Per image threshold, we obtained an average processing time $\bar{\tau} \in [60, 90]$ milliseconds, following a Gaussian distribution $\mathcal{N}(73.8, 31.7)$. Furthermore, we tested for correlation of $\bar{\tau}$ with various image features such as the amount of hair pixels detected in an image, amount of hair crossings, and average hair length. No significant correlations were found. This strengthens the earlier observation that our method's throughput is dominated by number of processed image thresholds (for a given image resolution) and not by the type and/or amount of hair to remove. In turn, this indicates that, if our current morphological operations and gap detection implementations were further optimized, significant performance can be consistently gained. Separately, this tells that our method can be trivially accelerated by using newer GPUs that offer more processing cores.

Tumor segmentation use-case: A practical way to measure the quality (and usefulness) of our DHR method is to see how different the results of *tumor segmentation* are for images with hair and with hairs removed by our method. Tumor segmentation is a crucial step in the computation of image descriptors used for skin lesion classification, since such descriptors need to be assessed only over the lesion area and not over surrounding healthy skin [45, 132, 172, 194].

To assess this difference, we considered several skin images having high-contrast hairs. Such hairs adversely influence most automatic segmentation methods that try to separate the tumor from surrounding skin (see *e.g.* [226], Sec. 9.4.2, Fig. 9.9). We considered next two segmentation methods which are applicable to skin tumors: superpixel graphs based on the image foresting transform [182] and the more specific normalized-cut method in [85], which claims to be robust for skin lesions occluded by hairs. We also tried other known segmentation methods, such as the active contour approach used in skin tumor segmentation in [172], the mean shift method [57], and the level-set approach in [143]. However, these additional methods showed much larger sensitivity to input image characteristics, including hairs but also lesion details, as compared to [182] and [85]. As such, we deemed them less suitable candidates for skin segmentation in general, and eliminated them from further detailed inspection.

Focusing on the two most robust segmentation methods of hair-occluded tumors [85, 182], we see that both methods still have significant problems for images containing long high-contrast hairs. These problems manifest themselves in terms of creating segments which either contain large parts of skin outside the lesion or have boundaries that follow hairs that intersect the lesion (Fig. 5.9 b,c, red markers). Such suboptimal segmentations create *major* problems for *e.g.* the computation of reliable image descriptors that should characterize the precisely delimited tumor area, to be used in automatic lesion classification [45, 194]. After removing hairs by our DHR method (Fig. 5.9 d), both considered segmentation methods achieve a very good segmentation result that closely follows the apparent skin tumor boundary without being distracted by crossing hairs (Fig. 5.9 e,f). This shows that our DHR method can

be used as an automatic preprocessing filter for robust skin-tumor segmentation in tumor classification pipelines.

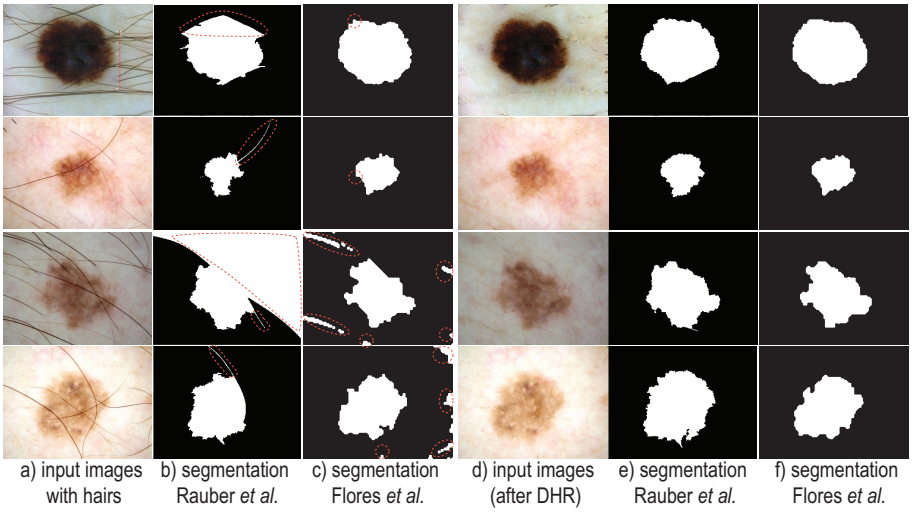


Figure 5.9: Tumor segmentation: (a) Input images with hairs, and corresponding segmentations using the methods of (b) Rauber *et al.* [182] and (c) Flores *et al.* [85]. Segmentation artifacts are marked in red. (d-f) Results of segmenting the same images with the same methods after hair removal.

Qualitative validation: We have shown all our input images, and obtained DHR results, to two dermatologists having over 11 years of clinical experience, in blind mode – that is, the two specialists did not know of each other's assessment results, nor did they know about the aims of the evaluation or specifics of the DHR method being used to remove hairs. We asked whether the raw *vs* DHR-processed images would lead them to different interpretations, diagnoses, or insights. For all images, the answer was negative. While a more formal measurement would bring additional insights, this test already tells that our DHR method does not change the images in *undesirable* ways from the perspective of specialist users who assess them. Separately, hair removal is obviously *desirable*, *e.g.* when using images in automated image analysis and classification procedures [2, 104, 172, 181], such as the tumor segmentation use-case discussed above.

Quantitative validation: To quantitatively assess the effect of hair removal, we performed the following experiment. We created three image databases, each having 10 different images (within each database and across databases), all images having the same resolution. Database 1 contains skin-tumor images without occluding hairs. Database 2 contains images with significant amounts of occluding hairs. Database 3 contains images created by our DHR method by removing hairs from a set of images with occluding hairs (not present in database 2). For each image in the three databases, we next computed the gradient magnitude at each pixel and its standard deviation over each image. Separately, we computed the standard deviation of the

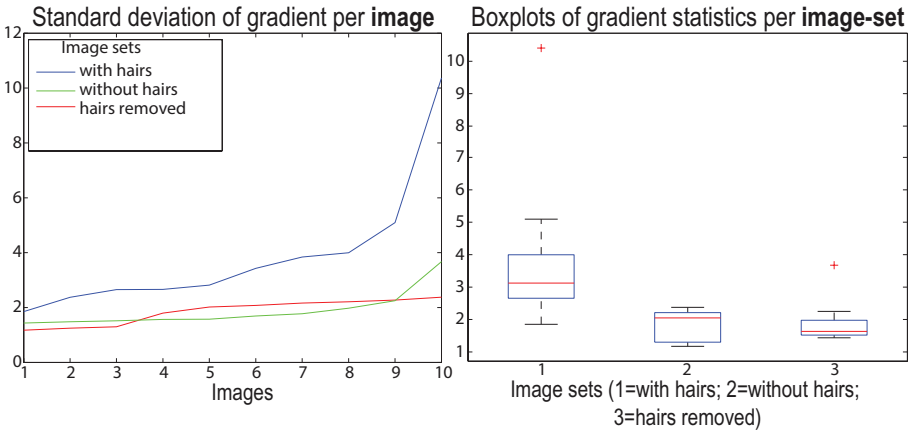


Figure 5.10: Left: Standard deviation of per-image gradients in three considered image-sets. Right: Minimum, maximum, average, and standard deviation, per-image-set, of image gradients.

per-image gradients over all images in each database. These metrics give an overall characterization of image features relevant for a wide range of tasks such as image classification [6, 60, 129], since most image descriptors such as color histograms, tumor boundaries, edge histograms, and texture descriptors used by such techniques strongly depend on local image gradients [132, 194]. Plotting the gradient standard deviation for images in all three databases, we see that the DHR images are very similar to the *different* hair-free images, while the hair-occluded images clearly stand apart (Fig. 5.10). This supports the hypothesis that our DHR method creates, on average, images which have the same statistical characteristics as hair-free images, *i.e.* images which could be used with the same success as hair-free images in various automatic analyses. While more accurate comparisons of the actual image features extracted from the raw *vs* DHR-processed images could be performed, this test already indicates a good statistical match between our results and typical naturally hair-free images.

Limitations: For very dense hairs of varying color on high-contrast skin (*e.g.* Fig. 5.2 h), we cannot fully remove all hairs. Yet, this image type is extremely atypical – it actually is a skin lesion of a Labrador canine subject, which has massively more hairs than typical humans; whose hairs are significantly thicker than human hair, half-transparent, and hollow; and whose underlying skin texture shows complex high-contrast striations. Also, other methods [104, 138] remove significantly less hairs in such cases. Separately, while our method's speed is around the average of the tested competitors, faster (albeit lower-quality) methods exist [84, 138]. Using a more conservative method to select a subset of layers from the entire threshold set of 256 binary images to further process to detect hairs, in line with similar layer-selection procedures used for image compression [246], would accelerate our method up to one order of magnitude. Indeed, as discussed earlier in this section, our speed is chiefly influenced by the number of processed layers. This would make our approach (com-

pete with) the fastest DHR method published so far.

5.7 CONCLUSIONS

We have proposed a new approach for digital hair removal (DHR) by detecting gaps in all layers of an image threshold-set decomposition. We find false-positives by using medial descriptors to find thin and elongated shapes. We compared our method against five known DHR methods on a set of over 300 skin images. To our knowledge, our work is the broadest DHR method comparison published so far. By observing the results obtained from this comparison, we notice that our method can better remove long curly hair and short stubble hair than its competitors. Separately, we showed how our method effectively improves skin tumor segmentation in the case of hair-occluded tumors, an important asset for automatic skin lesion processing. Performance analysis of our method show its linear dependence on the input image size and number of threshold sets identified in the image. Qualitative and quantitative validations support the claim that our method produces images which are perceptually and also quantitatively very similar to the original hair-occluded images.

Future work can target several directions. Machine learning techniques [6, 60, 129] could be used to improve false-positive filtering. Further false-negative avoidance can be improved by extending our method to use additional input dimensions besides luminance, such as hue and texture. Application-wise, our method can be straightforwardly incorporated into skin tumor classifiers for melanoma detection in order to make such techniques directly applicable to hair-occluded images too.

This chapter is based on:

J. Koehoorn, A. Sobiecki, D. Boda, A. Diaconeasa, S. Doshi, S. Paisey, A. Jalba, and A. Telea. Automated Digital Hair Removal by Threshold Decomposition and Morphological Analysis. In *Proc. International Symposium on Mathematical Morphology (ISMM)*, Springer, 2015

J. Koehoorn, A. Sobiecki, P. Rauber, A. Jalba, and A. Telea. Efficient and Effective Automated Digital Hair Removal from Dermoscopy Images. *Mathematical Morphology - Theory and Applications*, submitted, 2015.

6 QUALITATIVE COMPARISON OF CONTRACTION-BASED CURVE SKELETONIZATION METHODS

Chapters 4 and 5 have shown that 2D medial axes are efficient and effective tools that can support the design of a variety of 2D shape restoration methods, including segmentation, crack detection and removal, and digital hair removal. As such, it is natural to ask oneself whether such results can be extrapolated to the usage of 3D skeletons for the restoration of 3D shapes. In contrast to the 2D case, where the selection of a good skeletonization method was relatively easy, many different techniques exist that compute various types of skeletons for 3D shapes. Such techniques differ widely in terms of their compliance with desirable skeleton properties. To get more insight into this compliance, we perform a qualitative study focused on curve skeletons computed with mesh-based methods. Chapter 7 extends this work to the study of voxel-based curve and surface skeletons.

6.1 INTRODUCTION

Curve skeletons are among the most well-known, and widest used, descriptors for 3D shapes. They have been extensively used in applications such as shape matching and recognition, computer animation, virtual navigation, and shape processing [59, 203]. Earlier methods for computing curve skeletons used mainly voxel-based 3D shapes. In recent years, several methods have been proposed to compute curve skeletons from meshed 3D shapes, using a *contraction* principle, where the input mesh is iteratively shrunk towards its local center. Such methods are highly computationally scalable, and can easily handle mesh shapes with considerable more details than voxel-based methods. As such, they appear, at first sight, to be strong contenders for optimal tools to be used in skeleton-based applications. However, their algorithmic complexity makes it harder to reason analytically about the properties of the produced skeletons. In particular, it is not fully clear how their results relate to desirable skeleton properties.

To bring more insight herein, we compare six mesh-contraction-based curve skeletonization methods against six accepted quality criteria: centeredness, homotopy to the input shape, detail preservation, smoothness, and independence from the input shape's sampling. Our work extends the earlier survey of Cornea *et al.* [59] by adding to the comparison six new mesh-based curve skeletonization algorithms published after that survey was done. Our results reveal several limitations of the studied methods which, to our knowledge, have not been highlighted in the literature, and link these to algorithmic aspects of the studied methods.

The structure of this chapter is as follows. Section 6.2 overviews related work in curve skeletonization, with a focus on contraction-based methods. Section 6.3 details the quality criteria used for the comparison. Section 6.4 presents the comparison results. Section 6.5 discusses our findings. Section 6.6 concludes the chapter with future work directions.

6.2 RELATED WORK

For a shape $\Omega \subset \mathbb{R}^3$ with boundary $\partial\Omega$, we first define its distance transform $DT_{\partial\Omega} : \mathbb{R}^3 \to \mathbb{R}^+$

$$DT_{\partial\Omega}(\boldsymbol{x} \in \Omega) = \min_{\boldsymbol{y} \in \partial\Omega} \|\boldsymbol{x} - \boldsymbol{y}\|. \tag{6.1}$$

The surface skeleton of Ω is next defined as

$$S(\Omega) = \{\boldsymbol{x} \in \Omega \,|\, \exists\, \boldsymbol{f}_1, \boldsymbol{f}_2 \in \partial\Omega, \boldsymbol{f}_1 \neq \boldsymbol{f}_2, \|\boldsymbol{x} - \boldsymbol{f}_1\| = \|\boldsymbol{x} - \boldsymbol{f}_2\| = DT_{\partial\Omega}(\boldsymbol{x})\} \tag{6.2}$$

where $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are the contact points with $\partial\Omega$ of the maximally-inscribed ball in Ω centered at $\boldsymbol{x}$ [93, 187], also called *feature transform* (FT) points [109]. Surface skeletons consist of several manifolds with boundaries which meet along a set of Y-intersection curves [48, 66, 141]. They can be computed by voxel-based or mesh-based methods [18, 36, 102, 171, 217]. A recent comparison of surface-skeleton extraction methods is given in [109].

In contrast to surface skeletons, curve skeletons are loosely defined as 1D structures "locally centered" within the input shape Ω. The lack of a unanimously accepted formal definition has led to many methods which compute curve skeletons following not necessarily identical definitions. This makes it hard to analytically compare, and reason about, the properties of the produced curve skeletons.

Tools from mathematical morphology [197] were among the first used to compute curve skeletons: The residue of openings, based on Lantuéjoul's formula [136], usually leads to disconnected skeleton branches, whereas methods based on homotopic thinning transformations [29, 136, 158, 171] yield connected skeletons.

Dey and Sun propose one of the first analytic definitions of curve skeletons based on the medial geodesic function (MGF), where the curve skeleton is defined as the locus of points having at least two equal-length shortest geodesics on $\partial\Omega$ between their feature points [70, 178]. Reniers *et al.* extend the MGF to regularize curve skeletons by assigning each skeleton point an importance equal to the area bounded by such geodesics [187], inspired by the so-called 2D collapse metric [168, 228]. A GPU implementation of the above metric is presented in [109]. The most recent work in the state of the art [11] present a distance-driven method to compute the surface and curve skeletons of 3D objects in voxel images.

Voxel-based methods typically require significant resources to store and process the large voxel volumes required to capture the fine details of complex 3D shapes. To be used on 3D meshes, such methods require a costly voxelization step. Mesh-based methods address these cost issues by working directly on a mesh representation of $\partial\Omega$. In recent years, several such methods have been proposed based on a *contraction* principle, which shrinks the input mesh until the 1D curve-skeleton structure is reached, as follows. Au *et al.* shrink the mesh via Laplacian smoothing until its volume gets close to zero, followed by an edge-collapse (to extract the 1D curve skeleton) and a re-centering step (to correct shrinking errors) [15]. Cao *et al.* extend this idea to extract curve skeletons from incomplete point clouds [41]. The ROSA method defines, and extracts, curve skeletons using rotational, rather than positional, symmetry: $\partial\Omega$ is cut with planes, and curve-skeleton points are found as the centers of planes which

minimize the variance between the plane's normal and $\partial\Omega$ normals along the cut curve [220]. Sharf *et al.* reverse the contraction direction: They find the curve skeleton as the centers of a set of competing fronts which evolve to approximate the input surface [201]. A similar method is presented by Hassouna and Farag [98]. Telea and Jalba define, and extract, curve-skeletons by contracting the surface skeleton $S(\Omega)$ (computed as in [151]) inwards, along the gradient of the 2D distance transform of $\partial S(\Omega)$, *i.e.* define the curve-skeleton as the result of a two-step skeletonization [227].

Mesh-contraction methods are currently deemed to be the state-of-the-art for extracting detailed curve skeletons from high-resolution shapes [221]. As 3D models become more complex, it is arguable that such methods will dominate the more costly voxel-based methods. Conceptually, such methods work very similarly to voxel-shape thinning. However, there are few, if any, comparisons of recent contraction-based methods. Also, the algorithmic complexity of mesh-contraction methods makes a formal analysis thereof more complex than for voxel-based methods. All in all, it is not clear if mesh-contraction methods are indeed always superior to voxel-based methods, and if not, which are their specific weak points with respect to desirable skeleton criteria.

6.3 COMPARISON CRITERIA

The literature knows a well-accepted set of quality criteria that curve skeletons should conform to. For curve skeletonization methods, such criteria are significantly more important than for surface skeletonization methods: While the latter can be rigorously checked against the formal surface skeleton definition (Eqn. 6.2), the former do not use a single curve-skeleton definition. As such, the only comparison available for curve skeletons is a qualitative one, from the perspective of desirable quality criteria. Following [59, 109, 203], we focus on the following generally-accepted quality criteria for a curve skeleton:

Homotopy: The curve skeleton is topologically-equivalent with the input shape, *i.e.* has the same number of connected components and tunnels.

Invariant: The curve skeleton should be invariant under isometric transformations of the input shape. We do not explicitly test against this, because our compared MBS methods are invariant by contraction.

Thin: The curve skeleton should be as thin as the sampling model used allows it. Voxel-based curve skeletons should be one voxel thick. Mesh-based curve skeletons should contain only lines, and not polygons or loose points. Point-cloud based curve skeletons should ideally have zero local thickness in any direction orthogonal to the largest eigenvector of the covariance matrix of point neighborhoods.

Centered: This is the hardest criterion to quantify, since it is not uniquely defined when a curve is centered within a 3D shape. However, several weak forms of curve-skeleton centeredness exist: The curve skeleton should be a subset of the surface skeleton (since the latter is by definition centered within the shape); and in no case

should the curve-skeleton exit the input shape.

Smoothness: As centeredness, smoothness is also hard to formally define. Surface skeleton manifolds are known to be at least $\mathscr{C}^2$ continuous [176, 203]. Curve-skeletons are centered subsets thereof [221, 227]. Hence, it is arguable that curve skeletons should be also piecewise, *i.e.* per branch, $\mathscr{C}^2$. In any case, curve skeletons should not exhibit curvature discontinuities induced by the sampling of either the input surface or curve skeleton representation.

Detail preserving: Curve skeletons should be able to capture fine-scale details, such as bumps, of the input shape, in a user-controlled manner. In other words, the user should be able to select the scale of input shape details which the curve skeleton should capture (being significant) and the scale of details to ignore (being regarded as noise).

Sampling robustness: Given two different samplings of an input shape (*e.g.* two different level-of-detail meshes), the difference between the two corresponding curve skeletons should be proportional with the difference of the two input meshes. In other words, small input-sampling differences should not cause large differences in the curve skeleton.

Reconstruction: Given a skeleton and the distances from its points to the input shape, *i.e.*, the so-called medial axis transform (MAT) of that shape, it is possible to reconstruct (approximations of) the shape from the MAT. In theory, given an exact MAT, the input shape can be perfectly reconstructed. This property does not, however, hold in practice, due to approximations in the skeleton computation, resolution limitations, and skeleton simplification. Also, it is well known that reconstruction requires the usage of a *surface* skeleton – the *curve* skeleton does not capture enough information to reconstruct anything beyond locally tubular shapes. Given the above, and our focus in this chapter on curve skeletonization methods, we do not discuss the reconstruction property further.

Different skeletons have the same number of branches as far as possible, we tried it, as much as possible. The fact that the results arre not very similar is simply a limitation of these methods.

6.4 COMPARISON

Given our core question on how mesh-contraction-based curve skeletonization methods perform, we compared six such methods (further denoted in the paper by the abbreviations listed below):

Au *et al.* (AU) [15]: We included this method as it is arguably the best-known mesh-based skeletonization technique in existence [98, 109, 221].

Tagliasacchi *et al.* (ROSA) [220]: We chose this method given its advocated noise-resistance and since it works on point clouds, which is a different type of input than

the other methods.

Cao *et al.* (CAO) [41]: We chose this method since it uses a contraction similar to [15], but works on point clouds, like [220].

Telea and Jalba (TJ) [227]: In contrast to all other curve skeletonization methods, this technique contracts the surface skeleton, rather than the input mesh, to compute the curve skeleton. It produces a point cloud rather than a polyline curve-skeleton. For comparison fairness, we postprocessed the produced point cloud using the polyline reconstruction proposed in [15].

We also developed and tested two extensions of [15], as follows.

Au *et al.* improved (AUI): A well-known limitation of Au *et al.* is its skeleton re-centering step [221]. As the input mesh is contracted, it can go off-center due to numerical and discretization inaccuracies of the Laplacian smoothing. To address this issue, we proceed as follows. During the Laplacian contraction and edge-collapse steps of the method, we maintain a backwards, skeleton-node-to-mesh-vertex mapping $\Pi : S \rightarrow \partial\Omega$, which can be used to identify those mesh vertices $\boldsymbol{v} \in \partial\Omega$ that 'collapsed' into a given skeleton node $\boldsymbol{s} \in S(\Omega)$. The re-centering step uses Π to compute the final position of each node $\boldsymbol{s}$ as a weighted average of the vertices in $\Pi(\boldsymbol{s})$, with weights given by the areas of the input-mesh triangles with vertices in $\Pi(\boldsymbol{s})$.

Au *et al.* using surface skeletons (AUS): The improved re-centering outlined above cannot fully correct errors accumulated during the iterative contraction. To further reduce these, we start the Laplacian contraction from the surface skeleton, which is closer to the final target (curve skeleton) than the input mesh, along the idea proposed in [227].

Global considerations: In our method choice, we focused on recent contraction-based techniques, not studied in the survey of Cornea *et al.* [59], proven by their authors on complex shapes, and which use different curve-skeleton detection principles. All methods, also directly satisfy the thinness criterion, since they model the curve-skeleton as a polyline. We used the original implementations provided by their authors, all running on a Windows PC with 4 GB RAM. Since not all studied methods claim computational efficiency, we excluded timings from the comparison.

Comparison material: For comparison, we used a set of 21 3D shapes which are frequently encountered in the curve-skeleton literature (for details, see [242]). Figures 6.1, 6.2, 6.3 and 6.4 and show relevant samples from this set, within space limitations. The models have between 20K and 300K vertices. We used MeshLab [234] to clean mesh models for normal orientation consistency, T-vertices, and duplicate vertices. To factor our parameter settings, we ran each method for uniformly-sampled values of all its documented parameters, and retained in our final comparisons the best results with respect to the quality criteria mentioned in Sec. 6.3.

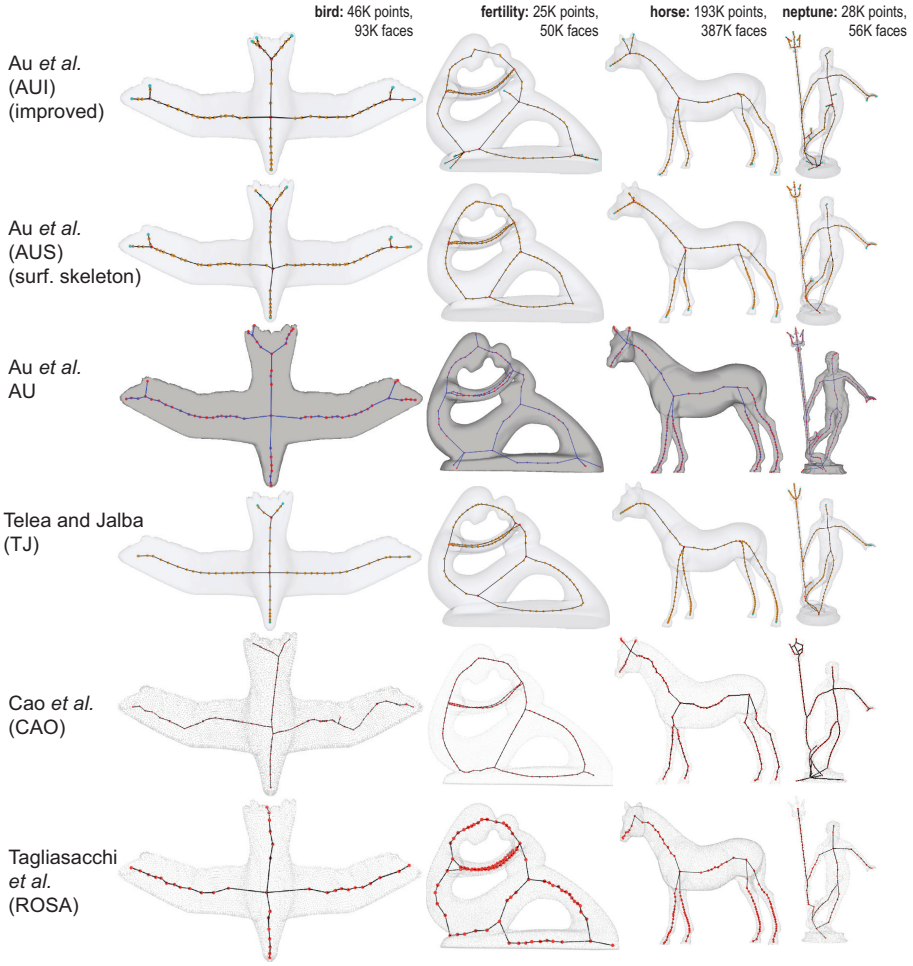


Figure 6.1: Overview comparison of skeletonization methods.

6.4.1 *Overview*

Figure 6.1 shows an overview of several curve skeletons extracted by the compared methods. Even at this level, we quickly notice that not all skeletons are equally well centered, equally smooth, and have the same number of terminal (detail) branches. We next zoom-in on each criterion and discuss our findings with respect to the studied methods.

6.4.2 *Homotopy*

For relatively simple shapes of genus 0 or higher, all studied methods behaved equally well, *i.e.* produced curve skeletons homotopic with the input shape (Fig. 6.1). Still, detail differences exist. Skeleton junctions are not always identical, so the produced

skeleton graph is different, see *e.g.* the marked limbs-to-body junctions of the *bird* model in Fig. 6.2 (left) and the *horse* model in Fig. 6.3 (right). Differences get larger for small-scale details, where curve skeleton terminal branches enter saliencies of the input shape, see *e.g.* Fig. 6.3 (*neptune*, *frog*). An extreme case happens when the input mesh has self-intersections, *e.g.* Fig. 6.2 (*frog*). Here, CAO and ROSA create curve skeletons whose topology is far from the input shape (fake loops and branches).

6.4.3 *Centeredness*

The methods AU, AUI, and AUS produce similar, well centered, results. Among these, AUS is the best: Since contraction starts from the surface skeleton, nodes go less off-center, as the surface skeleton is already centered by definition and closer to the curve skeleton than the input mesh. For mesh-based methods, TJ produced the best centering. This is due to the fact that TJ contracts the surface skeleton along the gradient field of its 2D distance transform, which is by definition tangent to the surface skeleton itself, so the curve skeleton stays inside the surface skeleton by construction. In contrast, AU, AUI, and AUS contract in the direction of the shrunken surface's normals. These are delicate to estimate as the shape shrinks and develops singularities (creases). The different re-centering steps performed by these methods alleviate, but cannot fully correct, these problems.

ROSA's results are quite poorly centered in several areas. As mentioned in [220], orientation information is unreliable around junctions, where the input shape has many points with diverse orientations. To overcome this, ROSA treats junctions specially. This works well for junctions whose branches correspond to tubular shape parts of similar size. However, we discovered that junctions where shape parts of very different sizes and shapes meet create problems, see *e.g.* Fig. 6.2 for the *bird* model (wings joining rump) and *neptune* (arm-torso junction).

The *frog* model (Fig. 6.2) reveals two other challenges. First, the model has several very sharp bends around the leg joint. Secondly, in the same area, the mesh has several self-intersections. Meshless methods (CAO, ROSA) generate seriously erroneous skeletons here, and even skeleton disconnections. In these areas, TJ still creates a smooth skeleton, but cannot handle centeredness perfectly. This is due to the fact that the surface skeleton it starts from has errors in self-intersecting areas, since the technique used to compute it [109] cannot handle self-intersecting surfaces. In contrast, AU, AUI, and AUS generate very similar, relatively well-centered, skeletons in these challenging areas.

The *neptune* model (Fig. 6.2) highlights the situation where a relatively thin object part (arm) joins a thick one (torso). In such areas, curve (and surface) skeletons exhibit so-called ligature branches which connect the skeleton branches of the two parts [176]. If the two parts form an angle different from 90°, like in our case, the ligature branch has to rapidly turn [203]. This turn is best captured by AU. In contrast, all other methods emphasize smoothness too much, which results in clearly off-centered skeletons close to the armpit.

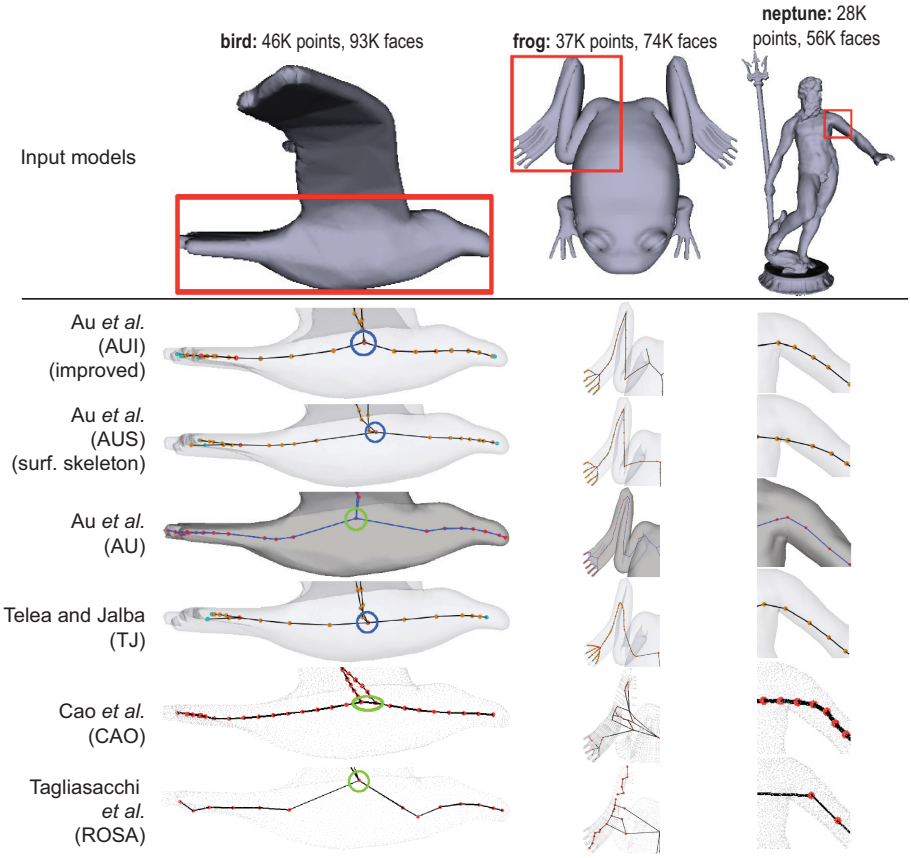


Figure 6.2: Centeredness comparison.

6.4.4 *Detail preservation*

Detail preservation refers to the generation of separate curve-skeleton terminal branches for all input shape bumps, or salient convexities, at a user-specified scale. Detail preservation is important for applications such as shape matching, retrieval, and reconstruction [59, 187]. Large details, such as the limbs of shapes in Fig. 6.1, are well captured by skeleton branches by all studied methods. For smaller-scale details, the situation is different, see Fig. 6.3 left. The problem is that all methods include explicit actions to smooth the computed skeletons. Although desirable (see next Sec. 6.4.5), such smoothing will remove some small-scale branches.

AU and AUI preserve small-scale, detail, branches best. In contrast, AUS and TJ find detail branches of long protrusions (*e.g.* Fig. 6.3, *neptune* and *frog* fingers) quite well, but fail to find branches for shallower bumps, such as *gargoyle*'s wing-tips. Upon closer analysis, we found that this is caused by the fact that the surface skeletons that both AUS and TJ start from, fail to capture such details, hence these details cannot appear further in the curve skeleton. CAO and ROSA perform the worst for this criterium. These methods fail finding most detail skeleton branches found by the other

studied methods. Moreover, when found, small-scale terminal skeleton branches seem to be arbitrary, as Fig. 6.3 shows for all three models on the left.

Small-scale noise is ignored equally well by all methods. For mesh-based methods, this is an effect to their built-in smoothing.

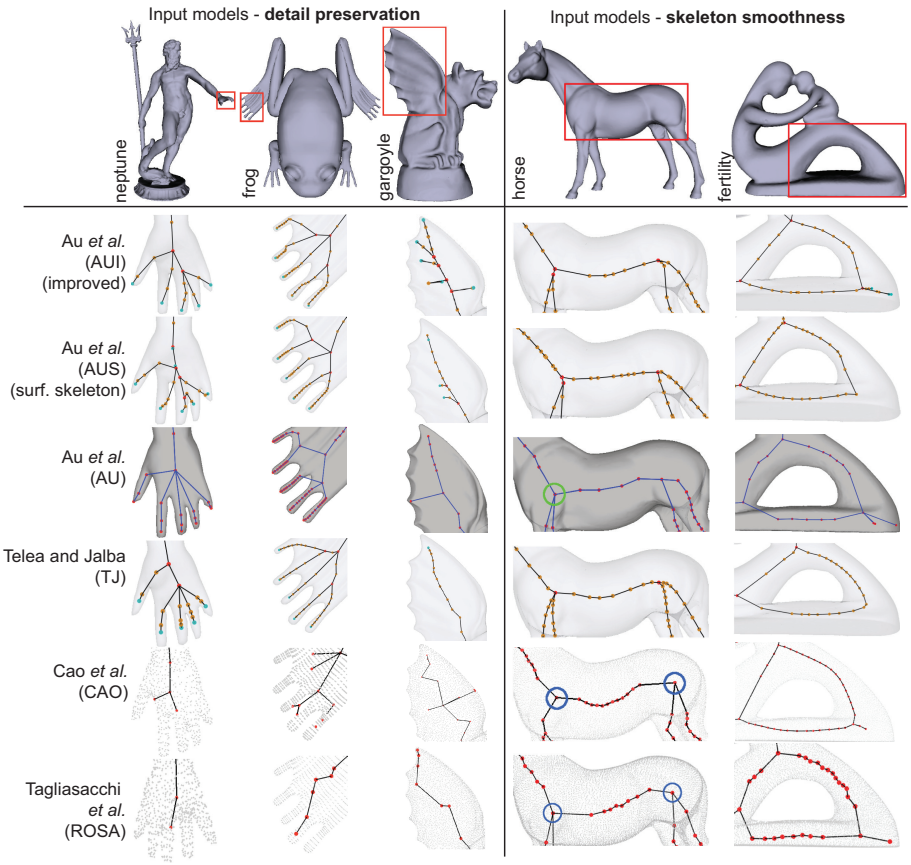


Figure 6.3: Comparison for detail preservation comparison (left) and skeleton smoothness (right).

6.4.5 *Smoothness*

As outlined earlier, curve-skeleton branches should be at least $\mathscr{C}^2$ continuous curves (Sec. 6.3). Hence, skeletonization methods should follow this property as well as possible. Voxel methods are inherently constrained here by the sampling resolution. In contrast, mesh-based methods which model the curve skeleton as a polyline should distribute the computed skeletal points, or sample the skeleton, to optimally approximate the desired smooth curve. Hence, for these methods, the issue of skeleton smoothness is implicitly connected to the skeletal curve sampling.

Contraction-based methods, as the ones we studied, have an additional challenge here. As the input mesh is contracted, the local point density naturally increases in convex areas and decreases in concave ones. This potentially leaves too few nodes to approximate well the curve skeleton in concave areas. Ligature branches are an extreme case hereof. An example are the ligature branches that connect the horse's leg-skeletons to its rump-skeleton (Fig. 6.3 right). Here, CAO, ROSA, and up to some extent AU, clearly show a lower point density – see branches meeting at the marked junctions. This in turn creates spurious kinks in the rump's curve skeleton. In contrast, AUS, AUI, and TJ create smoother skeletons. The skeletons of TJ and AUS follow the rump's curvature best. This is explained by the fact that their contraction is constrained to stay on the surface skeleton, whose shape already captures the input shape's curvature. AU and AUI both fail capturing the rump's curvature, since they have no such constraint. The same non-uniform skeletal point distribution is also observed for the *fertility* model (Fig. 6.3 right). Here, again, AUS and AUI yield the most uniform point distribution, and ROSA and AU the least uniform one (which leads to unnatural kinks).

6.4.6 *Sampling robustness*

Sampling robustness refers to the relation between the resolution of the input shape and changes in its curve skeleton. Ideally, we would like that when the former changes slightly, the curve skeleton also changes only slightly. This property is closely related to the concept of *regularization*, which states that small changes in the input shape Ω should only yield small changes in its skeleton [109, 187, 228].

To study this, we produced three versions of the *dragon* model (see Fig. 6.4), using the Yams mesh resampling tool [88]. We also produced three voxel models of the same shape from the highest-resolution mesh, using *binvox* with three sampling resolutions. Next, we ran the studied skeletonization methods on these datasets, and analyzed the results. In the comparison, we had to exclude CAO and ROSA, as the provided implementations of these methods were too slow to complete, even in several hours, for the largest-resolution meshes.

The method AU is quite sensitive to the mesh sampling. Looking at Fig. 6.4, we see that, in the dragon head area, the small and large resolution models produce relatively similar skeletons, but the medium-resolution model yields a very different skeleton topology. Given that higher resolution can only potentially add extra details, but not remove existing ones, we expect to get an increasingly rich curve skeleton (in terms of terminal branches), but the core structure of this skeleton should not change significantly. This is not the case, which hints to an important instability of the method with respect to mesh resolution.

In contrast, AUS and AUI show a much stabler curve skeleton with respect to mesh resolution. Although these methods do not produce identical skeletons for the same resolution, the changes of their respective skeletons as the resolution changes, are quite small. Both methods find more terminal skeleton branches as the resolution increases, which is expected since higher-resolution models capture more surface details.

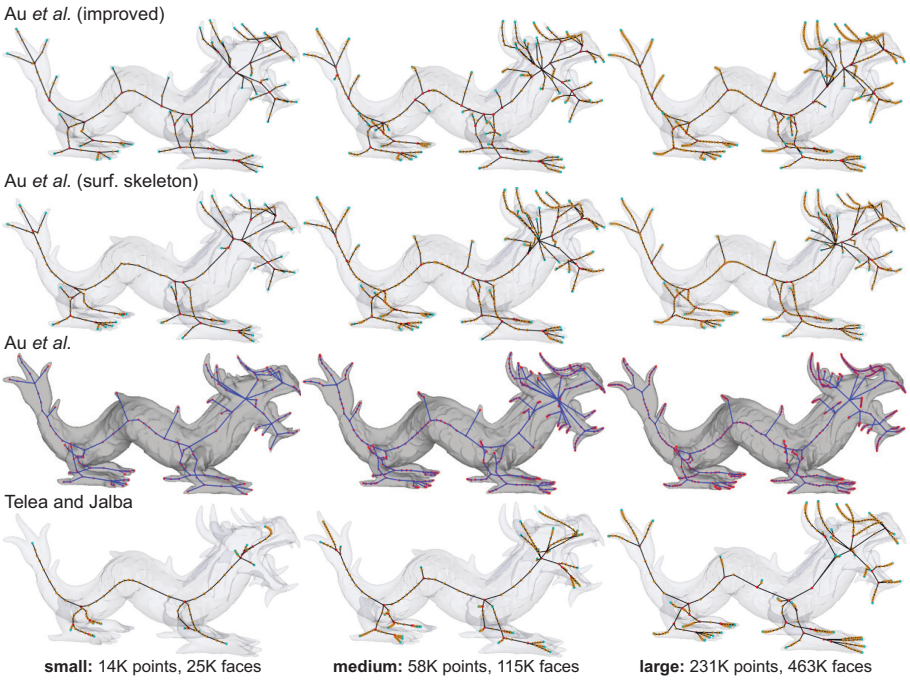


Figure 6.4: Sampling robustness comparison.

The TJ method is the most sensitive to sampling. For the low-resolution model, the method simply fails to extract many significant branches. Although more branches are found for the high resolution model, many significant surface details, like the upper spikes on the back and tail, fail to generate branches. This can be directly traced to the quality of the surface skeleton: The underlying method used to compute it [151] produces as many skeleton points as surface points. To accurately capture the surface skeletal structure, very densely-sampled models are required [109]. Less densely sampled surface skeletons will in turn create a noisy distance-transform gradient, which will contract the skeleton mesh in the wrong directions.

6.5 DISCUSSION

Contrary to our initial belief, based on the contraction-based skeletonization literature, all contraction methods appeared to be much more sensitive in terms of *all* studied quality criteria than implied by the examples in the literature. The CAO and ROSA methods performed significantly under expectations. The AU method performed relatively well for smooth shapes, but showed limitations for centeredness and smoothness for more complex shapes. This is the main reason for us having designed the two improved variants AUI and AUS. The trade-off between these variants is as follows: While AUS yields smoother skeletons, AUI delivers a better centeredness. The TJ method dominates all others in terms of smoothness, but has clear centeredness

problems in ligature areas, and requires a very high input mesh sampling to generate even moderately-detail skeleton branches.

One aspect which needs further study is to compare MBS with VBS methods. This comparison is far from trivial, since both the surface and the curve-skeleton are sampled in a different way in VBS as compared to MBS, that is, the MBS uses piecewise linear sampling, while the UBS use piecewise constant sampling. For this reason, we leave such a comparison for future work.

However, the main challenge for REN, and similar voxel-based methods, is *scalability*: Voxelizing complex meshes to resolutions over 1000^3 voxels, and further processing such volumes to extract curve skeletons, is much slower, and more memory demanding, than using mesh-based methods. For instance, our highly optimized parallel implementation of REN processes the 700^3 *dragon* model (Fig. 6.4) in around 15 minutes; the equivalent mesh model (463K faces) is processed in under a minute by mesh-based methods. Moreover, the memory consumption of REN is an order of magnitude larger than for mesh-based methods. If efficient data representation and GPU parallelization schemes were designed to reduce this overhead, voxel-based methods may in the end be a very strong competitor to mesh-based methods.

Table 4 summarizes our comparison of the six studied skeletonization methods from the perspective of the six desirable properties discussed in this chapter. The summary is based on an ordinal scale (worst,bad,satisfactory,good,best). While this table cannot present all the (subtle) comparison insights discussed at length in the text, it provides an easy-to-use overview of the comparison results. In detail, the five-point quality scale we use implies the following characteristics:

- **Worst:** The skeleton does not visually match the input shape;
- **Bad:** The skeleton fails with respect to several of the properties discussed earlier in this chapter, but does visually match the input shape;
- **Satisfactory:** The skeleton visually complies with the desired properties for about 50% of its extent;
- **Good:** The skeleton visually complies with the desired properties for about 75% of its extent;
- **Best:** The skeleton visually complies with the desired properties for all its points.

6.6 CONCLUSIONS

In this chapter, we have presented a qualitative comparison of six contraction-based curve skeletonization methods and one boundary-collapse voxel-based method. The methods were compared from the perspective of several accepted quality criteria: homotopy, thinness, centeredness, detail preservation, smoothness, and robustness to sampling. In contrast to recent insights from the mesh skeletonization literature, the studied mesh-based methods appeared to perform less optimal than expected. Also, the studied voxel-based method appeared to outclass all mesh-based methods, with relatively few limitations, apart from computational scalability.

Table 4: Qualitative evaluation of skeletonization methods.

Methods against Criteria						
Methods	Homotopy	Thin	Centered	Smoothness	Detail Preserving	Sampling Robustness
AUI	Best	Best	Satisfactory	Good	Good	Good
AUS	Best	Best	Satisfactory	Satisfactory	Good	Satisfactory
AU	Best	Best	Satisfactory	Bad	Good	Best
TJ	Best	Best	Best	Good	Best	Satisfactory
CAO	Best	Best	Bad	Bad	Worst	-
ROSA	Best	Best	Bad	Bad	Worst	-

Although our comparison is far from exhaustive, it raises a number of important points about the current state of mesh-based curve skeletonization techniques. To put these insights into perspective, and thereby obtain a more complete picture of the state-of-the-art of current skeletonization methods, we extend our study in the next chapter to the analysis of voxel-based skeletonization methods.

This chapter is based on:

A. Sobiecki, H. Yasan, A. Jalba, and A. Telea Qualitative Comparison of Contraction-based Curve Skeletonization Methods. *In Mathematical Morphology and Its Applications to Image and Signal Processing,* 2013.

7

QUALITATIVE AND QUANTITATIVE COMPARISON OF VOXEL-BASED CURVE AND SURFACE SKELETONS

Driven by our interest to use 3D skeletons for shape restoration, Chapter 6 has presented a qualitative comparison of several recent mesh-based curve skeletonization methods. The overall result of this study has pointed out to important limitations of these methods, which make them less appropriate candidates for our shape restoration use-case. This chapter studies the remaining part of the skeletonization-methods space, namely voxel-based curve and surface skeletons. Based on the combined insight of Chapters 6 and 7, conclusions are next drawn as to the suitability of existing skeletonization methods as building-blocks for our shape restoration use-case.

7.1 INTRODUCTION

Skeletons are shape descriptors with many applications in shape processing, registration, retrieval, matching, animation, and compression [203]. 3D shapes admit two types of skeletons: *Surface* skeletons are 2D manifolds formed by the loci of maximally-inscribed balls within a shape [176, 203]. *Curve* skeletons are 1D curves which are locally centered in the shape and capture the shape's part-whole structure [59].

Since the early skeleton definition by Blum [34], many methods have been proposed to compute the two skeleton types. Such methods differ in theoretical aspects, *e.g.* the exact definition for curve skeletons, and practical aspects, *e.g.* space discretization (voxels *vs* meshes); the various approximations being used; and the actual skeleton extraction algorithm. These aspects, and the inherent sensitivity of skeletons to boundary noise, makes different methods produce widely different skeletons for the same input. This causes challenges for the users of skeletons in both research and practical contexts.

Recognizing these challenges, Cornea *et al.* [59] have presented a taxonomy of curve skeletonization methods and the way these satisfy a set of desirable skeletal properties, and illustrated these for four such methods. Since this publication, several new skeletonization methods have been proposed. A recent study, presented in Chapter 6 extended the work in [59] by comparing methods of a particular class (contraction-based methods for meshed shapes) against Cornea's criteria [207]. These two studies however cover only a very limited fraction of the current skeletonization methods. Also, papers introducing new skeletonization methods typically present only few additional comparisons. Table 5 illustrates this for a selection of methods, which is by no means exhaustive. Finally, very few comparisons of surface skeletons with curve skeletons are presented, so there are still many open questions on the relationships of the two skeleton types. As computational advances allow implementing increasingly complex skeletonization methods, the challenge of understanding the relative pro's and con's of such new methods only grows.

In this chapter, we address the above challenges by presenting a comparison of 4 surface and 6 curve skeletonization methods. In contrast to Chapter 6, we focus here on voxel-based methods. In addition to [59], we cover here methods having emerged after their study was published. We use in our comparison the same desirable criteria as in [59]. In addition, we also propose a detailed comparison that aims to provide a fine-grained detail view on the subtle differences between skeletons computed by different methods, including comparisons of curve with surface skeletons. Our results offer additional insight in limitations and challenges of current methods which, to our knowledge, have not been highlighted so far. These results represent further support for the quest of designing better skeletonization methods.

This chapter is organized as follows. Section 7.2 reviews related work. Section 7.3 presents the compared methods and comparison criteria. Section 7.4 presents the comparison methodology. Section 7.5 presents our comparison results. Section 7.6 discusses these results. Section 7.7 concludes the paper.

7.2 RELATED WORK

7.2.1 *Skeletonization methods*

For a shape $\Omega \subset \mathbb{R}^3$ with boundary $\partial\Omega$, we first define its distance transform $DT_{\partial\Omega} : \mathbb{R}^3 \rightarrow \mathbb{R}^+$

$$DT_{\partial\Omega}(\boldsymbol{x} \in \Omega) = \min_{\boldsymbol{y} \in \partial\Omega} \|\boldsymbol{x} - \boldsymbol{y}\|. \tag{7.1}$$

The surface skeleton, also called the medial surface, S_Ω of Ω is next defined as

$$S_\Omega = \{\boldsymbol{x} \in \Omega \,|\, \exists\, \boldsymbol{f}_1, \boldsymbol{f}_2 \in \partial\Omega, \boldsymbol{f}_1 \neq \boldsymbol{f}_2, \|\boldsymbol{x} - \boldsymbol{f}_1\| = \|\boldsymbol{x} - \boldsymbol{f}_2\| = DT_{\partial\Omega}(\boldsymbol{x})\} \tag{7.2}$$

where $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are the contact points with $\partial\Omega$ of the maximally-inscribed ball in Ω centered at $\boldsymbol{x}$ [93, 187], also called *feature transform* (FT) points [109]. When $\Omega \subset \mathbb{R}^2$, Eqn. 7.2 yields the 2D skeleton, also called the medial axis, of the shape Ω. Surface skeletons contain several manifolds with boundaries which meet along a set of Y-intersection curves [48, 66, 141]. Curve skeletons are loosely defined as 1D structures locally centered within a shape $\Omega \subset \mathbb{R}^3$.

Surface and curve skeletons can be computed by geometric, distance field, general field, and thinning methods. **Geometric** methods include Voronoi diagrams [72] and subsets thereof [8], mesh contraction in normal direction [15, 41, 146, 220], mean-shift-like clustering [105], and union-of-balls approaches [109, 151, 159]. Such methods use meshed shape representations and thus scale well to handle high-resolution models [109, 151]. **Distance-field** methods find S_Ω along singularities of $DT_{\partial\Omega}$ [91, 102, 125, 142, 204, 236] and can be efficiently done on GPUs [41, 217]. **General-field** methods use fields smoother (with fewer singularities) than distance transforms [5, 13, 58, 98]. Such methods are more robust for noisy shapes. Various regularization metrics, *e.g.* the angle between feature vectors [86, 217], or the geodesic distance between feature points [70, 187], are used to eliminate spurious skeleton details caused by noise on $\partial\Omega$. Field methods can also compute 3D curve skeletons by backprojecting 2D skeletons of 2D projections [150] or axis-aligned slices [228] of the shape back

Table 5: Recent 3D skeletonization papers. For each method, we show its type (**V**olume or **M**esh), and the surface- and/or curve-skeletonization methods it is compared with. Dashes show that a method does not compute the respective (curve or surface) skeleton type. Last three rows are survey papers.

Method		**Compared with**	
Type	Name	Surface skeleton	Curve skeleton
M	Jalba *et al.* [109]	[159]	[15, 187]
M	Giesen *et al.* [159]	[20, 49, 72]	–
M	Huang *et al.* [105]	–	[220]
M	Au *et al.* [15]	–	[58, 70, 90, 171]
M	Dey and Sun [70]	–	[58]
M	Tagliasacchi *et al.* [221]	–	[15, 70]
V	Arcelli *et al.* [11]	–	0
V	Reniers *et al.* [187]	0	0
V	Hesselink *et al.* [102]	0	–
V	Siddiqi *et al.* [204]	0	–
V	Liu *et al.* [149]	–	[171, 187]
V	Ju *et al.* [117]	[27]	[27]
M+V	Livesu *et al.* [150]	–	[15, 58, 70, 149]
M	Sobiecki *et al.* [207]	–	[15, 41, 220, 227]
M+V	Cornea *et al.* [59]	–	[8, 58, 90, 171]
V	**Our contribution**	[102, 117, 187, 204]	[11, 117, 149, 171, 187, 204]

into 3D. Field methods are implemented for both voxel and mesh shapes. **Thinning** methods remove $\partial\Omega$ voxels while preserving connectivity [62, 171]. Tools from mathematical morphology [197] were among the first used to compute curve skeletons by thinning. The residue of openings, based on Lantuéjoul's formula [136], usually leads to disconnected skeleton branches, whereas methods based on homotopic thinning transformations [29, 136, 158, 171] yield connected skeletons. Constraining thinning by distance-to-boundary order [11, 179, 219] or flux-order [175] further enforces centeredness. Further details on related work and such methods are given in Sections 7.3.2 and 2.2.2.

7.2.2 *The challenge of comparison*

Unsurprisingly, the wealth of existing skeletonization methods makes an exhaustive comparison hard. Aspects which contribute to this challenge are (a) different shape representations (voxels *vs* meshes *vs* point clouds), (b) the unavailability of several implementations, and (c) different skeleton definitions. The last aspect is particularly important: For *surface* skeletons, one could argue that Eqn. 7.2 is a unique definition against which all methods can be checked. However, both spatial discretizations of Eqn. 7.2 and heuristic regularizations that remove small-scale 'noise' details allow multiple weak forms of Eqn. 7.2 [102, 159, 187]. For *curve* skeletons, the problem is even harder, as these have no unique definition, not even in the continuous $\mathbb{R}^3$ space.

Such aspects make it hard to analytically compare, and reason about, the properties of the produced skeletons. As such, qualitative comparisons have been proposed.

In 2007, Cornea *et al.* compared four curve-skeletonization methods, one from each class listed in Sec. 7.2.1. To facilitate the comparison, they also propose several quality criteria that skeletons should obey. Six years later, this comparison was extended for six other contraction-based curve-skeletonization methods [207] (see also Chapter 6). Schaap *et al.* proposed a quantitative comparison of 13 centerline extraction algorithms for coronary artery datasets [193]. As reference, they use a centerline constructed by manual annotation by expert users. However, in contrast to the tubular artery shapes considered in [193], manual construction of curve, and even more so of surface, skeletons for general 3D shapes not feasible. A cursory scan over many skeletonization papers shows that such method comparisons are very limited (see Tab. 5). As such, more comparison studies are strongly needed to better understand the strengths and limitations of existing methods.

7.3 METHODS

We next describe a study that adds 4 surface and 6 curve skeletonization methods for voxel shapes to the existing comparison surveys mentioned in Sec. 7.2. Section 7.3.1 presents the desirable criteria that we compare against. Section 7.3.2 introduces the methods selected for comparison.

7.3.1 *Comparison criteria*

Following [59, 109, 203, 207], we focus on the following well-known quality criteria for curve and surface skeletons:

Homotopy: The skeleton is topologically equivalent to the input shape (same number of connected components, cavities, and tunnels).

Thin: The skeleton should be as thin as the sampling model used allows it. Voxel-based skeletons should be one voxel thick, *i.e.*, no 2×2 foreground-voxel configurations should exist.

Centered: For surface skeletons, this is equivalent to Eqn. 7.2. For curve skeletons, no unique centeredness definition exists. An useful weak form of curve-skeleton centeredness says that the curve skeleton should be a subset of the surface skeleton, since the latter is by definition centered in the shape [109, 221, 227].

Smoothness: As centeredness, smoothness is hard to formally define. Surface skeleton manifolds are at least $\mathscr{C}^2$ continuous [176, 203]. Curve-skeletons are centered subsets thereof [221, 227]. Hence, it is arguable that curve skeletons should be also per-branch $\mathscr{C}^2$. In any case, curve skeletons should not exhibit curvature discontinuities induced by the sampling of either the input surface or curve skeleton representation.

Regularization: Skeletons should capture fine-scale details, such as bumps or edges, of the input shape. Users should be able to select the scale of significant details which the skeleton should capture. All smaller-scale details are regarded as noise, and should thus be eliminated. This criterion subsumes the so-called noise robustness and detail preservation criteria. Since the definition of noise *vs* details is an application-

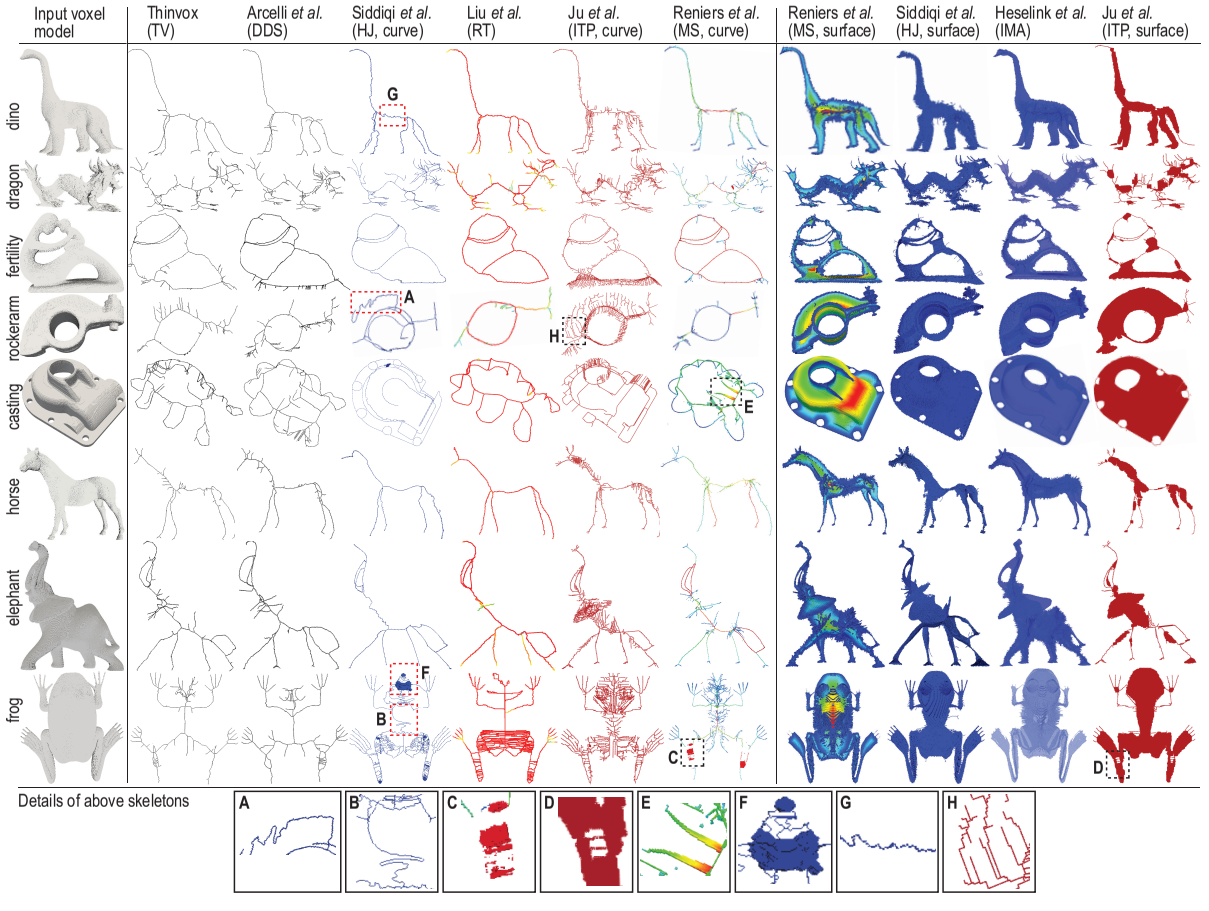


Figure 7.1: Curve and surface skeletons of different shapes, computed, from left to right, by 6 curve and 4 surface skeletonization methods (see Sec. 7.4). The input volume has a resolution of 512^3 voxels. Dotted markers indicate various problems of the produced skeletons and underlying methods. Details thereof are shown below the main image. (see Sec. 7.5.1).

dependent scale issue, we chose to use here instead the criterion of *regularization*, defined as the ability of user-controlled skeleton simplification [109, 151, 227].

Sampling robustness: The difference between skeletons computed for two voxel samplings of a shape should be proportional with the input-sampling differences.

Scalability: Skeletonization methods should be able to extract skeletons of large (1024^3 or similar) voxel volumes in (tens of) seconds on a modern PC with 16 GB RAM.

7.3.2 *Selected methods for comparison*

To select actual methods, we used the following criteria:

- *model:* We study only voxel-based methods. Mesh-based methods were recently separately covered in [207] (see also Chapter 6);
- *type:* We chose methods in each class (distance-field, general-field, and thinning). Geometric methods are not studied, as these typically use a mesh representation;
- *coverage:* We chose to compare several methods not surveyed by [59];
- *quality:* We chose methods whose advertised features match the quality criteria in Sec. 7.3.1, in particular, methods which can handle large voxel volumes;
- *generality:* We chose methods that handle any shapes, regardless of form, complexity, or genus. In particular, this eliminates methods that cannot handle shapes with tunnels, methods that only work for tubular shapes and/or branch-less shapes;
- *availability:* We chose methods with a public (or easily replicable) implementation, so our results can be verified.

Using these criteria, we selected 6 curve skeletonization (*CS*) and 4 surface skeletonization (*SS*) methods, as follows (note that some methods produce both curve and surface skeletons):

Integer Medial Axis transform (IMA, *CS*): Roerdink *et al.* proposed IMA, a distance-field method that computes one feature point per voxel (also called single-point feature transform [187]) of $\partial\Omega$. Regularized surface skeletons are found as those voxels whose 27-neighborhoods contain feature points located on $\partial\Omega$ further apart than a user-given value γ, similar to the distance-and-angle regularization in [217]. IMA has a time complexity that is linear in the number of input voxels, is very simple to implement, and can be easily parallelized.

Multiscale Skeletons (MS, *SS* and *CS*): Reniers *et al.* proposed MS [187], a general-field method that computes the surface skeleton following Eqn. 7.2 using the feature transform of [163]. Surface skeletons are regularized by an importance metric equal to the length of the shortest path on $\partial\Omega$ between feature points. Curve skeletons are detected as those surface-skeleton points admitting two different such shortest paths, following [70], and regularized based on the area enclosed by the two shortest paths

mentioned above. Thresholding the importance delivers a hierarchy of nested skeletons describing the shape at different scales. Skeleton connectivity is implied by the conjectured, but not proved, monotonicity of the importance metric.

Hamilton-Jacobi Skeletons (HJ, *SS* and *CS*): Siddiqi *et al.* proposed HJ, one of the first general-field methods. HJ detects medial points, which coincide with the shocks of the grassfire flow, as points where the average outward flux of the distance-transform gradient is non-zero [204]. A flux-ordered homotopy-preserving thinning is used to simplify the surface skeleton. With sufficient simplification, curve skeletons are obtained. The idea has been further enhanced with subpixel flux calculations andimproved with error correction in areas of large curvature [14]. To our knowledge, this enhancement has only been tested for 2D shapes. We implemented the curvature-correction enhancement for 3D shapes for our comparison.

Distance-Driven Skeletonization (DDS, *CS*): To extract surface skeletons, Arcelli *et al.* combine iterative thinning and distance-field methods to remove non-skeletal voxels while updating the distance transform based on a $\langle 3,4,5 \rangle$ scheme [11]. From these, curve skeletons can be further extracted by a similar thinning based on several heuristics that approximate the distance transform $DT_{\partial S}$ of the surface skeleton boundary ∂S over the surface skeleton S. Both skeleton types are further regularized by an importance metric similar in concept with, but implemented differently from, the collapse metric in [187]. DDS guarantees connected, voxel-thin, and rotation-invariant skeletons.

Thinvox (TV, *CS*): TV implements the 3D directional thinning proposed by Pálagyi and Kuba [171] to compute curve skeletons. Several computational optimizations are added, including GPU acceleration. TV is part of the *binvox* package [166]. Given the wide popularity and usage of *binvox*, we included TV in our comparison.

Iterative Thinning Process (ITP, *SS* and *CS*): Ju *et al.* compute, with ITP, skeletons of volumetric models by alternating thinning and a novel skeleton pruning routine [117]. ITP creates a family of skeletons parameterized by two user-specified values that determine respectively the size of curve and surface features on the skeleton. The method was, however, tested mainly on tubular-and-thin-plate shapes.

Robust Thinning (RT, *CS*): In [149], Liu *et al.* propose RT, a thinning method that works on cell-complex representations built using voxelization techniques. RT uses a 'medial persistence' importance metric that discriminates object parts with different anisotropic elongations, *e.g.*, tubes or plates, similar to ITP [117]. Based on medial persistence, RT can produce a continuum between surface-and-curve skeletons and 'pure' curve skeletons, and is claimed to be more robust to noise than ITP. In our comparison, we used only the curve skeletons produced by RT, as the other tested methods do not produce mixed skeletons.

7.4 COMPARISON METHODOLOGY

We used the selected methods to extract curve and surface skeletons from a set of 38 shapes available in PLY mesh format. Shapes range from simple to very complex in

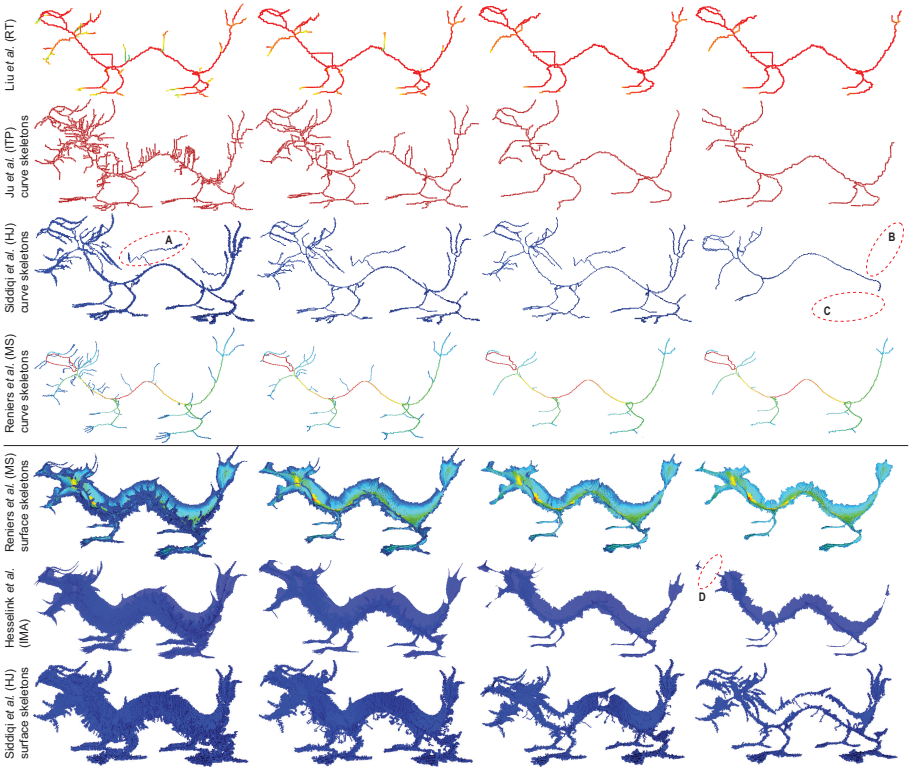


Figure 7.2: Regularization of curve skeletons (top 4 rows) and surface skeletons (bottom 3 rows). See Sec. 7.5.1.

terms of topology (branches and tunnels), surface detail, number of triangles (30K to over 1M), and cover both synthetic and natural objects. Figure 7.1 (left column) shows a selection. First, we used *binvox* [166] to voxelize the shapes to several resolutions ranging from 128^3 to 1024^3 voxels. Next, we ran each method on each voxel shape, tuning the method's regularization parameters (if any) so as to (a) eliminate spurious (noise) branches but (b) keep detail branches. In total, several hundreds of skeletonization runs were performed. Finally, for each method, and based on the values found in the previous pass, we chose a fixed set of parameter values that gives overall good results. These values were next used for all shapes treated by the respective method. Obtaining such results was quite easy for all studied methods, except ITP-curve, where we could not always eliminate spurious branches *and* keep important ones intact at the same time for all studied shapes. For testing, we used two 3.5 GHz, PC with 32GB RAM, running Windows 7 and Linux respectively, depending on the requirements of each method's implementation. The datasets and software implementations used are publicly available at [209].

7.5 RESULTS

Our results include two parts: A global comparison (Sec. 7.5.1) and a detailed comparison (Sec. 7.5.2). Both are described next.

7.5.1 *Global comparison*

Homotopy: All studied SS methods captured well the input shape topology, including protrusions and tunnels, and delivered connected skeletons (Fig. 7.1, 4 right columns). The only method that had issues here was ITP. The *frog* model (Fig. 7.1, bottom row) is an example of such issues: Although the input is of genus 0, the ITP surface skeleton exhibits a number of small spurious holes in the leg regions (Fig. 7.1, detail *D*). For curve skeletons, we see more variation: For most tubular shapes, all CS methods produce skeletons which match the input's topology. For non-tubular shapes, like *rockerarm* and *casting*, the topology of the produced results varies widely. The *frog* model is also the most challenging for CS methods: Although the model is relatively smooth and has, in most areas, a tubular structure, the computed curve skeletons vary strongly between all methods.

Thin: For our methods, this criterion implies one-voxel-thin skeletal manifolds and curve skeletons. Visual inspection shows that not all methods satisfy this. For surface skeletons, MS and HJ exhibit small-size thick clusters of several voxels around the manifold Y-intersection curves. IMA and ITP-surface perform the best. For curve skeletons, all methods produce voxel-thin curves, except HJ and MS, which create spurious thick *surface* fragments (see Fig. 7.1, details *E*, *F* on *casting* and *frog*). This is explained by the fact that, in contrast to the other studied methods, HJ and MS do not have an explicit thinning step or similar postprocessing to guarantee voxel-thin skeletons.

Centered: Visual comparison of the four SS types computed shows that these appear well centered within their input shapes. This may appear less evident for ITP (Fig 7.1,

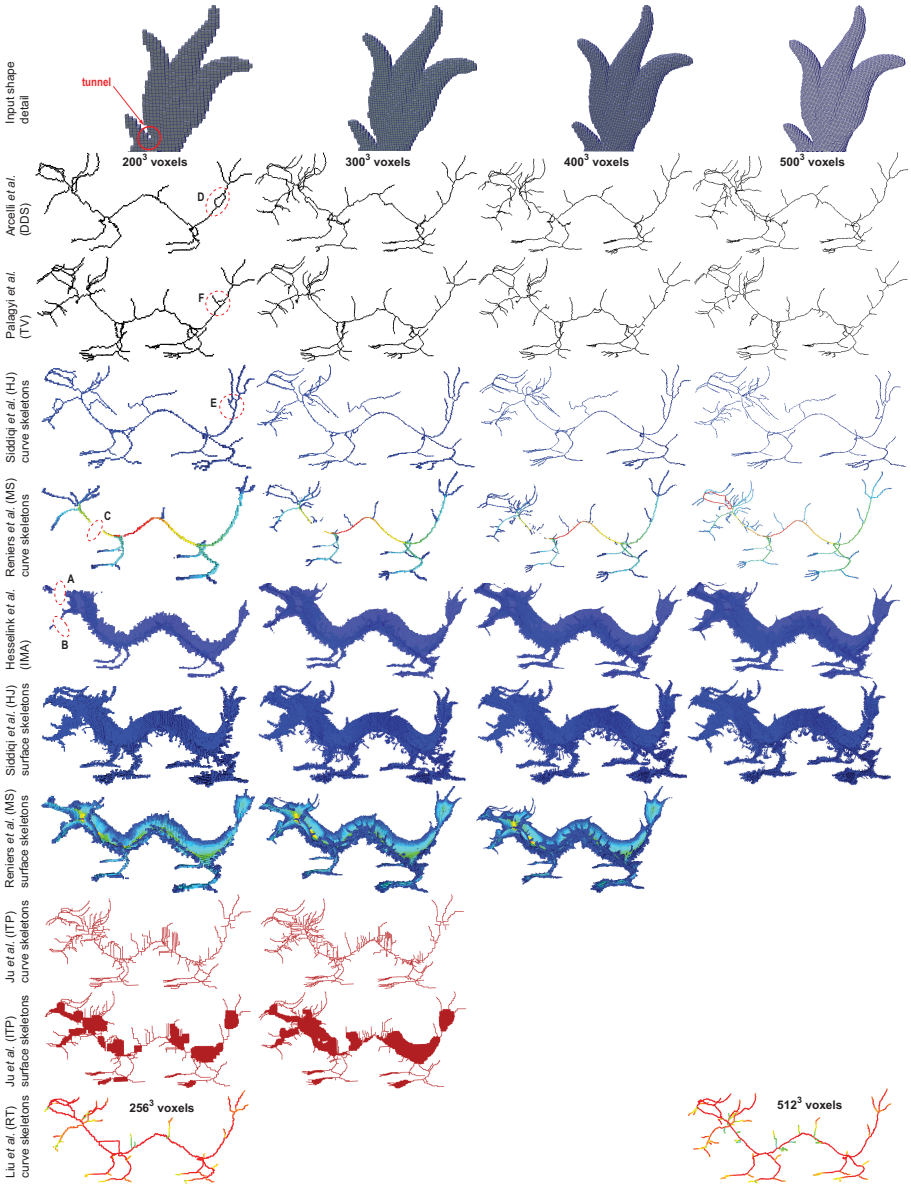


Figure 7.3: Sampling robustness of all studied methods for different voxel resolutions (see Sec. 7.5.1).

rightmost column). However, closer inspection shows that ITP differs from the other skeletons mainly around the boundaries of the skeletal manifolds, *i.e.*, in regions where different methods use different degrees of simplification. In 'core' areas, ITP skeletons are very similar to the other surface skeletons computed. For CSs, centeredness differences range from small for simple models (*horse*, *fertility*) to visibly

large ones (*rockerarm*, *frog*). This is partially expected, since all studied methods use different CS definitions. However, we also found CS fragments which arguably cannot be centered within any reasonable centeredness definition – see *e.g.* details *A*, *B* on *rockerarm* and *frog*, Fig. 7.1. To better assess centeredness, we propose detailed skeleton comparison further in Sec. 7.5.2.

Smoothness: The studied four SS methods exhibit negligible differences in terms of the smoothness of the extracted skeletal manifolds. In contrast, CS methods show a wide variation here. DDS, TV, and MS produced overall very smooth CS branches for all studied models. RT creates small-scale staircase effects, slightly larger than the voxel resolution used. However, these can be easily eliminated by increasing resolution. HJ creates several unexpected wiggles of various scales in the CS (*e.g.* Fig. 7.1, details *A*, *B*, *G* on *rockerarm*, *frog*, and *dino* respectively). We could not correlate the wiggles' appearance with shape properties such as curvature, thickness, smoothness, or voxel resolution. A possible explanation of these effects is that HJ is mainly designed to compute (simplified) *surface* skeletons. Highly simplifying these skeletons *can* produce curve skeletons, but the simplification order is *only* constrained by homotopy preservation. As such, HJ cannot guarantee that a skeletal manifold gets simplified with equal speed from all its boundaries inwards. When this does not happen, CS structures will still be contained by the SS, but not centered with respect to the SS boundaries. Finally, ITP creates coarse-scale staircase artifacts along CS branches (*e.g.* Fig. 7.1, *rockerarm*, detail *H*). This can explained by the fact that, unlike *e.g.* DDS or TV, ITP does not impose any geometry-based voxel removal order order during the thinning process.

Regularization: All studied SS and CS methods, except TV and DDS, offer one or more parameters to eliminate skeleton branches corresponding to small-scale noise on the input shape. TV does not need such a parameter for its curve skeletons, as these are noise-resistant by construction [171]. The DDS implementation we obtained from its authors uses the fixed simplification values indicated in [11]. To assess, in a global manner, the ease of noise elimination, we showed in Fig. 7.1 the results obtained by using a fixed set of regularization parameters, determined as described in Sec. 7.4.

Further increase of the simplification level should keep the most important skeletal branches, eliminating less important ones. To study the effect of the regularization parameters on simplification, we next varied these parameters for each method in order to progressively simplify the produced skeletons. Figure 7.2 shows this for four progressively simplified instances of the *dragon* skeleton for seven of the studied methods. Here, as we increase simplification, we notice growing differences between the studied methods. Among CS methods, MS and ITP offered the most intuitive way to eliminate small-scale details and keep the main skeleton structure. HJ-curve was the hardest to control: Too little simplification preserves spurious branches, such as the clearly not-centered branch on the dragon's back (Fig. 7.2, detail *A*). Increasing simplification removes this branch, but also looses important skeletal structures such as the dragon's legs and tail, which is an undesired effect (Fig. 7.2, details *B*, *C*). Among SS methods, MS and IMA simplify quite similarly, the effect being of removing SS voxels in increasing distance from the SS boundary. However, higher simplification values disconnect the IMA SS (Fig. 7.2, detail *D*). In contrast, MS never discon-

nected the SS, even for high simplification values. Simplifying HJ-surface skeletons also proved quite challenging: The divergence-based importance used by HJ is good for removing small-scale noise, but produces arguably incorrect results, like genus changes, when higher values are used (Fig. 7.2, bottom row). The explanation resides in the fact that the divergence metric does not have a monotonic evolution from the SS boundary inwards, but can exhibit various local maxima [14].

Sampling robustness: Figure 7.3 shows CS and SS results from the studied methods, each ran on several instances of *dragon*, sampled at resolutions between 200^3 and 512^3 voxels. For RT, we only used 256^3 and 512^3 voxels, since this method admits only power-of-two resolutions. We chose *dragon* for testing, as it is the shape having the highest amount of complex surface detail from our studied model collection. Regularization is tuned so as to obtain the visually most similar results for all methods. For both SS and CS methods, we see that, as resolution increases, progressively more skeletal details get captured, as expected. For SS methods and also for TV, MS-curve, and RT, as resolution increases, the overall skeleton shape does not change significantly. However, methods that not enforce homotopy (IMA and MS-curve) show skeleton disconnections at lower resolutions (Fig. 7.3, details *A*-*C*). These disappear when resolution is increased. The situation is more subtle for the remaining CS methods: HJ-curve, DDS, and TV show small-scale branch twists, and even topological events (loops) which appear and next disappear as resolution changes (Fig. 7.3, details *D*, *E*, *F*). Upon closer inspection, we see that these loops are correct, as they correspond to a tunnel in the input shape. In contrast, the lack of such a tunnel for MS is incorrect. For ITP-surface and ITP-curve, a large part of the SS, respectively CS structure varies with resolution in a hard-to-control manner. We also note that we could not run all methods for all resolutions – the empty places for MS and ITP in Fig. 7.3 correspond to resolutions for which the respective implementations crashed for the *dragon* model.

Scalability: Table 6 presents computational aspects of the tested methods for the models in Fig. 7.1. The same information is presented in Figure 7.4 by means of two graphs, one for speed and other for peak memory, for a better understanding. To make results independent of the input shape and its sampling, we computed *Speed* as the number of processed input foreground voxels per millisecond. This way, background voxels, which do not request computations in the tested methods, are ignored. In contrast, *Memory* gives the peak memory usage divided by the input volume size (voxels) for normalization, since the studied methods allocate data arrays of the size of the entire input volume (foreground and background voxels). Intuitively, *Speed* can be seen as the throughput of a given method, while *Memory* can be seen as the space cost per input voxel. The tested volumes range between 128^3 and 1024^3 voxels. For HJ, we include a single measurement for both the curve and surface skeletons, since both are computed using the same algorithm, the difference being only the simplification level applied as postprocessing. Absolute memory usage ranges from 1.5 GB (IMA, *fertility*) to 23.7 GB (HJ, *elephant*).

Several observations can be made. First, TV and IMA are the fastest methods. Even if factoring the possible lack of optimizations in the slower methods, this can be explained by the fact that TV and IMA use quite simple algorithms. More interestingly,

Table 6: Performance comparison for models in Fig. 7.1. *Speed* gives the number of processed foreground voxels per millisecond. *Mem.* gives the peak memory usage (bytes/voxel) per input volume size (see Sec. 7.5.1).

Methods	**TV**		**DDS**		**HJ curve/surface**		**RT**		**ITP, curve**		**MS, curve**		**MS, surface**		**IMA**		**ITP, surface**	
	Speed	Mem.	Speed	Mem.	Speed	Mem.	Speed	Mem.	Speed	Mem.	Speed	Mem.	Speed	Mem.	Speed	Mem.	Speed	Mem.
Dino	109.5	28.8	9.9	45.12	36.9	192	14.7	250	60.9	127	30.6	148	38.32	148	62.5	14.2	66.5	127
Dragon	61.2	24.6	10.6	46.8	14.8	597	10.8	175	25.7	127	7.2	135	27.2	120	45.8	11.9	25.7	127
Fertility	133.0	27.7	13.2	43.3	30.9	507	16.8	204	51.9	144	15.4	165	25.6	171	158.0	14.3	51.9	144
Rockerarm	111.4	28.4	14.3	50.7	52.3	270	22.3	203	51.9	114	17.9	161	13.5	161	99.4	13.9	49.3	120
Casting	109.8	31.7	12.6	45.6	49.7	322	19.7	203	48.6	114	9.3	124	29.1	124	87.5	14.1	53.2	115
Horse	120.5	29.2	10.2	47.3	52.1	270	25.9	204	54.9	125	13.7	145	32.9	157	94.3	14.0	54.9	127
Elephant	112.8	28.9	10.6	47.6	42.3	190	9.1	203	43.8	120	20.8	153	30.4	174	109.7	14.3	42.2	120
Frog	123.0	31.8	12.9	47.12	54.1	240	16.0	183	48.1	126	10.5	168	63.3	125	148.5	12.8	65.6	122

we see that the throughputs of all methods are relatively uniform, less so for HJ and MS. This indicates that most studied methods scale computationally well. For HJ, this can be explained by the particular homotopy-preserving thinning used, whose complexity depends on the number of detected skeleton end-points [14]. For MS, this is explained by the regularization metric used, which requires computation of geodesics between skeletal feature points. Both above operations are, indeed, strongly dependent on the input's shape, and not only on its size. Memory-wise, we see quite large differences: Here, again, TV and IMA require less than one order of magnitude less memory than the most expensive method, HJ. As for speed, this is explained by the relative simplicity of TV and IMA as compared to general-field methods – the former need only two such fields (input volume and skeleton), while the latter typically need to store many 3D floating-point fields over the entire volume.

A different outlier is visible for the *dragon* dataset, which is markedly slower to process than the other considered models. As *dragon* is the model having by far the most amount of surface detail, it will generate the most complex skeletons in terms of number of medial sheets or curves. Our observation that speed is related to skeleton complexity matches the similar separate observations in [58, 187]. However, we also see that the DDS and RT methods are less affected by the *dragon* model complexities in terms of performance. Although these methods are, on average, slower than the fastest considered methods, their throughput is much more *stable*, *i.e.* fluctuates less, for different models. This stability is an advantage in practice, as it allows one to estimate upfront the computational time required by for skeletonizing a given model.

7.5.2 *Detailed comparison*

The global comparison presented in Sec. 7.5.1 outlines differences between the studied methods in terms of all criteria in Sec. 7.3.1, except centeredness. Assessing centeredness differences from image pairs is harder, since such differences can be small-scale, local, and subtle. We next propose a visualization method that addresses the following centeredness questions:

- Given two surface skeletons SS_1 and SS_2, or two curve skeletons CS_1 and CS_2, which are the differences?
- Given a surface skeleton SS_1 and a curve skeleton CS_2, how well is CS_2 contained in SS_1?

Given two (curve or surface) skeletons S_1 and S_2, sampled over the same volume, we first define the scalar distance field

$$D_{12}(\mathbf{x} \in \mathbb{R}^3) = \begin{cases} \min_{\mathbf{y} \in S_2} \|\mathbf{x} - \mathbf{y}\| = DT_{S_2}(\mathbf{x}) & \text{if } \mathbf{x} \in S_1 \\ 0 & \text{if } \mathbf{x} \notin S_1 \end{cases}$$

To compare two skeletons of the same kind (CS_1 *vs* CS_2, or SS_1 *vs* SS_2), we draw the field $D_{12} + D_{21}$ over the voxel union $S_1 \cup S_2$, normalized by its maximum value, using a rainbow (blue-to-red) colormap. Voxels in a skeleton which are close to the other skeleton are blue. Note that $D_{12}(\mathbf{x}) = D_{21}(\mathbf{x}) = 0, \forall \mathbf{x} \in S_1 \cap S_2$. Voxels in a skeleton which are far away from the other skeleton are red (see Fig. 7.5, inset). Comparing a

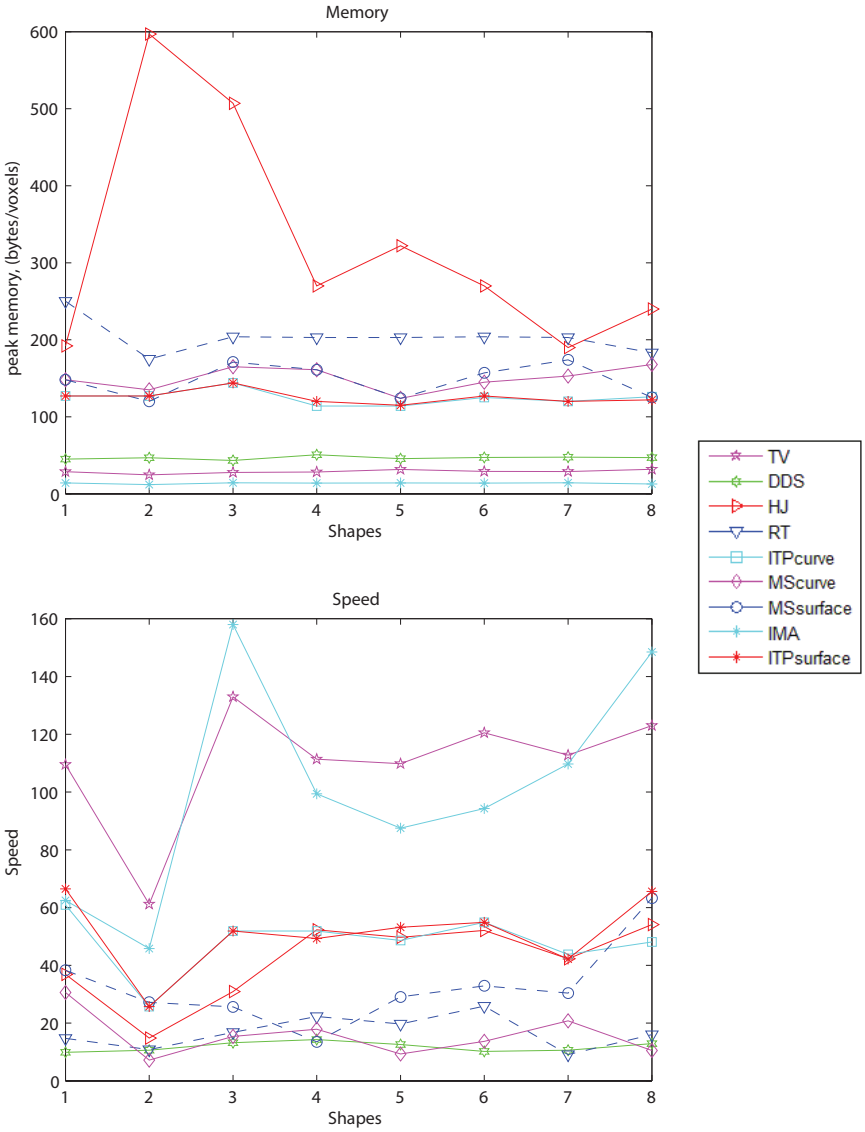


Figure 7.4: Top: Peak memory usage (bytes per voxel) for different skeletonization methods for the 8 shapes discussed in Tab. 6. Bottom: Skeletonization speed (number of foreground voxels processed per second) for the same methods and shapes.

curve skeleton CS_1 with a surface skeleton SS_2 is done differently, since we now want to show how well is CS_1 *contained* within SS_2. For this, we color voxels in CS_1 with D_{12}, and voxels in $SS_2 \setminus CS_1$ with gray. Voxels in CS_1 which are included in SS_2 are blue. Voxels in CS_1 far from SS_2 become red.

Figure 7.5 shows a subset of the performed comparisons, for several method-pairs, for the *dragon* model, at 500^3 resolution. Each row i or column j corresponds to a method; the image (i, j) shows the comparison of the skeleton-pair (S_i, S_j). Figure 7.6 shows the aggregated maximum and average distances between the considered skeleton pairs (S_i, S_j) for five different sampling resolutions. All distances are given as functions of the voxel size at resolution 100^3, *e.g.*, the size of a voxel at resolution 200^3 is $1/2$, the size of a voxel a resolution 300^3 is $1/3$, and so on, so that we can assess how distances change with resolution. From Figures 7.5 and 7.6, the following observations can be made.

Surface-*vs*-surface: Surface skeletons are very similar with each other, except at tips – see *e.g.* Fig. 7.5 (HJ *vs* IMA), which has an overall (dark) blue color. We noticed the same similarity in the other *SS*-*vs*-*SS* comparisons we did (not shown here for sake of space). The red tips in Fig. 7.5 (HJ *vs* IMA) indicate minor differences in terms of a few skeleton boundary voxels. These are expected given the different pruning heuristics of the considered methods. The fact that surface skeletons are quite similar strengthens (but does not prove) our hypothesis that centeredness is well captured – indeed, it would be surprising that methods using fundamentally different principles would yield the *same* errors.

Curve-*vs*-curve: Curve skeletons exhibit largest differences in terminal and central regions. In terminal regions, one skeleton can be longer than the other, as shown by the red tips of several skeleton branches. As for the *SS*-*vs*-*SS* case, this is due to the different pruning heuristics of the studied methods, and is an expected result which does not show lack of centeredness. However, along the dragon's central rump region, curve skeletons follow parallel, but quite different, paths. This means that *at least* one of the studied curve skeletons is poorly centered. This is a less expected insight, which cannot be inferred easily from the typical curve-skeleton images shown in typical skeletonization papers.

Curve-*vs*-surface: A perfect *CS*-in-*SS* inclusion is achieved only by the HJ curve and surface skeletons (*CS* voxels are all dark blue in Fig. 7.5, HJ-curve-*vs*-HJ-surface). This is expected, as both the *CS* and *SS* are computed by the same base method (HJ). For all other studied cases, we see a number of warm-colored curve-skeleton voxels (*e.g.* Fig. 7.5: TV-*vs*-HJ-surface, TV-*vs*-IMA, DDS-*vs*-IMA). Highest differences occur at skeleton tips, which is expected, as already explained. However, the pair HJ-surface-*vs*-TV shows such red voxels also deep inside the curve skeleton. Overall, we conclude that curve skeletons are generally well centered with respect to the medial surface, but less well centered *within* this surface.

Resolution: Figure 7.6 shows the variation of the maximal and average distances between the compared skeleton pairs for sampling resolutions ranging between 100^3 and 500^3 voxels. Several observations can be made. On average, the compared skeletons are quite close to each other (1 to 2 voxels), but the maximal distances can be large (up to 11 voxels). The minimal distances (not shown in the figure) are zero, for all considered skeleton-pairs. As resolution increases, most (but not all) compared skeletons tend to become closer, both in terms of average and maximal distances. This is not

surprising, since higher resolutions should allow a finer-grained placement of skeleton voxels, thus a better approximation of the actual skeletal locus. However, as the measured differences do not monotonically decrease with resolution in all cases, this suggests that there exist structural differences between the skeletons computed by the studied methods, which cannot be solved simply by a finer sampling. Interestingly, HJ-surface is the method which participates in the most similar skeleton-pairs, both in terms of average and maximal distances, for all resolutions. Conversely, IMA participates in the least-similar skeleton pairs. If we assume that all methods are, statistically speaking, equally valid with respect to centeredness, this implies that HJ-surface delivers a well-centered 'consensus' skeleton, and IMA delivers an outlier, far less well centered, skeleton.

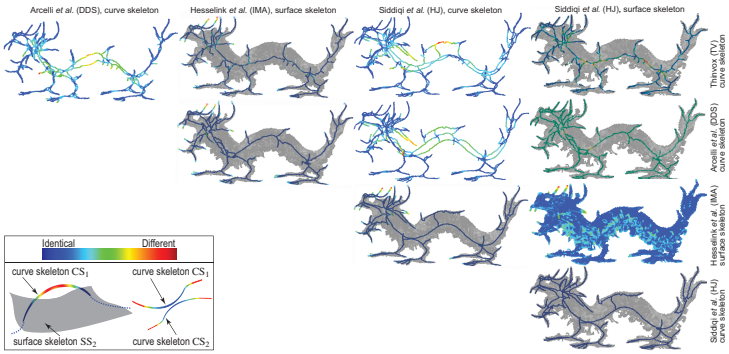


Figure 7.5: Detailed pair-wise comparison of curve *vs* curve, curve *vs* surface, and surface *vs* surface skeletons. Inset shows color mapping method.

7.6 DISCUSSION

Several points emerge from our comparison, as follows:

Best method: First, let us stress that the aim of our comparison was not to designate the 'best' skeletonization method, but to highlight pro's and con's of the studied methods with respect to a set of accepted quality criteria. From this viewpoint, no CS or SS method can be seen as optimal with respect to *all* considered criteria. For surface skeletons, all methods create well-centered, noise-free, skeletons. Noise (or detail) removal is the most intuitive with MS. The other studied methods can remove noise or details, but are less intuitive to control (produce disconnections, modify the topology, or remove skeletal parts which one may regard as 'core' to the skeleton). For curve skeletons, DDS and TV are simple to use (require no parameters), and produce clean, noise-free, but still detailed, and reasonably smooth, curve-skeletons. However, they offer less freedom for skeleton simplification. MS-curve arguably produces the smoothest curve-skeletons, but does not guarantee voxel-thickness. IMA produces centered, smooth, skeletons, and is very simple to use, but has less intuitive simplification parameters. HJ can produce high-quality surface skeletons. However, HJ cannot produce centered curve skeletons for more complex shapes, and also has

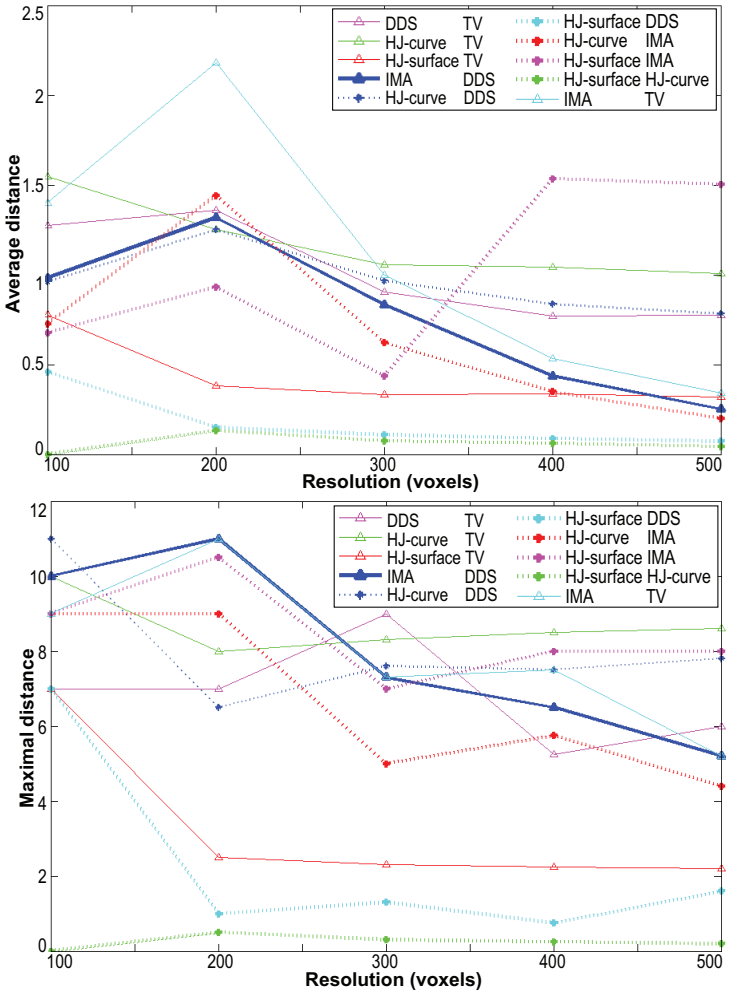


Figure 7.6: Average and maximal distances between skeletons computed by different methods at different resolutions (see Fig. 7.5).

a less intuitive simplification control. Performance-wise, TV and IMA are the fastest methods, and the least demanding in memory terms. However, the differences with the other studied methods are not that large so as to warrant a clear winner. To aid in doing this comparison, Table 7 summarizes the observed qualitative behavior of the studied methods with respect to the skeletal desirable criteria, based on the five-point ordinal quality scale introduced earlier in Sec. 6.5.

Criteria: The covered comparison criteria are clearly not exhaustive. Additional ones exist, *e.g.*, skeleton invariance to isometric transformations of the input shape [11, 59] and input reconstructibility from the skeleton [11, 109]. Given the available space, we chose to focus in more depth on a smaller number of criteria. Studying how CS and

SS methods perform on additional criteria is subject for separate future work.

Table 7: Qualitative evaluation of skeletonization methods.

Methods against Criteria							
Methods	Homotopy	Thin	Centered	Smoothness	Regularization	Sampling robustness	Scalability
TV	Good	Best	Best	Best	-	Best	Bad
DDS	Good	Best	Best	Best	-	Best	Good
HJ curv.	Satisfactory	Good	Best	Satisfactory	Satisfactory	Best	Satisfactory
RT	Good	Best	Best	Good	Best	Bad	Satisfactory
ITP curv.	Good	Best	Best	Good	Best	Bad	Satisfactory
MS curv.	Satisfactory	Good	Best	Good	Best	Bad	Satisfactory
MS surf.	Best	-	Best	Good	Best	Good	Satisfactory
HJ surf.	Best	-	Best	Good	Satisfactory	Best	Satisfactory
IMA	Best	-	Best	Good	Good	Good	Bad
ITP surf.	Satisfactory	-	Best	Satisfactory	-	Bad	Satisfactory

Methods: Our selection of compared methods cannot cover all existing CS and SS techniques in existence, so our findings cannot be directly generalized to any method. However, our comparison outlined several non-evident challenges of a good representative subset of recent methods. Most of the studied methods could extract curve and surface skeletons from a large variety of complex shapes. However, we also discovered several problems with respect to all considered quality criteria (except scalability), which are visible only for certain combinations of complex input shapes and method-specific parameter settings. As such, we argue that more comparative studies are required for a better understanding of the added value and limitations of skeletonization methods.

Standard: The various discretization and regularization techniques used in skeletonization algorithms, together with the lack of a unique formal CS definition, make comparisons of a given algorithm with a 'gold standard' difficult. Evident problems, such as thick or disconnected skeletons, are easy to check for. However, checking criteria such as smoothness, noise robustness, and CS centeredness, is much harder. As such, the question of what is the 'correct' skeleton of a given shape is very hard to answer in general. Rather than trying to answer this question, we advocate a comparative approach that highlights *differences* between several skeletonization methods with respect to input shape, input resolution, and simplification parameters.

7.7 CONCLUSION

In this chapter, we presented a comparison of six curve-skeletonization and four surface-skeletonization methods using voxel models. Compared to existing surveys in the area [59, 207], we extend insights by discussing ten methods (not covered by previous surveys) with respect to established quality criteria for curve and surface skeletons. We compare methods on a range of 3D shapes ranging from simple to complex, covering both natural and synthetic forms, and consider the effects of several parameters such as simplification level and input resolution on the obtained skeletons. We include also a quantitative performance in terms of speed and memory requirements. Finally, we propose a detail visualization able to highlight small-scale

centeredness differences between curve and surface skeletons. Our work highlights challenges of, and differences between, existing 3D skeletonization methods which to our knowledge have not been highlighted in the literature. On a higher level, our results expose several limitations of current skeletonization methods and underline the need for future work towards extending such comparisons and also towards creating better methods.

Future work in skeletonization comparison involves including additional methods and quality criteria in this comparison. On a more theoretical level, a promising direction is to devise new metrics for the quantitative comparison of the desired quality criteria in ways that help algorithm designers pinpoint and next solve causes for current limitations of such methods.

The joint comparisons performed in this chapter and Chapter 6 cover, fundamentally, all types of skeletons (curve and surface) and spatial samplings (mesh-based and voxel-based), except mesh-based surface skeletons. For this latter category, a separate comparison is not required, as very few such methods exist [109, 151, 159]. These methods are further compared in [109], and the results of this comparison indicate that all these methods are far from trivial to implement. As such, we rule them out as potential candidates for our shape restoration use-case.

This chapter is based on:

A. Sobiecki, A. Jalba, and A. Telea Comparison of curve and surface skeletonization methods for voxel shapes. *Pattern Recognition Letters, 47:147–156*, 2014.

8 AN UNIFIED METHOD FOR MULTISCALE PLANAR, CURVE, AND SURFACE SKELETONIZATION

The joint comparisons performed in Chapters 6 and 7 assessed sixteen skeletonization methods from the perspective of well-known quality criteria for skeletons and skeletonization methods. The general conclusion of this extended comparison work is that no single skeletonization method complies with all our desirable features. In addition, existing mesh-based surface skeletonization methods [109, 151, 159] are challenged by too complex implementations to be practical for our context where we require a simple to use and deploy tool. In addition to the above, we note that the *multiscale regularization* requirement is satisfied by a small minority of methods [80, 109, 168, 187, 228]. Interestingly, all these methods use the same conceptual idea of collapsed boundary importance (see Sec. 2.2.2), but each method proposes a different formulation of how to define and compute this importance. This makes comparing such methods additionally difficult. In this chapter, we aim to jointly solve the problems of (a) lack of an optimal 3D skeletonization method for our shape restoration use-case, and (b) lack of unified multiscale regularization of all existing skeleton types (2D planar, 3D curve, and 3D surface).

8.1 INTRODUCTION

Skeletons, or medial axes, are shape descriptors used in virtual navigation, shape matching, shape reconstruction, and shape processing [203]. 3D shapes admit two types of skeletons. *Surface skeletons* are 2D manifolds which contain the loci of maximally-inscribed balls in a shape [176, 203]. *Curve skeletons* are 1D curves which are locally centered in the shape [58]. Surface-skeleton points, with their distance to the shape and closest-shape points, define the medial surface transform (MST), used for animation, smoothing, and matching [13, 17, 79].

Many methods exist for computing 2D skeletons [79, 168, 228], 3D surface skeletons [102, 187, 204, 215], and 3D curve skeletons [15, 70, 98, 220]. Although recent methods demonstrate high accuracy, insensitivity to noise, and computational efficiency, several challenges remain open. We focus here on two modeling challenges, as follows. First, 2D skeletons, 3D surface skeletons, and 3D curve skeletons are typically extracted, and next simplified, using *different* methods and metrics. This makes the comparison and the formal reasoning about the properties of the extracted skeletons difficult. Secondly, few (if any) methods offer a continuous multiscale representation that addresses *all* skeleton types, *i.e.*, a model which encodes both the geometric *importance* of any skeleton point (useful for simplifying, or regularizing, noisy skeletons) and the *type* of skeleton point (non-skeleton, surface skeleton, or curve skeleton).

In this chapter, we present a framework for 2D and 3D curve-and-surface skeletonization that addresses the above two goals. We model both the skeleton detection and its importance using an advection principle that collapses mass from a shape boundary to its skeleton and next to the skeleton center (in 2D); and from the bound-

ary to the surface skeleton, next to the curve skeleton, and finally to the latter's center (in 3D). This allows us to detect all types of mentioned skeletons, and also to regularize them, *e.g.*, to remove detail branches, via a single model and a simple thresholding operation. We propose a single algorithm that unifies skeleton detection and regularization in 2D and 3D, and also establishes a formal connection between surface and curve skeletons. Our method is simple to implement, computationally efficient, and easy to use. We show that our results are very similar to the ones produced by several existing 2D and 3D skeletonization methods on a set of complex shapes.

The structure of this chapter is as follows. Section 8.2 reviews related work. Section 8.3 presents our skeletonization method. Section 8.4 details our method's implementation. Section 8.5 compares our results with one 2D, six 3D surface, and 11 curve skeletonization methods. Section 8.6 discusses our results. Section 8.7 concludes the chapter.

8.2 RELATED WORK

Given a shape $\Omega \subset \mathbb{R}^n$, $n \in \{2,3\}$ with boundary $\partial\Omega$, we first define its Euclidean distance transform $DT_{\partial\Omega} : \mathbb{R}^n \rightarrow \mathbb{R}^+$

$$DT_{\partial\Omega}(\boldsymbol{x} \in \Omega) = \min_{\boldsymbol{y} \in \partial\Omega} \|\boldsymbol{x} - \boldsymbol{y}\|. \tag{8.1}$$

The skeleton, or medial axis, of Ω is next defined as

$$S_\Omega = \{\boldsymbol{x} \in \Omega | \exists \boldsymbol{f}_1, \boldsymbol{f}_2 \in \partial\Omega, \boldsymbol{f}_1 \neq \boldsymbol{f}_2, \|\boldsymbol{x} - \boldsymbol{f}_1\| = \|\boldsymbol{x} - \boldsymbol{f}_2\| = DT_{\partial\Omega}(\boldsymbol{x})\} \tag{8.2}$$

where $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are the contact points with $\partial\Omega$ of the maximally inscribed balls in Ω centered at $\boldsymbol{x}$ [93, 187]. The points $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are called *feature transform* (FT) points [216]. The vectors $\mathbf{f} - \mathbf{x}$ are called *spoke vectors* [214]. For $n = 2$, S_Ω is a set of curves which meet at the so-called skeleton junction points [79]. For $n = 3$, S_Ω is a set of manifolds with boundaries which meet along a set of so called Y-intersection curves [48, 66, 141].

In contrast to 2D and surface skeletons (Eqn. 8.2), 3D curve skeletons CS_Ω admit many definitions [58], implemented by a wide variety of methods (discussed further below). As such, a formal relationship between S_Ω and CS_Ω is still not unanimously accepted. For instance, although it is commonly accepted that CS_Ω should be centered within S_Ω, only few skeletonization methods use and/or enforce this property [187, 227].

Skeletons can be computed by various methods, as follows.

Thinning: Thinning removes $\partial\Omega$ voxels (or pixels in 2D) while preserving connectivity [18, 171, 179]. Although simple and fast, thinning can be sensitive to Euclidean transformations.

Field methods: These methods find S_Ω along singularities of $DT_{\partial\Omega}$ or related fields [91, 102, 125, 142, 189, 228, 236] and can be efficiently done on GPUs [41, 216, 217]. General-field methods use fields smoother (with fewer singularities) than distance transforms [5, 13, 58, 98], and thus are more robust for noisy shapes. Siddiqi *et al.* find the skeleton as the non-zero divergence locus of $\nabla DT_{\partial\Omega}$ [204]. However,

$\nabla \cdot (\nabla DT_{\partial\Omega})$, with $\nabla\cdot$ the divergence operator, can be non-zero also at non-skeletal points. Torsello and Hancock correct this for a more accurate 2D skeleton detection by a momentum conservation principle $\nabla \cdot (\rho \nabla DT_{\partial\Omega}) = 0$, where ρ is the mass density on the evolving boundary $\partial\Omega$ [14]. Rossi and Torsello extend this idea to compute 3D surface skeletons [188]. However, this method does not compute curve skeletons and does not model the curve-surface skeleton relationship.

Mesh-based methods: Field methods volumetrically sample Ω, which can be expensive memory-wise. Mesh-based methods use a surface sampling of $\partial\Omega$, which allows processing higher-resolution shapes. Mesh methods include Voronoi diagrams to compute polygonal skeletons [71]. Amenta *et al.* compute the Power Crust, an approximation of a surface and its medial axis by a subset of Voronoi points [8]. Other methods use edge collapses [146], starting from a mesh segmentation [120]. Surface skeletons can be extracted from oriented point clouds [109, 151] or polygon meshes [141, 161] by searching for maximally inscribed balls tangent at at least two shape points. Curve skeletons can be extracted from point clouds as centers of cloud projections on a cut plane which optimizes for circularity [220]. *Contraction* techniques are a separate subclass of mesh methods. Like field techniques, they evolve $\partial\Omega$ under various types of normal flows, effectively collapsing it onto the surface-or-curve skeleton. Methods using a (constrained) Laplacian contraction by mean curvature flow deliver high-quality curve skeletons [15, 40, 52], or even 'meso skeletons' mixes of surface and curve skeletons [221]. A different approach is taken by Jalba and Telea who contract the surface skeleton to compute its curve skeleton counterpart [227]. A recent review of contraction methods is given in [207].

Multiscale skeletons: Clean skeletons are extracted from noisy shapes by thresholding *importance measures* $\rho : \Omega \to \mathbb{R}^+$. This prunes skeletal branches caused by small details [66, 199]. We distinguish between local and global measures [159, 187]. *Local measures* cannot separate locally-identical, yet globally-different, contexts (see *e.g.* [187], Fig. 1). Thresholding local measures can disconnect skeletons. Reconnection needs extra work [155, 176, 204, 217], and makes pruning less intuitive [199]. Local measures include the angle between the feature points and distance-to-boundary [8, 86, 217], divergence-based [36, 204], first-order moments [189], and points where $\nabla DT_{\partial\Omega}$ is multi-valued [214, 215]. Leymarie and Kimia topologically simplify point-cloud skeletons to capture Y-intersection curves and skeleton sheet boundaries in medial scaffolds [141]. A good survey of such methods is given in [203].

Global measures monotonically increase from the skeleton boundary ∂S_Ω inwards. Thresholding them yields connected skeletons which capture skeleton details at a user-given scale. Miklos *et al.* approximate shapes by unions of balls (UoB) and use UoB medial properties [94] to simplify skeletons [159]. Dey and Sun introduce the medial geodesic function (MGF), equal to the shortest-geodesic length between feature points [70, 178]. Reniers *et al.* [187] extend the MGF for surface and curve skeletons using geodesic lengths and surface areas between geodesics, respectively, inspired by the so-called collapse metric used to extract multiscale 2D skeletons [79, 168, 228]. A fast GPU implementation of this extended MGF is given in [109].

The MGF and its 2D collapse metric counterpart have an intuitive geometric meaning: They assign to a skeleton point $\mathbf{p}$ the amount of shape boundary that corresponds,

or 'collapses' to, **p** by some kind of boundary-to-skeleton mass transport. Skeleton points **p** with low metric values correspond to small-scale shape details or noise; points **p** with large metric values correspond to large-scale shape details. This allows an easy simplification of the skeleton: Thresholding by a value τ eliminates all skeleton points which encode less than τ boundary length or area units. If the collapse metric monotonically increases from the skeleton boundary to its center, thresholding delivers a set of connected and nested skeleton approximations, also called a multi-scale skeleton [70, 79, 187, 228].

8.3 PROPOSED FRAMEWORK

8.3.1 *Preliminaries*

Following the above, we aim to create a single model that

1. unifies the representation and detection of 2D skeletons, 3D surface skeletons, and 3D curve skeletons;

2. computes a monotonic, global importance metric for 2D and 3D skeleton regularization and simplification;

Conceptually, we aim to capture the desirable properties of the 2D and 3D boundary collapse metric [79, 168, 187, 228] in a single model, and also connect contraction-based and distance-field based methods in a single framework. Practically, we aim at a single, easy to implement and use, and computationally efficient method that extracts and regularizes all skeleton types.

To achieve this, we first introduce our unified skeleton definition: Given a shape $\Omega \in \mathbb{R}^{n\in\{2,3\}}$, we aim to compute an importance function $\lambda : \Omega \to \mathbb{R}^+$ so that the threshold sets $\lambda_\tau = \{\mathbf{x} \in \Omega | \lambda(\mathbf{x}) \geq \tau\}$ capture all existing skeleton types *and* all their simplifications. Specifically, we want λ_0 to be the full input shape Ω; λ_ε (for a small $\varepsilon > 0$) to be the full (unsimplified) surface skeleton S_Ω, which implies that $\lambda(\mathbf{x}) = 0, \forall \mathbf{x} \notin S_\Omega$. As τ increases, we want λ_τ to be progressively simplified surface skeletons, and as τ increases even further, progressively simplified curve skeletons. In the limit, when $\tau = \max_{\mathbf{x}\in\Omega} \lambda(\mathbf{x})$, we want λ_τ to be a single point (for genus 0 objects), which we next call the shape center C_Ω. This process can be seen as a 'recursive' skeletonization which computes the surface skeleton from the input shape, the curve skeleton from the surface skeleton, and the shape center from the curve skeleton. All skeletons λ_τ should satisfy the well-known desirable properties – centeredness, rotational invariance, homotopy to the input shape Ω, noise robustness, one-pixel (in 2D) and one-voxel (in 3D) thickness, inclusion of the curve skeleton in the surface skeleton, and computational efficiency [58, 207, 210].

8.3.2 *Physically-based skeletonization model*

For a shape $\Omega \in \mathbb{R}^{n\in\{2,3\}}$, we model our unified skeletonization as a contraction of Ω on whose boundary mass is distributed with unit density. Contraction is described by three fields: $\phi(\boldsymbol{x},t)$, $\rho(\boldsymbol{x},t)$, and $\boldsymbol{u}(\boldsymbol{x},t)$, with $\mathbf{x} \in \Omega$, and with $t \in \mathbb{R}^+$ being the time parameter, as follows. Similar to phase-field models [30], the field $\phi \to [-1,1]$ is

1 inside Ω and -1 outside, so that the boundary of the contracting shape is implicitly given by $\Gamma_t = \{\boldsymbol{x} \in \Omega \,|\, \phi(\boldsymbol{x},t) = 0\}$. For now, we assume that ϕ varies abruptly and monotonically over $[-1,1]$ in a small vicinity around Γ_t. The field $\rho \to \mathbb{R}^+$ gives the mass density of Γ_t. Finally, $\boldsymbol{u} \to \mathbb{R}^n$ gives the contraction direction of Γ_t.

Our contraction is described by a system of three PDEs:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{8.3}$$

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0, \;\; \text{with } \Gamma_t \cong \Gamma_0, \tag{8.4}$$

$$\mathbf{u} = \frac{\nabla \phi}{\|\nabla \phi\|}. \tag{8.5}$$

Equation 8.3 imposes mass conservation on the shrinking boundary. Equation 8.4 models boundary contraction with the topological homeomorphic constraint $\Gamma_t \cong \Gamma_0$. This ensures that the computed skeletons are homotopic to the input shape $\partial\Omega = \Gamma_0$. Equation 8.5 imposes inwards contraction of our shape, with unit speed in normal direction to Γ_t.

Eliminating $\mathbf{u}$ from Eqns. 8.3-8.5, we obtain

$$\frac{\partial \rho}{\partial t} = -\nabla \rho \cdot \frac{\nabla \phi}{\|\nabla \phi\|} - \rho \nabla \cdot \frac{\nabla \phi}{\|\nabla \phi\|} = -\nabla \rho \cdot \frac{\nabla \phi}{\|\nabla \phi\|} - \rho \kappa \tag{8.6}$$

$$\frac{\partial \phi}{\partial t} + \|\nabla \phi\| = 0, \;\; \text{with } \Gamma_t \cong \Gamma_0 \tag{8.7}$$

where κ is the (mean) curvature of Γ_t.

Equations 8.6-8.7 are supplemented by the initial conditions

$$\phi(\mathbf{x}, t=0) = \begin{cases} 1, & \text{if } \mathbf{x} \in \Omega \\ -1, & \text{if } \mathbf{x} \notin \Omega \end{cases} \tag{8.8}$$

$$\rho(\mathbf{x}, t=0) = \begin{cases} 1, & \text{if } \mathbf{x} \in \partial\Omega \\ 0, & \text{if } \mathbf{x} \notin \partial\Omega. \end{cases} \tag{8.9}$$

Let us define the time-of-arrival function $T : \Omega \to \mathbb{R}^+$ so that

$$\phi(\mathbf{x},t) = T(\mathbf{x}) - t. \tag{8.10}$$

Hence, $\Gamma_t = \{\mathbf{x} \in \Omega \,|\, T(x) = t\}$, *i.e.*, $T(\mathbf{x})$ is the time after which Γ_t passes through $\mathbf{x}$. Using Eqns. 8.7 and 8.10, we obtain $\|\nabla T\| = -\phi(\cdot,t) = 1$, the well-known Eikonal equation for arrival time T. The Euclidean distance transform $DT_{\partial\Omega}$ is the weak solution of this equation under Euclidean norm [198]. Hence, Eqn. 8.7 without the constraint $\Gamma_t \cong \Gamma_0$ is the PDE generating continuous multi-scale (flat) morphological erosions. Other norms are also possible [156], leading to various distance transforms.

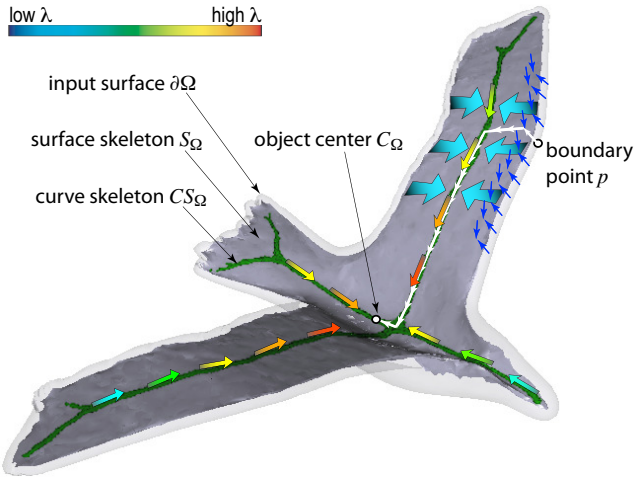


Figure 8.1: Density advection model from the surface $\partial\Omega$ of a genus-0 input shape to its surface skeleton S_Ω, curve skeleton CS_Ω, and object center C_Ω. Colors depict the importance λ of different spatial regions.

We finally define the skeleton importance λ as the maximum density that has reached a certain location $\mathbf{x} \in \Omega$, *i.e.*,

$$\lambda(\mathbf{x}) = \max_{t>0} \rho(\mathbf{x}, t). \tag{8.11}$$

Intuitively, our model describes a conservative advection process where mass, uniformly spread on $\partial\Omega$, flows on shortest paths from $\partial\Omega$ to its surface skeleton S_Ω; then, along S_Ω on shortest paths to the curve skeleton CS_Ω; and finally along CS_Ω on shortest paths to the shape center C_Ω (Fig. 8.1). Once all mass has reached C_Ω, we compute the (simplified) surface and curve skeletons by thresholding λ at increasing values.

8.4 SOLVING THE SYSTEM

To compute the importance λ, we solve the contraction model in Sec. 8.3.2 by discretizing Ω on a uniform cubic-cell (pixel or voxel) grid embedded in $\mathbb{R}^2$ and $\mathbb{R}^3$ respectively, as follows.

8.4.1 *Topologically-constrained boundary evolution by density-ordered thinning*

As stated in Sec. 8.3.2, Eqn. 8.7 must be solved with the constraint that Γ_t and Γ_0 are homeomorphic, for all t. Even without this constraint, it is well-known that the evolution of ϕ from Eqn. (8.7) develops discontinuities of the ϕ derivatives (shocks) within finite time [156, 198]. The skeleton S_Ω precisely coincides with the locations of these shocks [204].

Since Eqn. 8.7 can be written as an Eikonal evolution, or boundary-value problem (Sec. 8.3.2), one way to interpret the contraction is as thresholding $DT_{\partial\Omega}$ in-

side Ω at increasingly higher values, producing multi-scale morphological erosions of Ω. Additionally, the topological constraint $\Gamma_t \cong \Gamma_0$ should also be satisfied by each level set of $DT_{\partial\Omega}$ corresponding to Γ_t. For achieving this, we could consider using topologically-constrained level sets [97]. The problem with this approach is that it first performs an *un-constrained* step to update level-set values, following the motion equation (Eqn. 8.7). Then, at points $\mathbf{x}$ where the topological constrained is violated, the so-called level-set function ψ is next 'fixed' so that the points $\mathbf{x}$ lie on the corresponding side of the boundary dictated by the constraint. This fix *artificially* alters the ψ values, which creates spurious and unwanted discontinuities in ψ, ultimately leading to a not sharply-defined (in terms of our desired sharp transition of the level-set function in $[-1,1]$) and/or non-smooth evolution of Γ_t. In turn, this will drastically affect the quality of the extracted skeletons, as we verified in practice. As such, we chose not to use topologically-constrained level-sets for our context.

To handle all above issues, we use topology-preserving morphological thinning to define and steer the evolution (contraction) of Γ_t. Our thinning process is ordered both by $DT_{\partial\Omega}$ and by the density field ρ: As long as Γ_t is far from the skeleton S_Ω, ordering by $DT_{\partial\Omega}$ ensures a smooth Γ_t while solving Eqn. 8.7. Additionally, since thinning relies on a binary field, Γ_t is maintained sharp during its evolution.

We next explain why the thinning order is also given by the density field ρ, which is crucial when the evolving Γ_t reaches the (yet unknown) skeleton locations. Recall that such locations correspond to shocks of Eqn. 8.7. Hence, ordering by $DT_{\partial\Omega}$ (which is just a viscosity solution of Eqn. 8.7) becomes meaningless. As sketched in Fig. 8.1, we want the importance λ, and thus also the density ρ which determines this importance, to monotonically increase from $\partial\Omega$ to S_Ω, next from ∂S_Ω to CS_Ω, and finally from ∂CS_Ω (curve-skeleton endpoints) to C_Ω. Since Γ_t shrinks in normal direction (Eqn. 8.7), this is equivalent to transporting density on shortest paths from $\partial\Omega$ to S_Ω to CS_Ω and next to C_Ω. Fig. 8.1 shows such a path (in white) on which the mass of a point $\mathbf{p} \in \partial\Omega$ should flow during its advection to C_Ω. Consider now the set of all such paths from all points on $\partial\Omega$ to C_Ω. For a shape Ω of genus 0, following a reasoning similar to [187], these paths will form a tree having as leaves all (discrete) points of $\partial\Omega$ and C_Ω as root. The computation of ρ by means of our contracting Γ_t is analogous to traversing this tree from its leaves to the root. To ensure a correct density update, we thus need that, at any junction-point where several subtrees meet, all these trees to have been fully traversed and their roots' densities to be thus correctly updated. This is why our thinning visits points in Γ_t in increasing ρ order.

Figure 8.2 illustrates our thinning for a 3D shape. When using density-ordered thinning, Γ_t (drawn red) is kept smooth during collapsing. In contrast, if not using density ordering, the collapsing Γ_t will quickly develop irregularities (Fig. 8.2 e-h, insets). In turn, these will create irregularities in the signal ρ which will ultimately lead to jagged skeletons λ_τ after simplification.

8.4.2 *Algorithm*

Summarizing the observations from Sec. 8.4.1, our contraction algorithm should:

R_1: provide a sharp definition of the evolving boundary Γ_t;

R_2: allow interleaved iterative solves of Eqns. 8.6 and 8.7;

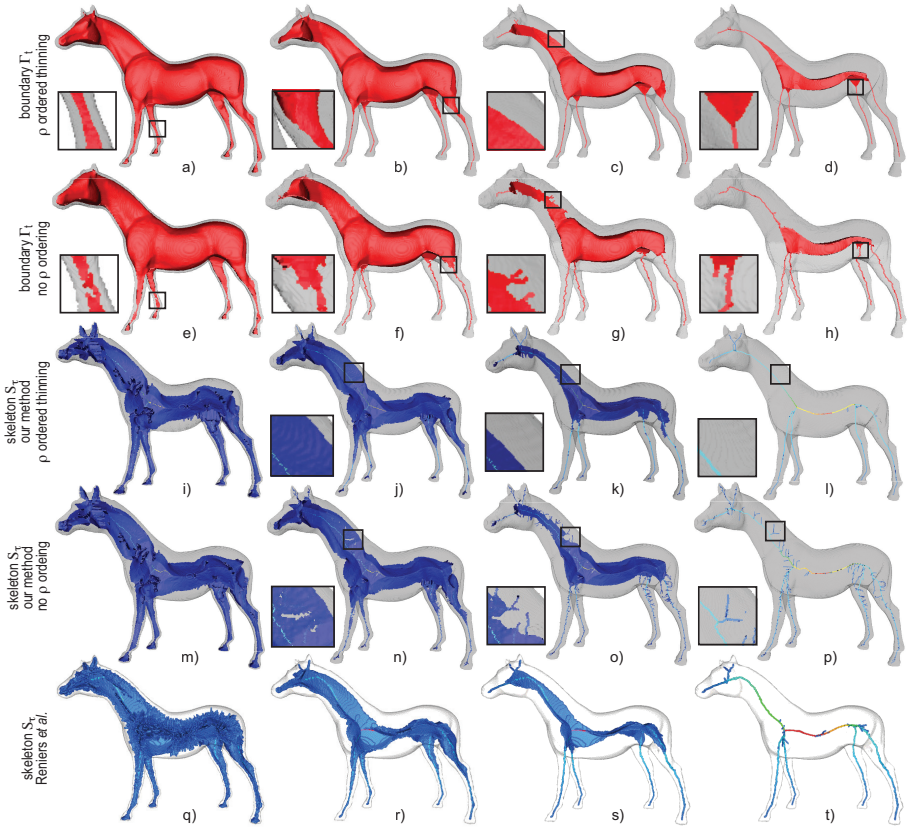


Figure 8.2: Boundary Γ_t at four moments, with (a-d) and without (e-h) density-ordered thinning. Surface/curve skeletons, four simplification levels – our method with (i-l) and without (m-p) density-ordered thinning; Reniers *et al.* [187] (q-t). All skeletons are color-coded by importance λ using a rainbow colormap.

R_3: ensure a smooth evolution of Γ_t, steered by $DT_{\partial\Omega}$ and ρ;

R_4: allow efficient computation.

Most existing thinning algorithms do not provide a representation of Γ_t which satisfies all requirements $R_1..R_3$ above. For example, the divergence-driven thinning algorithm in [204] uses a sorted heap to ensure the correct processing order, thus fails to provide an explicit Γ_t representation. In contrast, we use an explicit representation of Γ_t, modeled as a narrow-band of points (that is, pixels in 2D and voxels in 3D, respectively). Density is transported, according to Eqn. 8.6, only within this narrow-band, which is computationally efficient (R_4).

Let us note that some thinning algorithms combine the detection and removal of a so-called *topologically-simple* point in a single pass. The thinning result may then depend on the point processing order, as discussed in [107]. In contrast, we use an approach similar to [107], where we first find all simple points (detection phase), and eliminate these next (removal phase).

Skeletonize(Shape Ω, Field λ)
Data: Ω: discretized input shape
Result: λ: importance field
$DT_{\partial\Omega}(\cdot) \leftarrow$ 3D Euclidean distance transform of $\partial\Omega$;
$\rho(\cdot) \leftarrow 0$; $\lambda(\cdot) \leftarrow 0$; $M(\cdot) \leftarrow 0$; $Q_1 \leftarrow \varnothing$;
foreach $\mathbf{x} \in \Omega$ **do**
    if $DT_{\partial\Omega}(\mathbf{x}) > 0$ **then** $M(\mathbf{x}) \leftarrow 2$; *// interior points*;
foreach $\mathbf{x} \in \Omega$ **do**
    if $DT_{\partial\Omega}(\mathbf{x}) > 0 \wedge DT_{\partial\Omega}(\mathbf{x}) < 2 \wedge simple(\mathbf{x}, M)$ **then**
        $\rho(\mathbf{x}) \leftarrow 1$; $M(\mathbf{x}) \leftarrow 1$; $Q_1 \leftarrow Q_1 \cup \{\mathbf{x}\}$; *// boundary points*
$d \leftarrow 0$;
repeat
    $d \leftarrow d + \Delta d$; $Q_2 \leftarrow \varnothing$;
    foreach $\mathbf{x} \in Q_1$ **do**
        foreach $\mathbf{y} \in \mathcal{N}(\mathbf{x}) \wedge M(\mathbf{y}) = 2$ **do**
            $Q_2 \leftarrow Q_2 \cup \{\mathbf{y}\}$; $M(\mathbf{y}) \leftarrow 1$; *// new boundary point*
    Sort Q_1 in increasing ρ order;
    $C \leftarrow \varnothing$;
    foreach $\mathbf{x} \in Q_1$ **do** *// process Q_1 in increasing ρ order*
        if $DT_{\partial\Omega}(\mathbf{x}) < d \wedge simple(\mathbf{x}, M)$ **then** $C \leftarrow C \cup \{\mathbf{x}\}$;;
        else $Q_2 \leftarrow Q_2 \cup \{\mathbf{x}\}$; $M(\mathbf{x}) \leftarrow 1$;;
    $B \leftarrow \varnothing$;
    foreach $\mathbf{x} \in C$ **do**
        if $simple(\mathbf{x}, M)$ **then**
            $M(\mathbf{x}) \leftarrow 0$; $B \leftarrow B \cup \{\mathbf{x}\}$;
            $\lambda(\mathbf{x}) \leftarrow max(\lambda(\mathbf{x}), \rho(\mathbf{x}))$;
        else $Q_2 \leftarrow Q_2 \cup \{\mathbf{x}\}$; $M(\mathbf{x}) \leftarrow 1$;;
    Transport ρ from $\mathbf{x} \in B$ to interior points, using Eqn. 8.6;
    foreach $\mathbf{x} \in B$ **do** $\rho(\mathbf{x}) \leftarrow 0$; ;
    $swap(Q_1, Q_2)$;
until $B = \varnothing$;;

Algorithm 1: Skeletonization algorithm.

Our full skeletonization algorithm is now as follows (Alg. 1). During initialization, we compute the Euclidean distance transform $DT_{\partial\Omega}$ on Ω (line 2). Next, we initialize the full-grid fields ρ (density), λ (importance) and M (binary description of the contracting shape) to their default values (line 3). We use $DT_{\partial\Omega}$ to label interior points $\mathbf{x} \in \Omega$ with $M(\mathbf{x}) = 2$ and initial boundary points $\mathbf{x} \in \partial\Omega$ with $M(\mathbf{x}) = 1$ respectively (lines 4-8). Finally, we set the density of boundary points to one and gather them in the set Q_1. This set will keep, during the algorithm execution, all points processed by the current algorithm iteration.

The main loop (lines 10-29) iteratively solves the system of Eqns. 8.6 and 8.7. Here, the field M has two roles. First, M labels points outside ($M = 0$), on the boundary ($M = 1$), and respectively inside ($M = 2$) the shrinking shape, thus efficiently keeps track of this shape. Secondly, we use M to check if a shape point is topologically simple or not: the function $simple(\mathbf{x}, M)$ returns true if removing $\mathbf{x}$ from the shape given by $M(\cdot) > 0$ does not change the shape's topology and false otherwise.

The first inner loop (lines 12-14) fills a set Q_2 with unprocessed, 26-connected (8-connected in 2D) neighbors $\mathbf{y} \in \mathcal{N}(\mathbf{x})$ of the point $\mathbf{x}$ being processed. The set Q_2 captures points going to be processed in the next algorithm iteration (detailed further below). Next, we sort the current set Q_1 on increasing ρ (line 15), allowing the second inner loop (lines 17-19) to process these in increasing order of their density values. This second loop performs the detection phase of the thinning algorithm. Additionally, only points which are topologically-simple and within close distance (Δd) to the current Γ_t are added to set C for further processing. The other non-simple points are added to Q_2 for processing in the next iterations. The third inner loop (lines 21 to 25) performs the removal phase of the thinning algorithm. Topologically-simple points $\mathbf{x} \in C$ are removed (by labeling them with $M(\mathbf{x}) = 0$) and collected in the narrow-band set B. At this stage, their importance λ is also computed (line 24). Non-simple points are added to Q_2.

Set B models the current boundary Γ_t, thus meeting R_1. As shown in Alg. 1, B is built from current topologically-simple points (from Q_1) in increasing ρ order and by filtering them via the distance-threshold criterion (line 18). Hence, the boundary Γ_t is kept relatively smooth (discussed further in Sec. 8.4.3), thus R_3 is met. Once B is available, we can transport density (line 26) from points in B to interior points, thus meeting R_2. After density has been conservatively transported away from the current B, we set the density to zero at points $\mathbf{x} \in B$ (line 27).

At the end of the algorithm's main loop, the sets Q_1 and Q_2 are swapped (line 28). This is an essential aspect of our algorithm, as it facilitates an explicit and computationally-efficient representation of the boundary Γ_t (set B above). Having these sets, we can limit our computations only to a surface-like band of points around the current Γ_t, thus meeting R_4.

The algorithm stops when B becomes empty. For objects of genus 0, this happens when the shrunk shape and Q_1 contain only a single point, which is precisely the shape center C_Ω. This point clearly cannot be added to B since the topological constraint would be violated. For objects of higher genus, termination happens when the shrunk shape contains only curve-skeleton loops connected by non-terminal branches, a structure which cannot be shrunk any longer without disconnecting the skeleton (see example further in Sec. 8.4.4).

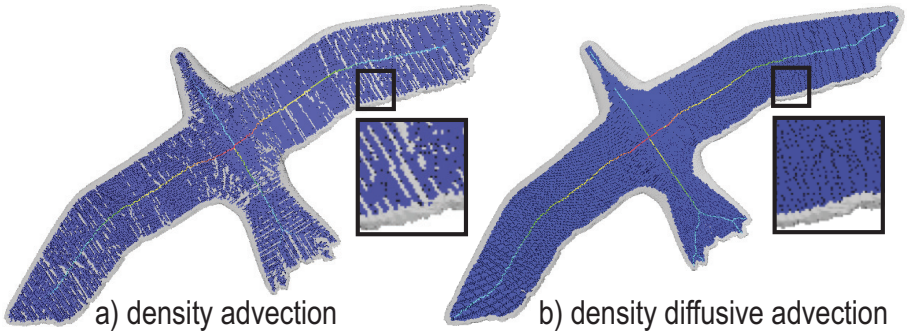


Figure 8.3: Density transport via advection *vs* diffusive advection (see Sec. 8.4.3).

Let us now discuss the smoothness of the boundary Γ_t captured by the set B. Away from singular points of Eqn. 8.7, Γ_t is captured (by the set B) as a level-set of $DT_{\partial\Omega}$, which is a Lipschitz-continuous function under the L_2 metric. At singular points of Eqn. 8.7, our thinning still endorses Lipschitz continuity, but under the L_∞ metric. Intuitively, at such points, the order in which new inner points are processed in lines 12-14 reinforces the L_∞ metric when solving Eqn. 8.6. Note that, although at non-singular points, topologically-constrained level sets [97] provide better smoothness properties of the evolving boundary (due to sub-pixel or sub-voxel precision), such methods have problems regarding the evolution of Γ_t along shocks of Eqn. 8.7, as discussed in Sec. 8.4.1. In contrast, our approach produces a smooth shrinking Γ_t, as shown by Fig. 8.2 a-d.

8.4.3 *Density transport*

We now focus on solving the mass conservation equation (Eqn. 8.6; Alg. 1, line 26). For 2D and 3D curve skeletons, discretizing Eqn. 8.6 with the unconditionally-stable, semi-Lagrangian scheme in [139] suffices. However, generating progressively-simpler *surface* skeletons by simply thresholding the importance field λ requires additional work.

Since our algorithm solves Eqn. 8.7 under the L_∞ norm along its singular points (see Sec. 8.4.2), and since noise, small errors and inacuracies due to the thinning process propagate into the density field evolution (Eqn. 8.6), simply thresholding λ would yield jaggies (indentations) in surface skeletons, for all but trivial shapes (see example in Fig. 8.3 a). To tackle this, we propose a smoothing of the density field ρ, which leads to the desired importance field λ, as follows.

The key idea of [139] for solving conservative-advection PDEs similar to Eqn. 8.3 is to follow so-called *characteristic curves* (along which the PDE becomes an ODE) both forwards and backwards in time, while ensuring that interpolation weights are equal to one for all grid cells, *i.e.*, the advected density is conserved. We constrain the density ρ to be zero outside the shrinking shape, so we only need the forward step. Figure 8.4 left shows a schematic example, assuming that density is transported along surface-skeleton points. For illustration simplicity, and without loss of generality, we next assume that ρ is one at all points in Γ_t. The density propagation directions, given

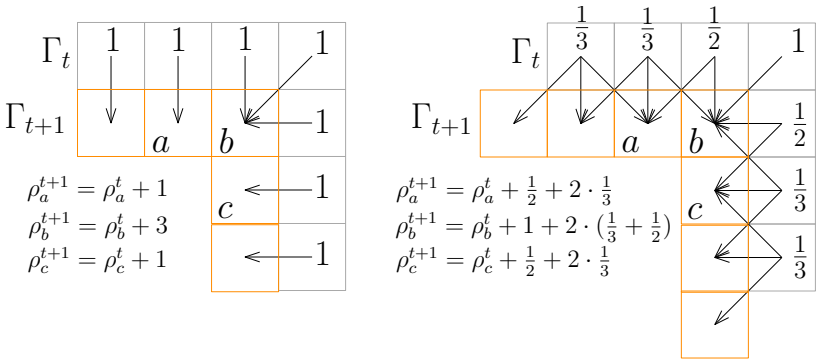


Figure 8.4: Conservative advection *vs.* diffusive advection. Density is transported on the surface skeleton from Γ_t to Γ_{t+1} by *left*: conservative advection and *right*: diffusive advection. Arrows show the directions in which density is transported. The new density values at grid cells a, b and c are also shown.

by $\nabla\phi/\|\nabla\phi\|$ (Eqn. 8.5), are shown by arrows. Hence, as shown in Fig. 8.4, the (linear) interpolation weights equal one, and the new density values at grid cells a, b and c have the indicated values. Since all weights equal one, mass is conserved, as desired.

One way to tackle the above inaccuracies is by endorsing density advection with a (conservative) diffusive component, yielding a smoother evolution of ρ. For this, we propose an anisotropic diffusion process, which we dub *diffusive advection* (as opposed to the well-known diffusion-advection PDE). That is, instead of transporting density solely in the (potentially-noisy) gradient directions, we also allow density to diffuse to other surrounding nearby cells (Fig. 8.4 right). As can be seen by following the arrows, each 'donor cell' now contributes to multiple nearby cells. The weights along each arrow per donor cell, as well as the new density values of cells a, b and c, are also shown. More formally, let χ^t be the *characteristic function* of the shrinking shape, obtained, *e.g.*, by upper-thresholding the field $M(\cdot)$ of Alg. 1 with value 1. Then, the new density value at a grid cell i of Γ_{t+1} for time-step $t+1$ can be expressed as

$$\rho_i^{t+1} = \rho_i^t + \sum_{j \in \mathcal{N}_i} \rho_j^t \frac{1-\chi_j^t}{\sum_{k \in \mathcal{N}_j} \chi_k^t}, \tag{8.12}$$

with being $\mathcal{N}_i$ the 26-connected neighborhood centered at i for the 3D case. Let $w_j^t = \frac{1-\chi_j^t}{\sum_{k \in \mathcal{N}_j} \chi_k^t}$ and assume that cell i receives density contributions from three surrounding cells $j \in \{1,2,3\}$ in Γ_t (see Fig. 8.4), *i.e.*,

$$\rho_i^{t+1} = \rho_i^t + \rho_1^t w_1^t + \rho_2^t w_2^t + \rho_3^t w_3^t. \tag{8.13}$$

This can be rewritten as

$$\rho_i^{t+1} - \rho_i^t = \sum_{k=1}^{3} w_k^t (\rho_k^t - \rho_i^t) + \rho_i^t \sum_{k=1}^{3} w_k^t$$

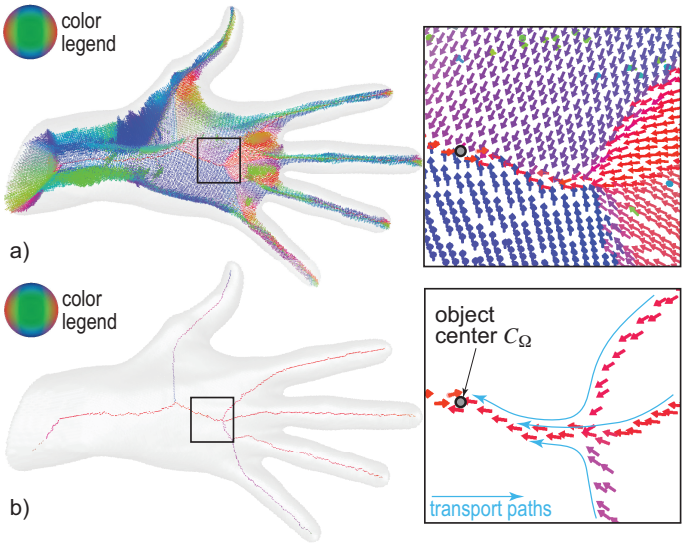


Figure 8.5: Mass transport directions for the *hand* model over the surface skeleton (a) and curve skeleton (b), as computed by our model. See Sec. 8.4.3.

which is a discretization of anisotropic diffusion [174] with an additional reaction term

$$\frac{\partial \rho}{\partial t} = \nabla \cdot (w \nabla \rho) + \rho \mathscr{C}. \tag{8.14}$$

Note that $\mathscr{C}$ above could be seen as a simple curvature estimate. Comparing Eqn. 8.6 with Eqn. 8.14, we see that our simple discrete rule (Eqn. 8.12) replaces conservative advective transport by conservative diffusive transport, while taking into account the geometry of the evolving Γ_t. Finally, to force the density to flow more along gradient directions ($\boldsymbol{u}$, see Eqn. 8.5), we replace the weights w_j^t in Eqn. 8.13 by

$$W_j^t = w_j^t / (1 + l_{ij}^2 / \sigma_a^2), \tag{8.15}$$

with $l_{ij} = \left\| \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|} - \boldsymbol{u} \right\|$ and σ_a a sensitivity parameter. Thus, for small σ_a values, density transport happens mostly along gradient directions, as required by Eqn. 8.6, whereas larger σ_a values result in density diffusion into cells nearby Γ_{t+1}.

Figure 8.5 shows the density transport directions, generated by diffusive advection, for a simple test shape. Directions are shown only for the surface skeleton (a) and the curve skeleton (b), to reduce occlusion and increase visual readability, and are directionally color-coded (red=vectors aligned with the x shape axis, green=vectors aligned with y, blue=vectors aligned with z). As visible from both overview and detail images, the transport directions move density from the input surface along shortest paths to the surface skeleton, next to the curve skeleton, and next to the object center C_Ω, where all mass from $\partial\Omega$ ultimately collapses. The practical result thus matches well the model proposed in Sec. 8.3.2 (see also Fig. 8.1).

Figure 8.3 compares the importance fields λ generated by the two density-transport methods above. Compared to conservative advection (Fig. 8.3 a), diffusive advection creates a smooth density flow along the surface skeleton, leading to the desired smooth and monotonically-increasing importance field (Fig. 8.3 b).

8.4.4 *Detecting different skeleton types*

The importance field λ allows us to easily detect both skeleton types and the shape's center. Specifically, we have that $S_\Omega = \{\mathbf{x} \in \Omega \,|\, \lambda(\mathbf{x}) \geq 2.0\}$, since surface-skeleton points are, by definition, situated at equal distance from at least two different points on the boundary $\partial\Omega$ (Eqn. 8.2), and thus have an importance equal to at least that of two (collapsed) points of $\partial\Omega$, *i.e.* at least two. For genus 0 objects which admit a center in the sense denoted in Sec. 8.3.2, we have $C_\Omega = \arg\max_{\mathbf{x}\in\Omega} \lambda(\mathbf{x})$. Curve-skeleton points could be readily detected by upper thresholding the importance field λ with a large threshold τ. However, there are two shortcomings with this approach: First, the resulting curve skeleton may not be always one-voxel thick. Secondly, its extremities may be removed due to the large threshold value used. In other words, for high τ values, we would obtain a simplified, rather than a full, curve skeleton. To alleviate these issues we detect *salient* curve-skeleton points, during the shrinking process, using

$$CS_\Omega = \{x \in B \,|\, \rho(\mathbf{x}) > c\,\hat{T}(\mathbf{x}) \wedge endPoint(\mathbf{x}, M)\}, \tag{8.16}$$

where $c > 0$ is a constant (explained next); $\hat{T}$ is a simple estimate for the time-of-arrival (approximating T from Eqn. 8.10), given by d in Alg. 1; and $endPoint(\mathbf{x}, M)$ returns true if $\mathbf{x}$ is a curve-skeleton end point. We justify Eqn. 8.16 as follows.

First, Eqn. 8.16 only selects points from the surface skeleton, since only these have a density larger than two. Consider now a point $\mathbf{b}$ on the skeleton boundary ∂S_Ω, such that $\mathbf{b} \in S_\Omega \setminus CS_\Omega$ (Fig. 8.6). The density $\rho(\mathbf{b})$ equals the length of the circular segment $C(\mathbf{b}) \subset \partial\Omega$ (drawn green in Fig. 8.6), which is $\alpha DT_{\partial\Omega}(\mathbf{b}) = \alpha T(\mathbf{b})$ with α its subtended angle. Let $\mathbf{x}$ be a neighbour of $\mathbf{b}$ such that $T(\mathbf{x}) = T(\mathbf{b}) + 1$. Using the boundary evolution equation (Eqn. 8.7) and the arrival-time definition (Eqn. 8.10), it can be easily shown that $\mathbf{x}$ must be in the 'upstream' direction from $\mathbf{b}$, since $\mathbf{x} - \mathbf{b}$ and $\nabla\phi$ are parallel vectors. When the evolving boundary passes through $\mathbf{b}$ (*i.e.*, $\mathbf{b}$ is removed by density-ordered thinning, see Alg. 1), $\rho(\mathbf{b})$ is 'pushed' in the upstream direction through (diffusive) advection transport. Thus, $\mathbf{x}$ will directly receive most density of $\mathbf{b}$, under advective density transport. Moreover, due to the boundary-propagation order, when the interface is about to pass through $\mathbf{x}$, point $\mathbf{x}$ must also be found to be part of the surface skeleton, and furthermore, $\rho(\mathbf{x}) > \rho(\mathbf{b})$. Indeed, since regular points $\mathbf{x} \in S_\Omega \setminus CS_\Omega$ also receive density contributions from its two (or more) feature-points on $\partial\Omega$ (boundary normals are preserved by the Eikonal equation), a rough estimate for the *minimum* density at $\mathbf{x}$ is $\rho(\mathbf{x}) > 2 + \alpha T(\mathbf{b})$. By a similar reasoning, for points $\mathbf{y} \in S_\Omega \setminus CS_\Omega$ situated in upstream direction from $\mathbf{b}$, with $T(\mathbf{y}) > T(\mathbf{b}) + 1$, we find $\rho(\mathbf{y}) > 2T(\mathbf{y}) + \alpha T(\mathbf{b})$ as a lower bound on their density. Finally, points $\mathbf{c} \in CS_\Omega$ collect density from at least two neighbor-points $\mathbf{y} \in S_\Omega \setminus CS_\Omega$; this is so since the curve CS_Ω locally divides the 2D-manifold surface S_Ω in two parts. For instance, in Fig. 8.6, $\mathbf{c}$ will receive density from at least two surface-skeleton neighbor points situ-

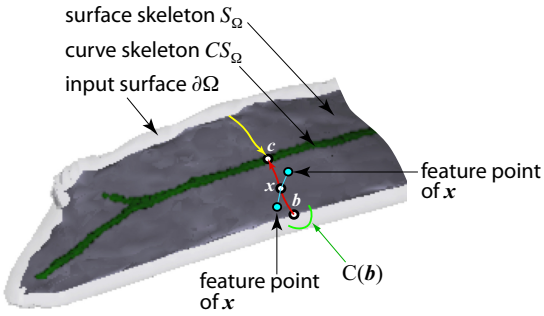


Figure 8.6: Selecting curve-skeleton points for importance boosting (Sec. 8.4.4). The red arrow shows the upstream density-advection from **b** through **x** up to the curve-skeleton point **c**. The yellow arrow shows the second direction from which **c** receives density from the surface-skeleton.

ated at the tips of the red, respective yellow, arrows. Additionally, **c** receives a (larger) density contribution from (at least) another downstream curve-skeleton neighbor, so $\rho(\mathbf{c}) > 4\sum_i T(\mathbf{y}_i) \geq 4T(\mathbf{c})$. Hence, setting $c = 4$ in Eqn. 8.16 performs conservative curve-skeleton detection. We verified empirically that setting $c = 4$ cleanly and robustly separates important curve-skeleton points from surface-skeleton points. Additionally, the end-point test in Eqn. 8.16 ensures that only points which are at the sharp 'tips' of the shrinking surface Γ_t are considered as salient curve-skeleton candidates.

Once the points $\mathbf{x} \in CS_\Omega$ are found using Eqn. 8.16, we 'boost' their importance during the iterative process by adding a constant fraction δ of the input shape's mass $|\partial\Omega|$. As a result, curve-skeleton points will have a significantly higher importance than surface-skeleton points. Also, as δ is constant for all $\mathbf{x} \in CS_\Omega$, the monotonic increase of importance along the curve skeleton branches is preserved. All in all, this allows us to threshold the final importance λ at precisely δ to cleanly and easily detect a full (unsimplified) CS_Ω, and to threshold λ at higher values to obtain progressively simplified curve skeletons.

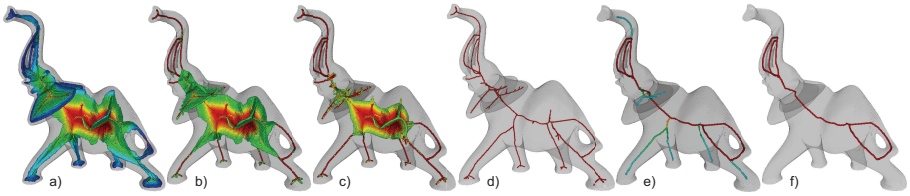


Figure 8.7: Progressively simplified skeletons: surface skeletons (a-c) and curve skeletons (d-f) for six increasing τ values. See Sec. 8.4.4.

Figure 8.7 shows the effect of increasing the importance threshold τ for a shape of genus 3. Skeleton voxels are colored by importance λ via a rainbow colormap. The first three τ values (Figs. 8.7 a-c) yield three increasingly simple surface skeletons. The last of these (Fig. 8.7 c) is a mix of surface and curve skeleton parts, *i.e.*, is an example of the meso skeletons described in [221]. As τ increases to δ, we find the

full curve skeleton (Fig. 8.7 d). Increasing τ further removes the end branches of the curve skeleton, without disconnecting its loops (Fig. 8.7 e). For visual clarity, we used in Fig. 8.7 e a local normalization of the colormap, based on the range of voxels in the respective simplified skeleton, rather than on the full importance range $[0, \lambda_{max}]$ used for all other figures. This shows better how the importance increases from the curve skeleton endpoints to its center, and is high over its loops. Finally, for the highest considered τ value, we get the fully simplified curve-skeleton without disconnections containing only loops connected by internal branches (Fig. 8.7 f).

8.4.5 *Implementation details and parameter settings*

Implementation: Our algorithm is implemented in C++ using OpenGL for skeleton rendering. We compute $DT_{\partial\Omega}$ on the CPU by the method of Meijster *et al.* [102], or on the GPU (if available) by the method of Cao *et al.* [41]. Both methods compute the exact Euclidean distance transform and are linear in $|\Omega|$, the number of foreground pixels or voxels in the input shape. We implement $simple()$ using the Euler number for $\Omega \subset \mathbb{R}^2$ and Malandain's criterion [28] for $\Omega \subset \mathbb{R}^3$ respectively. We detect curve-skeleton endpoints (function $endPoint()$, Eqn. 8.16) as those voxels $\mathbf{x} \in B$ with one (26-connected) neighbor $\mathbf{y}$ for which $M(\mathbf{y}) > 0$. We use the integral method of Neumann *et al.* [164] to estimate the gradient directions along which density is transported (Eqn. 8.5). Sorting Q_1 is implemented by the Standard Template Library (STL) *sort* function. Algorithm 1 requires K iterations of the loop in line 10, where K is roughly equal to $\frac{1}{2}\max_{\mathbf{x}\in\Omega} DT_{\partial\Omega}(\mathbf{x})$, the maximal thickness of Ω. Since at each iteration we sort the set Q_1, which is worst-case equal to the input boundary $\partial\Omega$, the total complexity of our method is $O(K\,|\partial\Omega|\,\log(|\partial\Omega|))$.

Parameters: Throughout the chapter we use the following parameter settings. To obtain a good approximation of the motion equation (Eqn. 8.7), we set the distance step Δd in Alg. 1 to $\Delta d = 1$. The parameter σ_a, controlling the density spread along gradient directions (Eqn. 8.15), is set to $\sigma_a = 0.2$. The parameter c in Eqn. 8.16, *i.e.*, the saliency of the detected curve-skeleton points, is set to $c = 4.0$. The parameter δ, representing the importance difference between curve and surface skeleton points (Sec. 8.4.4), is set to $\delta = 0.1$ of the total input surface mass $|\partial\Omega|$. The above values have been tested on a set of over 60 shapes, voxelized at various resolutions, and have consistently delivered good results like the ones shown in our figures here. As such, the only free parameter is the skeleton simplification threshold τ, whose use is explained in Sec. 8.4.4.

8.5 COMPARATIVE RESULTS

8.5.1 *Two-dimensional skeletons*

For 2D shapes, we compared our method with the Augmented Fast Marching Method (AFMM) [228]. AFMM is a good example of 2D skeletonization methods that computes centered, accurate, connected, and pixel-thin skeletons regularized by the collapsed boundary-length importance metric (like [79, 168]).

Figure 8.8 shows skeletons of several shapes from the database in [195] extracted with AFMM and with our method, for several simplification thresholds τ. Our method

Table 8: Performance comparison. Skeletonization speeds in columns 4-13 are given in foreground voxels/millsecond.

Model (512^3 **voxels**)	$\|\Omega\|$	$\|\partial\Omega\|$	**Our method**	**TV**	**DDS**	**RT**	**ITP(curve)**	**MS(curve)**	**MS(surface)**	**HJ**	**IMA**	**ITP(surface)**	**MBS(surface)**
Dino	3952385	214232	364.8 (10.83 sec)	109.5	9.9	14.7	60.9	30.6	38.32	36.9	62.5	66.5	1.62 sec
Dragon	1208845	119144	421.4 (2.86 sec)	61.2	10.6	10.8	25.7	7.2	27.2	14.8	45.8	25.7	14.0 sec
Fertility	7010557	343667	365.5 (19.18 sec)	133.0	13.2	16.8	51.9	15.4	25.6	30.9	158.0	51.9	4.13 sec
Rockerarm	2726188	186362	500.1 (5.45 sec)	111.4	14.3	22.3	51.9	17.9	13.5	52.3	99.4	49.3	4.2 sec
Casting	4280458	544345	400.6 (10.68 sec)	109.8	12.6	19.7	48.6	9.3	29.1	49.7	87.5	53.2	3.28 sec
Horse	9235212	329222	361.9 (25.51 sec)	120.5	10.2	25.9	54.9	13.7	32.9	52.1	94.3	54.9	13.65 sec
Elephant	7885332	310919	294.9 (26.73 sec)	112.8	10.6	9.1	43.8	20.8	30.4	42.3	109.7	42.2	3.92 sec
Frog	7796250	323397	335.5 (23.23 sec)	123.0	12.9	16.0	48.1	10.5	63.3	54.1	148.5	65.6	2.74 sec

and the AFMM produce visually identical skeletons, both in terms of position, but also branches kept at a given τ. This is a non-trivial result, given that our method and the AFMM have completely different models behind. Moreover, since λ_{AFMM} at a skeleton point $\mathbf{x}$ equals the length of boundary that collapses to $\mathbf{x}$ when advected in $\nabla DT_{\partial\Omega}$, and since $\lambda \approx \lambda_{AFMM}$ (Fig. 8.8), this supports the claim that our λ is indeed equal to the collapsed boundary length.

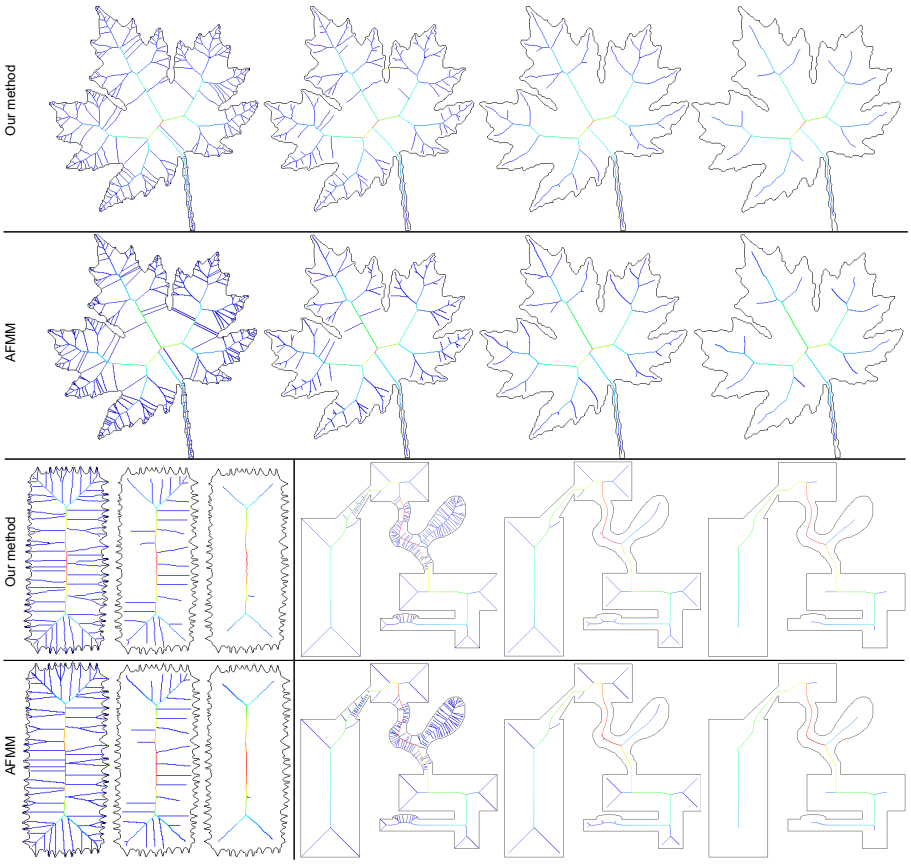


Figure 8.8: Comparison of importance-colored 2D skeletons computed with our method and the AFMM [228], for increasing skeleton-simplification levels.

8.5.2 *Surface skeletons*

Figure 8.9 (bottom 6 rows) compares our method with four voxel-based methods: multiscale skeletons (MS) [187], Hamilton-Jacobi (HJ) [204], integer medial axis (IMA) [102], and iterative thinning process (ITP) [117]; and with the multiscale mesh-based skeletonization (MBS) in [109]. Test shapes cover a wide range, including natural and synthetic, smooth and detailed, and objects of various genii (all voxelized at 512^3 resolution by *binvox* [171]). Our surface skeletons look very similar to those

created by other methods, and show similar power in capturing the input shape genus and boundary details.

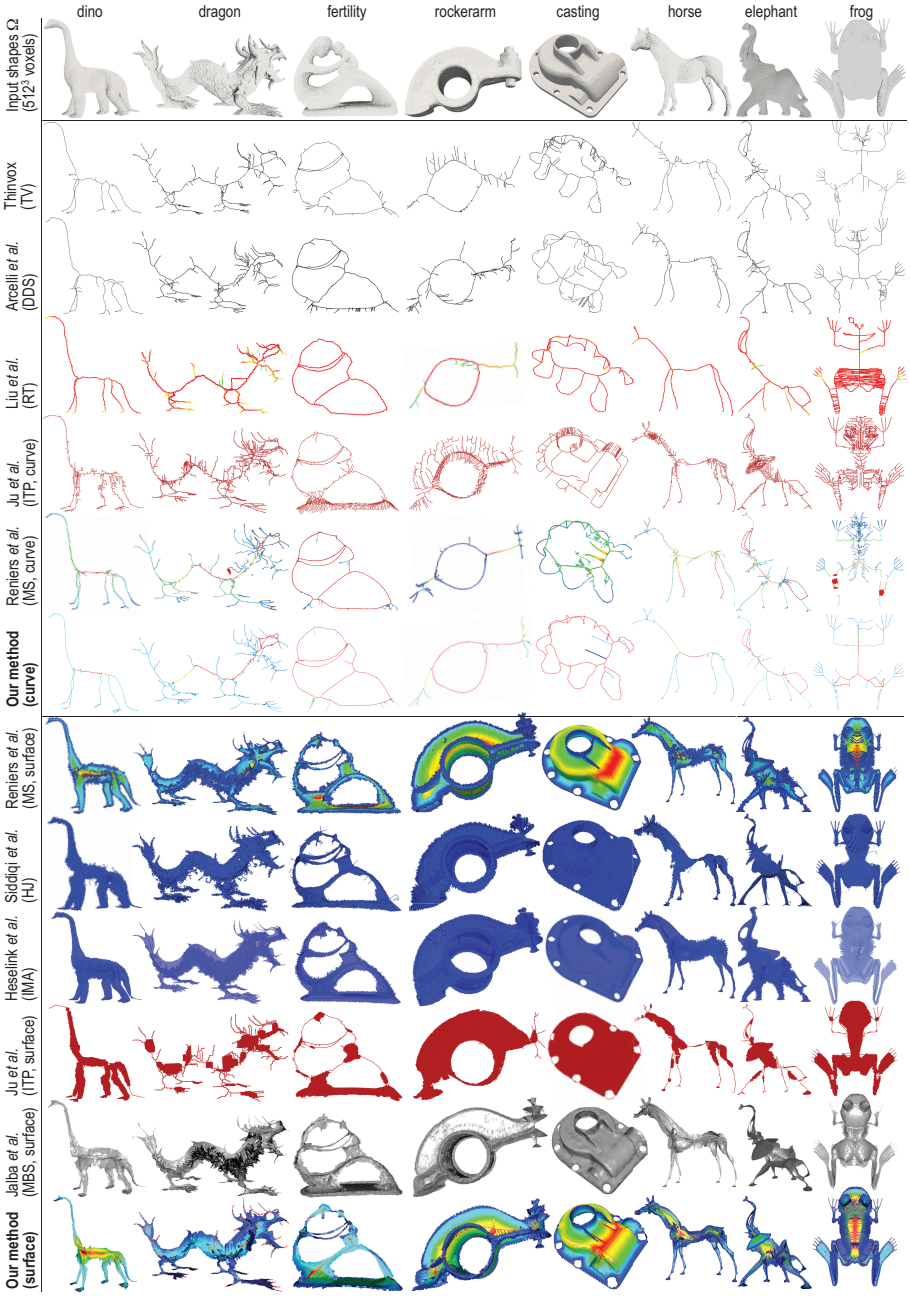


Figure 8.9: Comparison of our 3D surface and curve skeletons with 10 related methods. Top 6 rows: curve skeletons. Bottom 6 rows: surface skeletons.

The comparison with MS is particularly interesting. To our knowledge, MS is the only existing voxel-based technique that computes multiscale surface skeletons whose importance uses a boundary-collapse metric. For MS, this metric is

$$\lambda_{SS}(\mathbf{x} \in S) = \min_{\gamma=(\mathbf{f}_1^{\mathbf{x}} \rightsquigarrow \mathbf{f}_2^{\mathbf{x}}) \subset \partial\Omega} \|\gamma\| \tag{8.17}$$

i.e. the length $\|\gamma\|$ of the shortest geodesic path γ on $\partial\Omega$ between the two feature points $\mathbf{f}_1^{\mathbf{x}}$ and $\mathbf{f}_2^{\mathbf{x}}$ of a skeleton point $\mathbf{x}$.

As for the 2D case (Sec. 8.5.1), we see that our surface skeletons *and* importance values (color-coded in Fig. 8.9, row 12, by a rainbow colormap) are very similar to the MS ones (Fig. 8.9, row 7). Figure 8.2, two bottom rows, details this insight by showing four surface skeletons obtained by thresholding our importance λ, and λ_{SS}, at four increasing values. The monotonic increase of both λ and λ_{SS}, from low values on the surface-skeleton boundary to high values on the curve skeleton, and the resulting skeletons, are similar. This is an even more interesting result than the similarity of our results with the AFMM. Our importance $\lambda(\mathbf{x})$ equals the amount of boundary mass which reaches $\mathbf{x}$ subject to Eqs. 8.6,8.7. The fact that $\lambda \approx \lambda_{SS}$ supports the conjectures in [70, 187] that all boundary points on such a geodesic γ, if advected in $\nabla DT_{\partial\Omega}$, would reach the skeleton point $\mathbf{x}$. However, a formal proof of these conjectures still lacks. Separately, Fig. 8.2 (bottom row) shows that our λ monotonically increases as we advance inwards on the skeletal structures (Sec. 8.4), hence that thresholding λ yields connected skeletons.

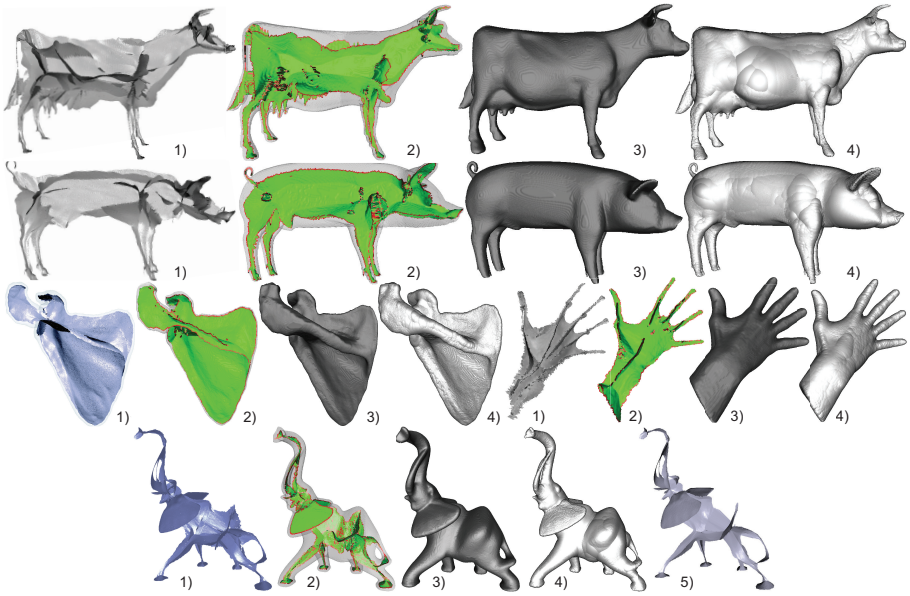


Figure 8.10: Comparison of surface skeletons: (1) Jalba *et al.* (MBS) [109]; (2) our method; Original shapes (3) *vs* our skeleton-based reconstruction (4). All voxel models have a 512^3 resolution. Last row: detailed comparison, including also (5) the surface-skeleton of Miklos *et al.* [159].

To better qualitatively assess our surface skeletons, Fig. 8.10 compares these with skeletons computed by the high-resolution MBS method [109], which uses the multiscale-importance in [187], but has a different skeleton detector, and uses a mesh rather than a voxel representation. For more insight, we colored our surface-skeleton border voxels red. Our skeletons are very similar to the MBS ones. Our method captures roughly the same amount of skeleton detail as MBS, even though the latter uses mesh, rather than voxel, skeleton-and-shape representations. Figures 8.10 (4) shows our reconstruction of the input shape by drawing balls centered at the surface-skeleton voxels $\mathbf{x}$ and whose radii equal $DT_{\partial\Omega}(\mathbf{x})$, using the rendering technique in [109]. As visible, our reconstructions are very close to the input shapes (Figs. 8.10 (3)). The small bubble-like differences are explained by the fixed resolution of the voxel grid. This verifies the reconstructibility criterion for our method and, implicitly, shows that our skeletons are correctly centered. The last row in Fig. 8.10 compares our method for the *elephant* shape from Fig. 8.7, which has three tunnels, large thin-and-flat areas (ears), near-cylindrical parts (legs), high positive-curvature areas (ear borders), and high negative-curvature areas (ear-head junctions). Our surface skeleton is very close to the one produced by MBS, and also to the skeleton produced by the discrete-scale axis (DSA) mesh-based method of Miklos *et al.* [159], one of the highest-accuracy existing surface-skeletonization methods.

8.5.3 *Curve skeletons*

Figure 8.9 (top 6 rows) compares our method with five curve-skeleton methods: Thinvox (TV) [171], distance-driven skeletonization (DDS) [11], robust thinning (RT) [149], iterative thinning process (ITP) [117], and multiscale skeletons (MS) [187]. In contrast to surface skeletons, we see now more variation between the compared methods. Our method delivers consistently thin (unlike MS), smooth (unlike ITP), noise-free (unlike ITP), and genus-preserving (unlike RT and MS) curve skeletons. Figure 8.11 shows extra insight, by comparing our method with six additional mesh-based curve-skeletonization methods (Kustra *et al.* [133]; Livesu *et al.* [150]; Telea and Jalba [227]; Au *et al.* [15]; Dey and Sun [70]; and Jalba *et al.* [109]). As visible, our method yields well-centered curve skeletons which compare favorably, in terms of smoothness and lack of spurious branches, with the highest-quality mesh-based skeletons.

As for surface skeletons (Sec. 8.5.2), let us detail the parallel with MS curve-skeletons. MS detects curve skeletons as those points having at least two equal-length geodesics on $\partial\Omega$ between their feature points (MGF criterion in [70]). MS extends MGF by assigning a curve-skeleton importance λ_{CS} equal to the *area* bounded on $\partial\Omega$ by the above two geodesics. Figures 8.9 and 8.11 show that our curve-skeletons and MS are very similar. The importances λ and λ_{CS} are also quite similar (see Fig. 8.2 l *vs* Fig. 8.2 t and Fig. 8.9, row 5 *vs* 6), except for the *rockerarm*, *casting*, and *frog* models, where λ_{CS} is smaller. Upon closer inspection, this shows a defect of MS: Low λ_{CS} points appear on curve-skeleton loops, whose geodesics do *not* cut $\partial\Omega$ in two separate parts according to the Jordan theorem [231], so following [187] these points should get a high importance, to prevent loop disconnection when simplifying skeletons. Our method correctly finds such loops and assigns them a high importance.

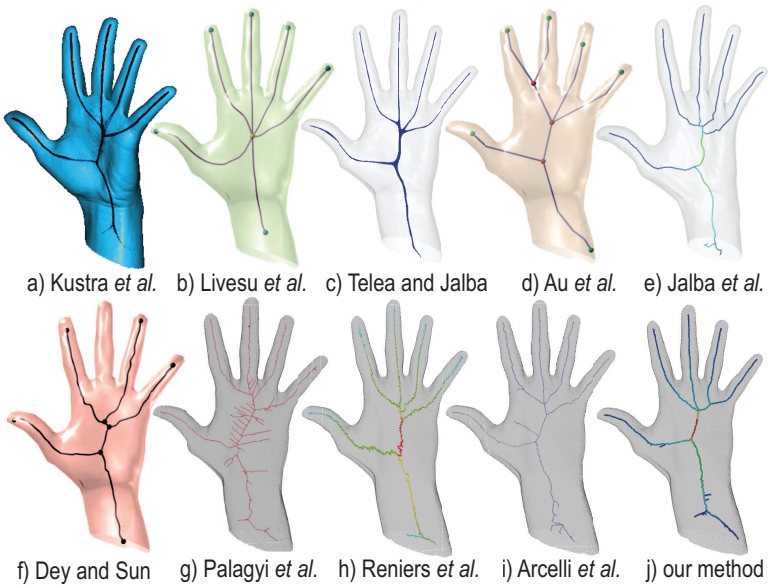


Figure 8.11: Comparison of curve-skeletonization methods. Mesh-based methods (a-e). Voxel-based methods (f-i). Our method (j).

8.6 DISCUSSION

8.6.1 *Method properties*

Unification: Our method extracts 2D skeletons, 3D surface and 3D curve multiscale skeletons. To our knowledge, this is the first time that all these three skeleton types, including multiscale regularization, are computed by a single method which uses a single simplification metric. From both a theoretical and a practical perspective, we believe this to be an important result.

Robustness and accuracy: Our method computes thin, centered, homotopy-preserving, and connected skeletons from potentially noisy 2D and 3D shapes of arbitrary genus. Figure 8.12 quantitatively compares our skeletons with four other methods, using the technique in [210]. In detail, given two (curve or surface) skeletons S_1 and S_2, we first define the distance field

$$D_{12}(\mathbf{x} \in \Omega) = \begin{cases} \min_{\mathbf{y} \in S_2} \|\mathbf{x} - \mathbf{y}\| = DT_{S_2}(\mathbf{x}) & \text{if } \mathbf{x} \in S_1, \\ 0 & \text{if } \mathbf{x} \notin S_1. \end{cases}$$

To compare two same-kind (curve or surface) skeletons, we draw the field $D_{12} + D_{21}$ over the voxels $S_1 \cup S_2$, normalized by its maximum value, using a rainbow colormap. Close skeleton fragments are blue, while outlier ones are red. Comparing a curve skeleton CS_1 with a surface skeleton SS_2 shows how well is CS_1 *contained* within SS_2. For this, we color voxels in CS_1 by D_{12}, and voxels in $SS_2 \setminus CS_1$ with gray. Hence, vox-

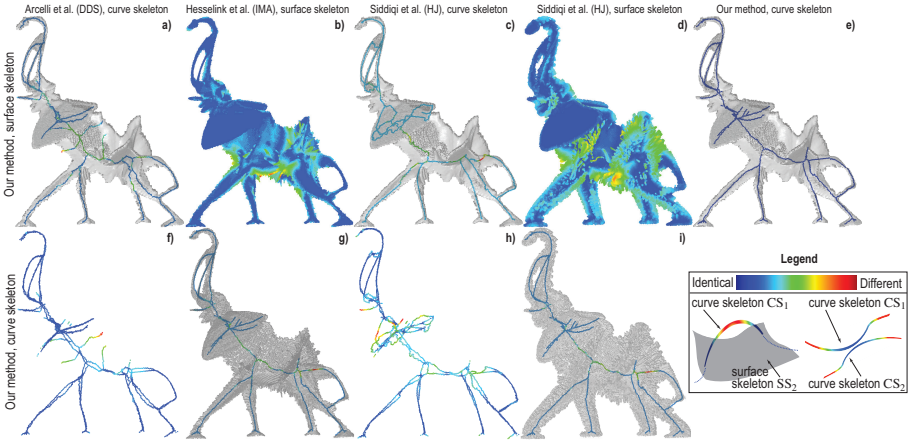


Figure 8.12: Quantitative comparison of surface and curve skeletons. Color mapping indicates skeleton differences (see Sec. 8.6.1).

els in $CS_1 \cap SS_2$ are blue, and voxels in CS_1 far from SS_2 become red (Fig. 8.12 inset). Looking at Fig. 8.12, we see that our surface skeletons are very similar to those produced by IMA and HJ (Fig. 8.12 b,d). Warm colors, showing differences, occur mainly on the surface-skeleton boundary, and are due to the different simplification methods (and simplification levels) used by the compared methods. Our curve skeletons are fully contained in our surface skeletons, as expected (Fig. 8.12 e), but also nearly fully contained in the IMA and HJ skeletons (Fig. 8.12 g,i). Conversely, the DDS and HJ curve skeletons are well contained in our surface skeletons (Fig. 8.12 a,c). The largest differences, found between curve skeletons themselves (Fig. 8.12 f,h), are still quite small in absolute value.

Scalability: Table 8 (column 4) shows the speed of our method, implemented in C++ on a Linux 3.5 GHz PC with 32 GB RAM and an NVidia 690 GTX for the shapes in Fig. 8.9. Columns 2 and 3 give the shapes' surface areas $|\partial\Omega|$ and volumes $|\Omega|$, in voxels. We note a high and relatively shape-independent throughput (foreground voxels/second), in line with the complexity stated in Sec. 8.4.5. Compared to the tested voxel-based methods (columns 5-13), our method is one order of magnitude faster on average. The next-fastest method is TV [171]. However, TV does not compute surface skeletons nor an importance metric. Compared to MS, the only other voxel-based method which computes a multiscale importance metric, we are on average 10 times faster. The last column in Tab. 8 shows the speed of the MBS mesh-based skeletonization in [109], the second other method we are aware of (apart from MS) which computes multiscale skeletons. Compared to our absolute timings (Tab. 8, column 4, figures in brackets), MBS is 2.6 times faster on average. However, MBS is parallelized on the GPU, while our method is sequential and on the CPU. Our cost is essentially dominated by the number of foreground voxels (Sec. 8.4.5), while MBS is dominated by the cost of its geodesic tracing, which is $O(n^{3/2})$ for a meshed surface of n vertices. As such, our method is of comparable speed or even faster than MBS for shapes like *dragon* or *rockerarm*, which have a low volume (thus, generate few

voxels for our method) but are represented by highly-refined meshes (thus, are costly for MBS). Conversely, our method is about ten times slower than MBS for shapes having a high volume, and whose mesh-representation uses few vertices, *e.g. dino* and *frog*.

Simplicity: Our framework has a single, simple, 2D and 3D implementation (under 2000 lines of C++), with no complex computational geometry operations or degenerate cases, unlike some mesh-based methods [159, 187, 215]. Its only user parameter, the importance threshold τ, is simple to use: given the initial uniform density on $\partial\Omega$, τ selects skeletal structures which encode input-shape details whose length (in 2D) or area (in 3D) is larger than τ, similarly to [79, 168, 187, 228].

Classical properties: We next summarize the behavior of our method *vs* several recognized desirable skeleton properties.

1. Centeredness: Centeredness is ensured by the unit-speed evolution of Γ_t (Eqn. 8.5). We verify centeredness, both for curve and surface skeletons, against several methods known to formally respect this property [11, 102, 187, 228] (see Fig. 8.10).

2. Thinness: Our 2D and 3D skeletons are one-cell (pixel or voxel) thin, by construction. To argue this, suppose that this would not be so. Then, a skeleton λ_τ would (a) be thicker than one cell, and (b) would have, over a cross-section, the same importance λ. If such a thick cross-section existed, our shrinking algorithm would continue, since the shape can be further shrunk without disconnecting it (see Fig. 8.2).

3. Homotopy: Homotopy of the skeleton with the input shape is guaranteed by construction, by the constraint in Eqn. 8.4 and its corresponding implementation (Alg. 1).

4. Reconstructibility: The ability to reconstruct (smoothed versions of) the input shape from (simplified versions of) its skeleton is shown in Fig. 8.10. As visible, our reconstruction is quite accurate (compare *e.g.* Fig. 8.10 with Fig. 4 in [109]), modulo the natural limitation imposed by the fixed voxel grid.

5. Rotational invariance: The discretization of our proposed PDE system (Eqns. 8.3-8.5) described in Sec. 8.4.3 is rotationally invariant by construction. We have verified that we indeed obtain nearly voxel-identical multiscale skeletons for the same input shape rotated at random angles with respect to the cell grid (not shown here for sake of brevity).

6. Curve vs *surface skeletons:* Our curve skeletons are by construction included in the surface skeletons of the same shape, since they are both obtained by thresholding the same single-and-global importance field λ (see Sec. 8.4.4).

7. Multiscale and noise resistance: The field λ describes the whole space between the input surface $\partial\Omega$, surface skeleton S_Ω, curve skeleton CS_Ω, and shape center C_Ω. Thresholding λ with increasing values yields the S_Ω from Ω; simplified surface skeletons (without branches due to small-scale shape details); the (simplified) CS_Ω; and the shape center, or zero-dimensional skeleton, of Ω. Reniers *et al.* get similar results, but they need two *separate* surface and curve skeleton importances and corresponding algorithms [187]. We compute λ making no distinction between the two skeleton types. Telea and Jalba compute multiscale curve skeletons by collapsing surface skeletons inwards, following the idea that the former can be seen as the medial

loci of the latter [227]. Yet, as in [187], their surface and curve skeleton algorithms are fundamentally different, and they also do not propose a curve-skeleton importance metric. Thresholding λ at values τ between the average importance of S_Ω and CS_Ω yields a meso-skeleton structure [221] that continuously shrinks from S_Ω towards CS_Ω as τ increases (see Sec. 8.4.4).

Limitations: Our method stores four voxel scalar volumes (ρ, λ, M, and $DT_{\partial\Omega}$ in Alg. 1), *i.e.* can handle shapes up to roughly 1000^3 voxels on a 16 GB PC. Mesh-based skeletonization methods [109, 151] need far less memory. For instance, the mesh models for all shapes in this chapter, which are up to 1 M triangles, need only 24 MB with the method in [109]. Separately, we acknowledge that our comparisons highlight differences between our skeletons and those produced by other methods, but do not explicitly show which skeletons are more suitable for a specific application, *e.g.* shape retrieval, classification, or segmentation. A thorough qualitative comparison with this goal is an important topic for future work.

8.6.2 *Comparison with Hamiltonian methods*

Equation 8.3 is similar with Torsello and Hancock's (TH) mass conservation model $\nabla \cdot (\rho \mathbf{u}) = 0$ used for 2D skeletonization [14]. Yet, several key differences exist. Numerically, TH transforms the mass conservation $\nabla \cdot (\rho \mathbf{u}) = 0$ into a system of two ODEs (Eqns. 7 in [14]) by the substitution $\sigma = \log(\rho)$. These ODEs are solved with a second-order Crank-Nicholson divergence-discretization scheme coupled with semi-Lagrangian advection. In contrast, we use the *conservative* semi-Lagrangian scheme of Fedkiw [139], which only needs linear interpolation and clamping. This has several advantages. First, our scheme is numerically very stable, and conserves density well. Secondly, we do not need the second-order divergence discretization of TH. Thirdly, by computing the density logarithm σ, TH also needs exponentiation to find the final density $\rho = \exp(\sigma)$. We noticed, in practice, that this creates important numerical problems, such as an infinite value of ρ at several points in Ω.

Torsello and Rossi extend the TH density advection to extract 3D medial surfaces [188]. In their model, density advection stops when reaching the surface skeleton. In contrast, we continue advection by collapsing the surface skeleton to the curve skeleton and the latter to the shape center, yielding all desired skeletal representations within a single process.

8.7 CONCLUSIONS

We have presented a unified framework for computing 2D skeletons and 3D surface and curve skeletons. We detect all skeleton types by a single algorithm, and also compute a single importance metric which assigns to each skeletal point the amount of (2D or 3D) input boundary described by that point. Comparing our skeletons and their computed importance with results computed by related methods shows a very good match. We present a simple implementation of our method which achieves good performance results on a range of complex 2D and 3D shapes, *i.e.*, over 3 times faster than the fastest voxel-based skeletonization method we are aware of, and over 10 times faster than comparable multiscale methods.

Future work can target several directions. Porting our method to massively-parallel platforms (*e.g.*, CUDA) using sparse voxel grids will increase scalability. Separately, modulating the input-surface density by *e.g.* curvature or application-specific metrics would allow different feature-sensitive skeleton simplifications.

The unified multiscale skeletonization framework proposed here is, obviously, of added value in any application context where one requires a 2D planar, 3D surface, or 3D curve skeletonization method that complies with the quality criteria outlined in Sec. 8.6.1 and is also simple to implement. In our specific context of shape restoration, this framework provides us with the ideal tool to be able to, finally, extend our 2D gap-removal techniques introduced in Chapter 4 to 3D. This extension is the subject of the next chapter.

This chapter is based on:

A. C. Jalba, A. Sobiecki, and A.C. Telea. An Unified Multiscale Framework for Planar, Surface, and Curve Skeletonization. *IEEE TPAMI, 38(1):30–45,*, JAN, 2016.

9 GAP DETECTION AND REMOVAL FOR 3D SHAPES

Chapters 4 and 5 have described a set of methods for the detection and elimination of gaps present in two-dimensional images, such as 2D binary shapes and grayscale and color images. As discussed there, such methods are efficient and effective for a number of image processing operations such as the digital removal of hairs from dermoscopy images prior to further image analysis. An important technical component of these methods are the corresponding 2D skeletons, or medial axes, extracted from shapes present in the images. In this chapter, we show how the equivalent problem of detection and removal of gaps can be formulated and solved in the context of 3D shapes.

9.1 INTRODUCTION

Our method is an extension of our 2D gap-filling on binary shapes proposed in chapter 4 at 2D gray level images and 3D shapes. For 2D gray level images there is a combination of morphological operators filters, statistical threshold and an classical inpainting filter [229]. For 3D shapes we use two kinds of skeletons, curve and surface skeletons, curve skeletons are used to identify such gap and surface skeleton is used to fill such gap, then in the gap region, we match curve and surface skeletons. In the follow subsections, we are going to explain each application separately.

To start with, let us define the approached problem. In the 2D gap detection-and-removal context, we made the following assumptions:

1. Shapes are represented as densely and uniformly sampled 2D datasets, *i.e.*, pixel images;
2. The above images can be either binary or grayscale. Color images are reduced to grayscale image by considering their luminance component;
3. Gaps to be treated are described by thin-and-elongated structures that penetrate deeply into the shape from its boundary. Given the dimensionality of our shapes (2D), such gaps are, thus, one-dimensional structures, such as cracks in binary images (Chapter 4) or hairs in grayscale images (Chapter 5);
4. During the gap detection-and-removal, shape details which are not part of such gaps, should be altered as little as possible.

By analogy with the above, we make the following assumptions for our 3D gap detection-and-removal work presented in this chapter:

1. Shapes are represented as densely and uniformly sampled 3D datasets, *i.e.*, voxel volumes;
2. The above volumes can be either binary or grayscale. We do not treat here the case of color volumes, since we do not avail of such data in the context of our

applications to be discussed next. However, if such volumes were available, we could reduce them to grayscale volume by following a similar procedure to the one used for color images;

3. Gaps to be treated are described by thin-and-elongated structures that penetrate deeply into the shape from its boundary (like in the 2D case). Given the dimensionality of our shapes (3D), such gaps can be *either* one-dimensional structures, such as wires present in a 3D scan (Section 9.2) or a mix of one-dimensional and two-dimensional structures, such as the cuts and cracks present in 3D binary volumes (Section 9.3);

4. During the gap detection-and-removal, shape details which are not part of such gaps, should be altered as little as possible.

As discussed in Section 2.1.4, there are far fewer methods for detection and removal of gaps in 3D volumes as compared to the range of methods existing for 2D images. Arguably the best known such class of methods use morphological filters such as closing to both find and remove (close) gaps whose size is under a user-prescribed threshold [197]. While very simple to implement and fast, such methods have to be set up with great care so that they do not remove details from the image which should be kept (false positives), or, conversely, leave defects in the image which should be removed (false negatives). More advanced 3D shape restoration methods exist, such as [21, 22, 122, 244]. However, most such methods treat the detection-and-removal of *surface* defects, rather than *volumetric* defects; or have several limitations, such as removing both erroneous and important shape details, or requiring non-trivial effort from the end user in the form of manual delineation or parameter setting.

The structure of this chapter is as follows. In Section 9.2 we discuss the case of removing one-dimensional (curve-like) defects from 3D grayscale volumes. To this end, we adapt and extend our 2D gap filling method originally designed for 2D binary shapes (Chapter 4). In Section 9.3, we discuss the case of removing both one-dimensional and two-dimensional (curve-like and surface-like) defects from 3D binary volumes. To this end, we use our 3D skeletonization method presented in Chapter 8. Section 9.4 concludes the chapter.

9.2 REMOVAL OF ONE-DIMENSIONAL GAPS FROM GRAYSCALE VOLUMES

As outlined in the introduction above, our first use-case for 3D inpainting is the detection and removal of one-dimensional (curve-like) gaps present in grayscale volumes. The utilization context for this method is described next.

9.2.1 *Application context*

Positron emission tomography (PET) is a functional imaging modality, whereby we can deduce the spatial distribution of a radio-labelled pharmaceutical which has been injected into the subject (in our case, a small animal used for experimental research). However, the spatial resolution of PET imaging is poor, and the radioactivity is often confined to a few small regions (*e.g.*, tumors) with no information on the spatial

context of those regions. In a PET/CT scanner, a computed tomography (CT) or cone-beam computed tomography (CBCT) system is incorporated into the PET scanner, allowing the acquisition of high resolution anatomical images which can be spatially co-registered with the PET functional data. For this process to work, the quality (high resolution, low noise levels) of the acquired CBCT data should be as high as possible.

In the above-mentioned applications, it is often necessary to carry out physiological monitoring of the subject (*e.g.*, measure its temperature). To do this, various types of sensors are inserted into the subject. In our current data, sensors consist of two basic materials: soft plastic tubes and sheaths (S) and hard metal wires (H). Figure 9.1 show a 2D X-ray image with the (H) and (S) component types. As visible, S components appear as 'soft' artifacts on the acquired X-ray images, and do not overall interfere badly with the 3D reconstruction process which creates the 3D grayscale volume from the available 2D X-ray images. In contrast, H bits cause streak artifacts, making the CBCT reconstruction (onto which the PET data is next overlaid) unusable. Moreover, H bits are also distracting when visualizing the data as transaxial slices.

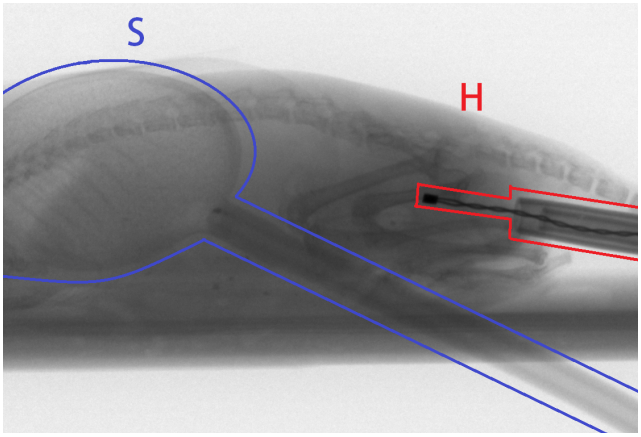


Figure 9.1: 2D X-ray images with hard metal wires (H) and soft plastic tubes (S).

Given the above, our goal is to remove the H artifacts from the final 3D reconstruction. Doing this by postprocessing a CBCT reconstructed volume is possible, but quite hard, as the metallic (H) artifacts cause streaks of high frequency and large spatial extent. Also, this process is arguably computationally intensive. An alternative route is to detect and remove the H artifacts from the entire set of 2D X-ray images which are next used in the 3D volumetric reconstruction – in our case, there are 511 such images, each being acquired from a different viewpoint, or angle, around the subject, all having 1184 by 1120 pixels. This has several advantages. First, detecting and removing *one-dimensional* (wire-like) high-contrast artifacts, like our H structures, from 2D images is arguably much easier, and faster, than removing the *three-dimensional* streak artifacts those H structures cause in the 3D reconstruction. Secondly, by preprocessing the 2D X-ray images to remove such artifacts, we can use the existing method and software already provided by the PET/CT system in place to create a wire- and artifact-free reconstruction and 3D rendering.

Several related works for detection and removal of wire-like and tube-like artifacts from 3D volumetric data exist. Closest to our use-case, Lessard *et al.* have presented an approach to segment wires in fluoroscopic images during cerebral aneurysm endovascular interventions [140]. Bismuth *et al.* presented a locally shortest-path technique for the detection and removal of of curvilinear structures from X-ray fluoroscopic images acquired live during video monitoring of the intervention [31]. The technique aims at segmenting so-called 'guide wires' inserted into the vascular system of patients during various surgical interventions. Both above methods work on 2D grayscale images which are similar to our X-ray CBCT images.

9.2.2 *Proposed method*

Our approach to wire detection and removal also works on 2D grayscale images, but uses a different set of techniques. The flowchart of our proposed method is shown in Fig. 9.2.

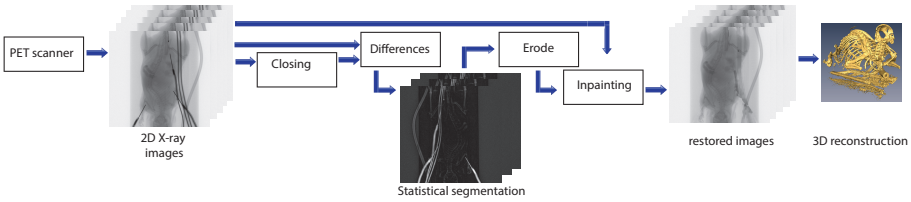


Figure 9.2: Flowchart of the proposed method.

The first step of our pipeline is to apply a morphological closing of each of the input images Ω. In detail, given a 2D disk structuring element H, we consider the dilation of Ω by H, *i.e.*, the union of copies of H_x, the element H centered at all pixels $x \in \Omega$, *i.e.*

$$\Omega \oplus H = \bigcup_{x \in \Omega} H_x. \tag{9.1}$$

Similarly, we define the erosion of Ω by H, which keeps only pixels $x \in \Omega$ where H_x fits inside Ω, *i.e.*

$$\Omega \ominus H = \{x \in \Omega | H_x \subseteq \Omega\}. \tag{9.2}$$

Since our images are grayscale rather than binary, the overall effect of dilation is to replace each pixel x by its highest-value neighbor in H_x [101]. Similarly, erosion on grayscale images replaces each pixel x by the lowest-value neighbor in H_x. Having the above operations, we next define the opening of Ω as erosion followed by dilation, *i.e.*

$$\Omega \circ H = (\Omega \ominus H) \oplus H, \tag{9.3}$$

and, analogously, the closing of Ω as dilation followed by erosion, *i.e.*

$$\Omega \bullet H = (\Omega \oplus H) \ominus H. \tag{9.4}$$

In the second step, we apply a so-called black top-hat transform, *i.e.*, compute the absolute difference D between the input grayscale image Ω and its closing $\Omega \bullet H$. The image D captures the H-like artifacts which are thinner than the size of the structuring element H. In the third step, we upper threshold the grayscale difference image D to obtain a binary mask M that precisely delineates the H artifacts. Setting the threshold value τ is, however, not a trivial process. Indeed, the difference images D computed for various input images Ω of our set of 511 X-ray images can significantly differ in terms of overall contrast and range. To arrive at a robust setting of τ, we first normalize the luminance of all images D to the same range, so as to better highlight differences. Next, for each set of input images, we manually select $N = 5$ images and segment these manually, by marking the pixels which correspond to the H artifacts. The final threshold τ to use corresponds then to the average of the grayscale values of these marked pixels over all the manually selected N images.

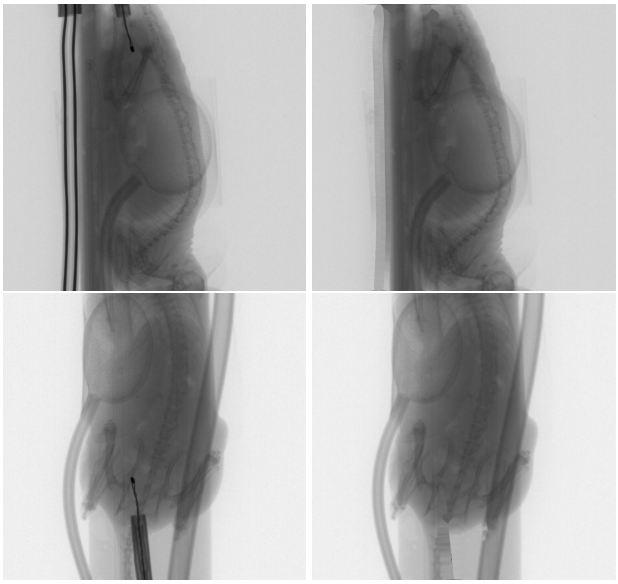
Using the threshold τ computed as above produces a conservative binary mask M which covers the H artifacts but also additional nearby pixels of similar luminances – in other words, M generates few false negatives but may contain several false positives. To remove these, we slightly erode the binary mask M by a few (2..4) pixels to generate the final mask M^{final}. In the fifth step, we use this final mask M^{final} to inpaint the captured H artifacts using the classical inpainting technique in [229], which was also used in Chapter 5. The sixth and last step of our process reconstructs the 3D grayscale volume from the set of inpainted images, using any of the standard 3D volumetric reconstruction techniques provided by the VIVID software [92] that comes with the CT scanner platform of our application.

9.2.3 *Results*

Figure 9.3 shows two images containing H artifacts (left) and the images generated by our method that detects and removes such artifacts (b). As visible, both the hard wires inserted in the animal and their surrounding sheaths are detected and removed in ways that do not create implausible image structures. Separately, S artifacts, such as the soft plastic tube visible in the bottom two images, are left in place.

Figure 9.4 shows the 3D reconstruction obtained from the set of raw unprocessed images in which artifacts have not been removed (left) compared to the reconstruction from images where artifacts have been removed using our method (right). For both cases, the shown images are computed by volumetric raytracing of the 3D volume with an isosurface-like ray function that emphasizes the subject's skeletal structure. For the top dataset, we see that there are only very few differences between the two reconstructions *and* also that both reconstructions are quasi noise-free. This indicates that, for input data where the adverse effect of H artifacts is minimal, our processing does not significantly change the obtained reconstruction (which is desirable). For the bottom dataset, we see, however, that the reconstruction from the raw images creates a significant amount of streak noise (Fig. 9.3, bottom-left). In contrast, reconstructing from our processed images largely eliminates all such noise *and* keeps the main skeletal structure visible in the noisy image.

Turning back to the main problem discussed in this section – the detection and removal of one-dimensional curve-like structures from X-ray images – a natural ques-



(a) (b)

Figure 9.3: Removal of hard artifacts (wires and sheath bits) from 2D X-ray images. (a) Original 2D X-ray images and (b) images generated by our method.

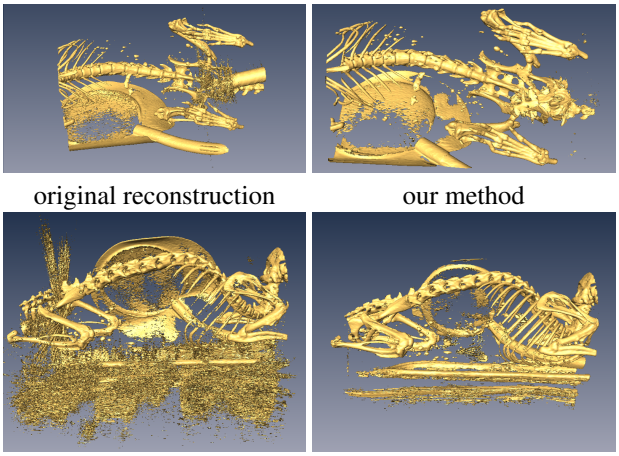


Figure 9.4: Comparison of 3D reconstructions when using the original unprocessed images (left) and the ones preprocessed by our method (right).

tion to pose is whether we can use the digital hair removal (DHR) method described in Chapter 5 for this task. Indeed, both hairs and wire-like artifacts share a number of common aspects, such as the one-dimensional structure, variable position and length in the image, and variable contrast *vs* surrounding structures. Figure 9.5 shows the application of the DHR method presented in Chapter 5 to our X-ray images. To create these results, we simply replaced all steps in the pipeline in Fig. 9.2 (except the final 3D reconstruction step) by the execution of our DHR method on the 2D X-ray images. As visible, the quality of the H-artifact detection and removal, and also the overall elimination of noise in the final reconstructions, is similar with the results of the morphology-based method described in Sec. 9.2.2. However, finding suitable parameters for applying our DHR method on our tested X-ray images is much harder than finding similar parameters for the digital hair removal use-case. The main issue here is the quite variable artifact thickness (both between different wires but also along a single wire partially wrapped in a sheath) that can be present in an image. Careful trial-and-error parameter setting does work, but the overall robustness of the DHR method for this use-case is lower than for the use-case of detecting and removing hairs. Separately, a large part of the complexity of the DHR method was dedicated to the (delicate) detection of low-contrast entangled hairs. Such low contrast and entanglement are not issues in the CBCT datasets we have seen so far. As such, the simpler morphology-based method presented in Sec. 9.2.2 appears to be a simpler and more robust solution for the H artifact removal problem than the reuse of the DHR method presented earlier.

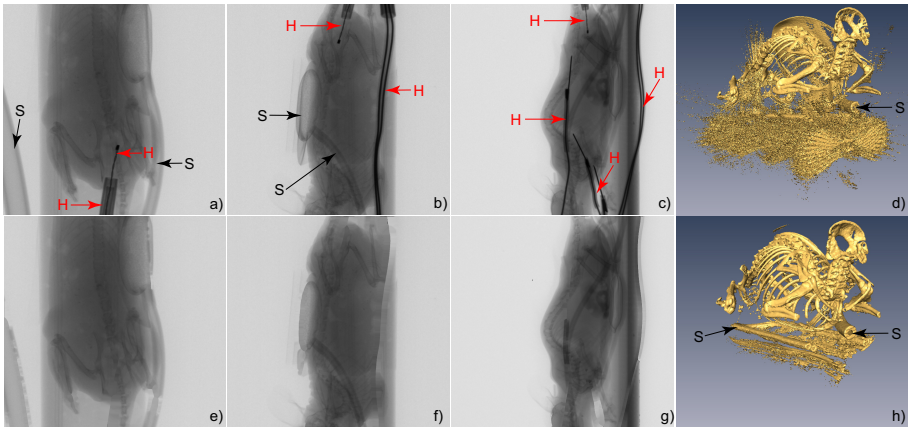


Figure 9.5: Removal of H artifacts from X-ray images using the DHR method described in Chapter 5.

9.3 REMOVAL OF ONE- AND TWO-DIMENSIONAL GAPS FROM BINARY VOLUMES

As outlined in Sec. 9.1, our aim is to detect and remove both one-dimensional and two-dimensional gap-like artifacts from both grayscale and binary 3D volumes. Section 9.2 has presented a method that treats the relatively simpler case of detection and removal

of one-dimensional artifacts. In this section, we present a separate method that focuses on the detection and removal of the joint set of one-dimensional and two-dimensional gap-like artifacts from 3D binary volumes.

9.3.1 *Proposed method*

The proposed method is an adaptation to 3D of the 2D gap-filling method presented in Chapter 4. To understand the rationale of the performed changes, we briefly outline the 2D method next: Given a binary pixel shape $\Omega \subset \mathbb{Z}^2$, we first compute the skeleton $S(\Omega_{oc})$ of the shape Ω_{oc} obtained by opening, next closing, Ω by the corresponding morphological operations. Note that Ω_{oc} with a disk structuring element of radius ρ fills all gaps in Ω whose thickness is smaller than ρ. Next, we compute the set of skeletal fragments $F = S(\Omega_{oc}) \setminus \Omega$, *i.e.*, points of the skeleton of the hole-free shape which are outside of the original shape. As discussed in Sec. 4.3.2, such fragments correspond one-to-one to gaps that cut deeply in the input shape Ω. The final step of the 2D method is to fill the gaps by essentially convolving the skeleton-fragment set F with 2D disks whose radii equal the distance transform values of the shape Ω_{co} obtained by first closing, then opening, Ω. For full details, we refer to Sec. 4.3.3.

Technically speaking, we can immediately generalize the above method to 3D by replacing the two-dimensional medial axis of a shape by its so-called surface skeleton, which is its exact equivalent in 3D following the skeleton definition (Eqn. 2.2). Figure 9.6 (bottom path) shows the effects of this process for a test case – a frog model cut in the middle by a single simple thick planar cut. In the figure, voxels $\mathbf{x} \in \Omega$ in the original shape are marked red; and voxels added by the gap-filling process are marked green, respectively. As visible, using the surface skeleton has the desired effect of filling in the gap present in the model quite well. However, as by-product, many shallow gaps present on the model's surface are also filled – see *e.g.* green details between the frog's fingers and in the creases behind the ankles of the hind legs. These artifacts are not surprising: indeed, the shape Ω_{oc} whose surface-skeleton $S^{surf}(\Omega_{oc})$ we use, is an 'inflated' version of Ω_{oc}. Recall, this inflation is needed to close the present gaps before the skeleton computation. However, this inflation also closes *detail* gaps, like the ones mentioned above. As such, these gaps may generate surface-skeleton fragments which are outside Ω, thus, are part of F. In turn, reconstructing the shape from such fragments produces the undesired fill-ins of detail gaps.

At a closer analysis of the above phenomenon, we observed that the main cause for it is the higher sensitivity of the 3D surface skeleton to inflation-induced changes on a shape, as compared to 2D medial axes. One way to alleviate this problem would be to simplify the surface skeleton $S^{surf}(\Omega_{oc})$ prior to its use in computing the gap-set F. However, this creates the undesired effect that the gaps are next not filled in a way that reconstructs the local shape surface *smoothly*. Indeed, a gap in a 3D objecty can be seen as a cross-section whose shape is, in most cases, far from circular. As such, to fill the gap in a plausible way, one needs to use the (almost) full surface skeleton. Note that, in contrast, this is not an issue in the 2D case: The skeletal fragments present in 2D gaps are simple one-dimensional curve segments (internal skeleton branches), which capture perfectly well the simpler configuration of a 2D gap, and which actually are not affected by skeleton simplification. In 3D, as outlined above, a gap will contain

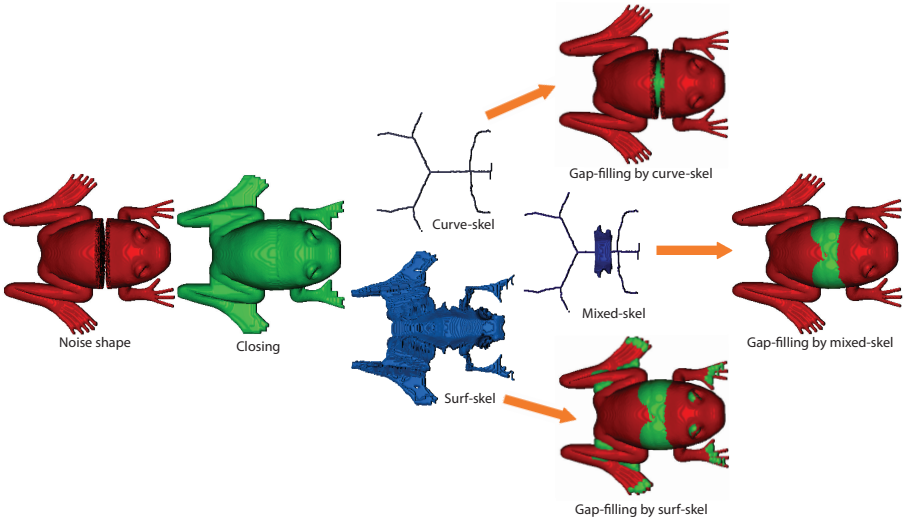


Figure 9.6: Three-dimensional gap detection and filling using surface and curve skeletons.

a *mix* of both surface-skeleton terminal manifolds (which capture shape details) and internal surface-skeleton manifolds (which capture the coarse shape topology).

A second refinement of the above idea is to use the curve skeleton $S^{curve}(\Omega_{oc})$ in the 3D detection-and-reconstruction process, instead of the surface skeleton. The main advantage of this idea is that the curve skeleton has the same simple one-dimensional topology as the classical 2D medial axis, so robustly detecting its fragments which are present in the gap-set F is very easy. However, as well known, a curve skeleton does not capture the local shape *geometry* well. More precisely, if we use such a skeleton in the reconstruction step of the algorithm, we will reconstruct our shape by filling in the detected gaps with locally-tubular structures whose thickness equals the local shape thickness. This situation is shown in the top path in Fig. 9.6. As visible, we now do not spuriously fill in small-scale surface gaps, but on the other hand we also do not succeed to fully fill the large central cut (which has a non-circular cross-section).

Summarizing the above, we conclude that, in the 3D context

- curve skeletons are good for gap detection, but not for gap removal;
- surface skeletons are good for gap removal, but not for gap detection.

This observation leads us naturally to the use of a combination of the two skeleton types to handle our problem. Specifically, we proceed as follows:

1. Compute the curve skeleton $S^{curve}(\Omega_{oc})$ and use it to create the gap-set F^{curve} of all points in the curve skeleton which are outside the input shape Ω;
2. Compute the set F^{surf} of points in the surface skeleton which are outside the input shape Ω;
3. From F^{surf}, we eliminate all voxels which are not connected (within F^{surf} itself) to at least a voxel in F^{curve}, by executing a simple flood-fill operation from

F^{curve} onto F^{surf}. This, in other words, removes from F^{surf} all spurious fragments which led to the filling of small surface details, and yields the final set of fragments F^{final};

4. Use F^{final} to reconstruct the shape by convolving the voxels $\mathbf{x} \in F^{final}$ by balls of radii $DT_{\partial\Omega_{co}}(\mathbf{x})$, where $DT_{\partial\Omega_{co}}$ is the distance transform of the surface of the input shape after morphological closing and opening.

The middle path in Fig. 9.6 shows the result of applying this mixed curve-and-surface skeleton method to our test shape. As visible, the large central cut is restored as well as when using the surface-skeleton only (bottom path in the figure), and no spurious detail is filled in, just as when using the curve-skeleton only (top part in the figure).

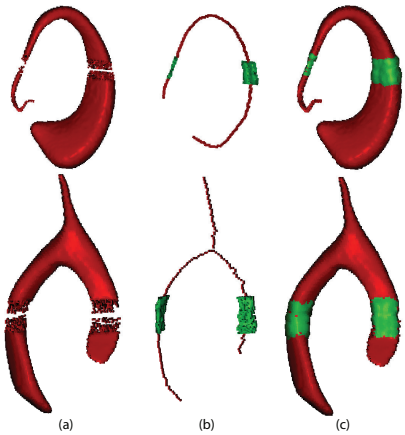


Figure 9.7: Reconstruction of tubular shapes. (a) Damaged shape; (b) Curve skeleton (in red) with selected portions of surface skeleton (in green); (c) Reconstruction result.

9.3.2 *Results*

We implemented the above gap-detection-and-removal method for voxel shapes as follows. Morphological operations (opening, closing) are implemented by using simple convolution with a 3D structuring element. In our experiments, we used three well-known such structuring elements: cubic, ball, and cross (see details at top of Fig. 9.8). Reconstruction done by using surface-skeletons only is achieved by using surface skeletons computed by the integer medial axis method (IMA) [102]. The reason for this choice is that IMA is fast, simple to implement, and delivers overall high-quality (unsimplified) surface skeletons, as we have seen from our evaluation preented earlier in Chapter 7. As such, IMA is an ideal method if we only need surface skeletons. Reconstruction done by using combined curve-and-surface skeletons is achieve by using both skeleton types as computed by our advection method described in Chapter 8. The choice for this method is motivated by its high speed, centeredness of the delivered skeletons, ability to compute both skeleton types, and the fact that it guarantees that

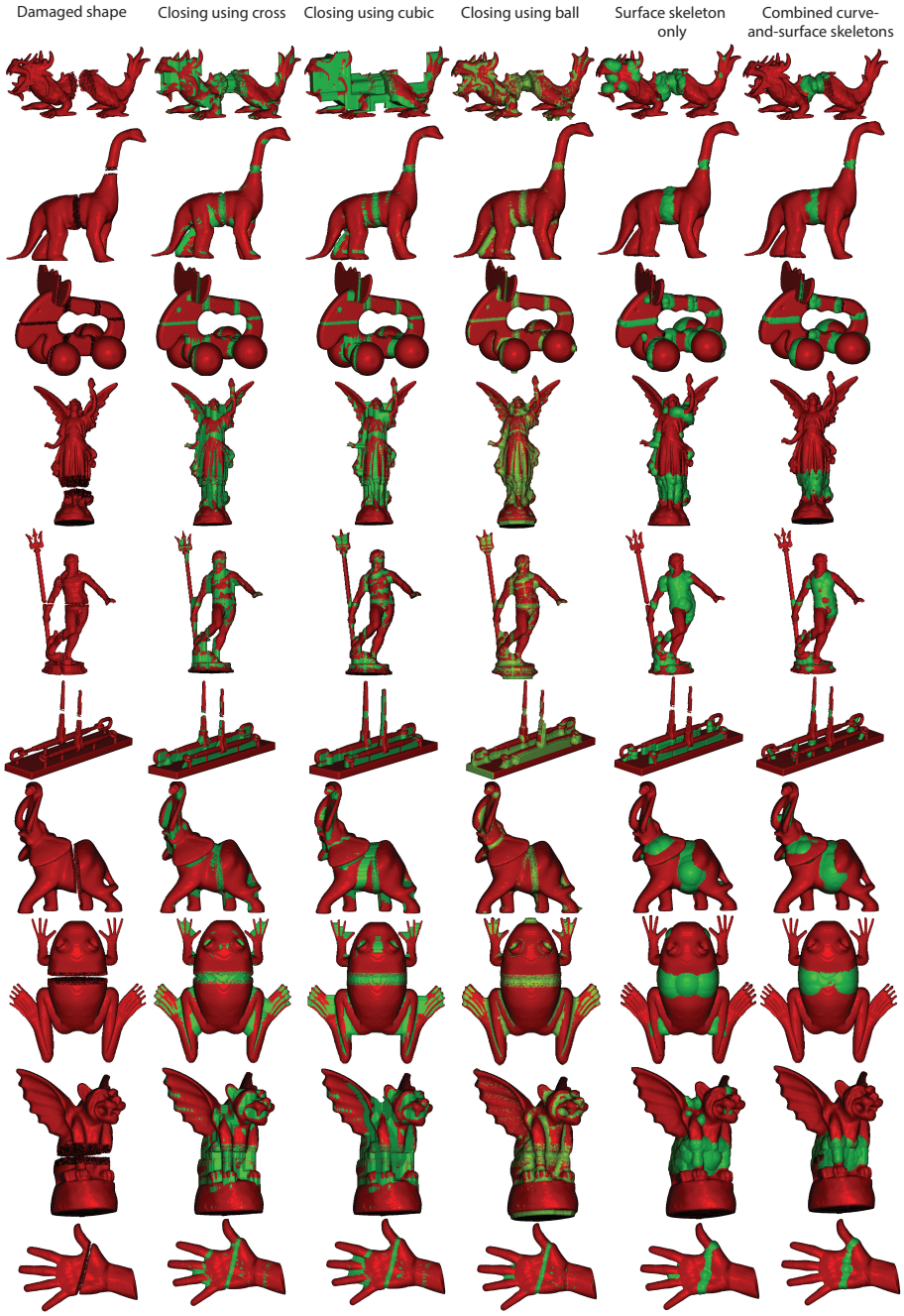


Figure 9.8: Comparison of our gap-filling method (right column) with four other methods.

curve skeletons are contained in their corresponding surface skeletons. This containment property is *crucial* for the success of our reconstruction: Indeed, in step 3 of the method (described above), we need to execute a flood-fill from the curve skeleton onto the surface skeleton. For this to work, the curve skeleton should be embedded in the surface skeleton.

To assess both the qualitative results of our method and its scalability, we ran it on about 20 different shapes, voxelized from polygonal models at resolutions up to 5100^3 voxels, using *binvox* [166]. Our method was implemented in C++ on the CPU and executed on a desktop PC Intel Core i7-2600 3.40 GHz with 16 GB of RAM.

Figure 9.7 shows a first example of skeleton-based reconstruction for a relatively simple tubular shape. As visible, the combined curve-and-skeleton based reconstruction method can successfully detect and fill in even the relatively large gap having jagged edges shown in the figure. Figure 9.8 shows several additional reconstruction examples, for shapes having a more complex geometry and topology than the ones shown earlier in Figure 9.7. Figure 9.8 shows a subset of 10 shapes from our test set. Gaps were created in the input shapes procedurally, by cutting them at various places with implicit functions modeling planes of various orientation and spheres of various radii, respectively. Furthermore, noise was added on the internal cut surfaces by randomly removing a small set of voxels from these surfaces. This creates more jagged cuts, which are arguably more challenging to reconstruct. Finally, we compared our proposed method (usage of both surface and curve skeletons) to four other alternative methods: simple top-hat morphological gap closing by using three types of structuring elements (ball, axis-aligned cube, and axis-aligned cross); and the usage of the surface-skeleton only in the method presented in Sec. 9.3.1. In Figure 9.8, red shows original points in Ω which are also present in the reconstruction; green shows points added to Ω by the reconstruction; and blue shows points removed from Ω by the reconstruction.

9.3.3 *Discussion*

We next discuss several relevant aspects for our 3D gap detection-and-removal method, organized with respect to a number of desirable properties of the reconstruction.

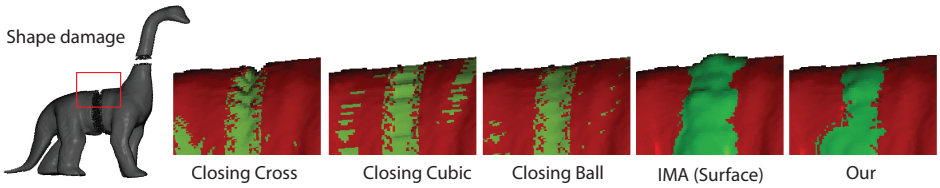


Figure 9.9: Reconstruction smoothness.

Smoothness: As mentioned at several points during this thesis, inpainting-like operations should produce a 'plausible' reconstruction of the damaged areas. While a formal definition of plausibility is hard to give in general, the vast majority of 2D

inpainting literature mentions *smoothness* of the reconstructed signal over, and along the boundaries of, the reconstructed area to be a key requirement. As such, we naturally adopt this requirement for our 3D case too. As visible from Figure 9.8 and the detailed Figure 9.9, variations in the reconstruction smoothness exist between the different tested methods. Overall, the simpler morphological closing methods produce noisier results, while the skeleton-based reconstruction produces smoother results. Within the last category, we also see that the combined curve-and-surface reconstruction produces the smoothest results, in the sense of a reconstructed surface (green) which closely follows the curvature of the surrounding original surface (red).

Locality: As outlined in Chapter 4 (for the equivalent 2D case) and earlier in this chapter, our gap-filling reconstruction should detect and remove only deep gaps that significantly cut the shape, but leave shallow gaps (detail indentations) of the input surface untouched. Similarly to the test example in Fig. 9.6, Figure 9.8 shows that the morphological closing methods cannot, in general, make this difference – as such, they fill all gaps whose size is smaller than the structuring element size. This is easiest visible by considering the amount of green present in the reconstructed images which does not correspond to the locations of the cuts visible in the shapes in the leftmost column. Examples of this situation are filling the small gaps between the details of the dragon surface (top row), spikes of the trident (neptune model, fourth row from top), or frog's fingers (third row from bottom). The combined curve-and-surface method suffers far less from such issues.

Simplicity and scalability: Implementing the proposed 3D inpainting method is simple: The entire method consists of a number of trivial morphological opening and closing passes, the computation of a 3D curve and surface skeleton, a flood fill operation on voxel volumes, and the reconstruction of a 3D voxel shape from a selected number of skeleton points, by using the shape's medial axis transform (MAT). All these operations can be computed in linear time in the size $\|\Omega\|$ of the input shape (see Chapter 8). As such, the overall 3D inpainting times are comparable with the skeletonization times reported in Sec. 8.4.5.

9.4 CONCLUSIONS

In this chapter, we have addressed the problem of detection and removal of thin and elongated gaps that damage the volumetric structure of 3D shapes. To this end, we explored two separate use-cases, as follows.

First, we consider the detection and removal of one-dimensional (curve-like) gap-like defects from 3D grayscale volumes. Since these defects maintain their one-dimensional structure in 2D projections of the data, it is possible to cast the 3D detection-and-removal problem as a two-dimensional image processing problem followed by a 3D reconstruction problem. For the processing of the 2D images, we have compared our digital hair removal (DHR) approach with a simpler approach based on morphological operations. For our application at hand – the detection and removal of metallic wire artifacts from CBCT scans – we have shown that the simpler morphological approach yields results which are visually as good as the more complex DHR

approach, leading to the possibility to remove artifact-induced noise from the final 3D reconstructions.

Our second use-case considers the detection and removal of one-dimensional and two-dimensional gap-like defects, such as deep and thin volumetric cuts in binary 3D shapes. Since these defects have a significantly more complex geometry than the one-dimensional curve-like defects discussed above, we advocate a volumetric method for their detection and removal. To this end, we extend the gap detection and removal method described in Chapter 4 to 3D, by using a combined curve-and-surface skeleton. Such skeletons can be readily and efficiently provided by the 3D skeletonization method described in Chapter 8. The proposed 3D inpainting method was shown to produce better results, in terms of detail preservation and gap removal, as compared to classical morphological closing methods.

10 CONCLUSION

10.1 SUMMARY

As stated in Chapter 1, the main goal of this thesis has been to explore whether skeletal descriptors can be effectively and efficiently used to support the construction of novel shape restoration methods. As outlined in Chapter 2, the above-mentioned restoration process can be decomposed into two parts – detecting the damaged regions to restore, and the restoration proper. As such, the success of using skeletons for shape restoration can be mapped to how skeletons support the two operations mentioned above.

At a high level, our conclusion is that skeletons can, indeed, provide added value for designing various types of restoration methods for both 2D and 3D shapes. For the 2D context, our preliminary work on facial inpainting shows that the major difficulty in this area is not the restoration proper (which can be done effectively using standard inpainting methods), but the automatic and robust detection of regions to restore. Using typical detection methods which either analyze the image locally or only consider global statistical image properties are of limited effectiveness, as they cannot guarantee structural invariants of the detected regions. As such, inpainting is not guaranteed to act on precisely those areas where it is required to.

Following this observation, and given the well-known power of skeletons to capture both local geometrical and global topological aspects of a shape, our hypothesis that skeletons are useful for shape restoration gains additional weight. In Chapter 4, we showed the first example of skeletal-based restoration for 2D binary images, in the context of removing thin-and-deep cracks from such images. The same method was next adapted to perform crack-free segmentations of noisy skin lesions from dermatoscopic images. Here, skeletons proved essential to model the joint aspects of thinness and depth of cracks occuring in a binary image. We next extended this method in Chapter 5 to automaticaly detect and remove hairs from color dermatoscopic images. As for the crack detection-and-removal use-case, skeletons proved here essential instruments to model the geometry and topology of thin hairs.

While the first part of this thesis (Chapters 3-5) focuses on using 2D medial axes to restore 2D images, the second part of this thesis (Chapters 6-9) focuses on using the more complex 3D surface-and-curve skeletons for the restoration of 3D volumetric shapes. A first critical step in exploring this avenue is determining which are efficient and effective 3D skeletonization methods that can be optimal candidates to be subsequently used in our 3D restoration work. At the time we did this work, however, no recent detailed survey of 3D skeletonization methods was available, while increasingly many skeletonization methods were appearing in the scientific arena. As such, our first step has been to execute two surveys on the state-of-the-art of 3D skeletonization. The first survey, described in Chapter 6, covers the computation of 3D mesh-based curve skeletons with collapse methods, which were deemed at the time of writing to be among the most successful skeletonization methods from the perspective of accuracy, computational speed, and scalability. Our second survey, presented in Chapter 7,

extends the scope of the first survey by covering both surface and curve skeletons, with a focus on voxel-based methods.

The joint conclusion of both surveys was that no skeletonization method, from the examined ones, is optimal from the perspective of all considered quality criteria. In particular, finding a skeletonization method able to deliver both surface and curve skeletons with high accuracy, high computational scalability, multiscale regularization, and having a reasonably simple implementation, was hard. While this may not appear as a surprise, this fact posed an obstacle to our subsequent aim to use such skeletons for our 3D shape restoration research. As such, we embarked on the task of constructing such a method. Our work, described in Chapter 8, presents a 3D skeletonization method that complies with all above desired requirements. Additionally, the proposed method unifies the so-far different concepts of multiscale importance used for 2D medial axes, 3D surface skeletons, and 3D curve skeletons, in a single mathematical framework based on the advection of mass from a shape's boundary to its surface skeleton, next to its curve skeleton, and next to the shape's center.

Having the above 3D skeletonization tool, we next turned to our aim of creating skeleton-based restoration methods for 3D shapes. Chapter 9 presents two such methods. The first method adapts our 2D gap-removal algorithm presented in Chapter 4 to find and remove gaps from 2D views of a 3D shape, and next synthesizes the gap-free 3D shape from the restored views. To validate our method, we applied it to the task of removing wire artifacts from cone beam computer tomography (CBCT) images. The second method removes gaps from 3D binary shapes, using the surface and curve skeletons computed by our advection skeletonization presented in Chapter 8. Compared to classical morphological gap repairing, our method was shown to deliver better results in terms of restoration smoothness and detail preservation.

Summarizing the above, we believe that the research presented in this thesis has shown substantial evidence that supports our initial hypothesis – the fact that skeletons are useful and usable tools for the creation of methods for shape restoration using inpainting and inpainting-like techinques, for a diverse range of use-cases and applications. In the following sections, we further reflect on our conclusions, considering each research topic separately.

10.2 PART 1: REPAIRING TWO-DIMENSIONAL IMAGES

10.2.1 *Facial Image Restoration (Chapter 3)*

The first step of our research focused on the specific task of repairing a set of damaged facial images by using inpainting techniques. For the detection of damaged regions, we proposed to use a statistical approach based on comparing the damaged image with a predefined collection of high-quality images of the same type. For the repairing of the detected damages, we combined several standard inpainting methods with an exemplar-based approach that reuses information from our high-quality image database.

The global outcome of this work was to find out that, for the context of 2D image restoration, a good detector for the damaged regions is far more critical than the quality of the subsequently used inpainting method. Moreover, when comparing a num-

ber of existing inpainting methods for our type of damages (which include relatively small-scale areas of the input image), we found little variation in terms of quality. As such, we decided to next focus the core of our work on the effective detection of damaged areas and less on the creation of novel inpainting techniques.

10.2.2 *Gap-Sensitive Segmentation and Restoration of Images (Chapter 4)*

Our first step into using medial descriptors to detect and restore defects considers the case of 2D binary shapes affected by cracks that are both thin and penetrate deeply from the boundary of the shape into its interior. Such cracks occur in a variety of contexts, such as poor threshold-based segmentation of noisy grayscale images, intentional damages done to a binary image, or occluding objects appearing in front of the image. One major difficulty of standard restoration methods for this type of defects is that they remove both the defects and important small-scale details present on the shape's boundary [21, 22, 122, 244]. Also, several such methods require non-trivial effort from the end user in the form of manual delineation of the defects or parameter setting.

To address this problem, we proposed an automatic method that both detects and removes cracks from 2D and 3D binary shapes. Key to our method is the analysis of the defect's size and position with respect to the shape's skeleton. This simple heuristic allows an effective and efficient way to discriminate boundary details (which should be preserved) from deep cracks (which should be removed). In the same time, the skeleton offers a simple and efficient way to remove (fill in) the cracks by using its associated medial axis transform information. One salient application of this method is the segmentation of tumors occluded by hairs from dermatoscopic images, so that the hairs are automatically removed, but small-scale details on the lesion boundary are preserved.

10.2.3 *Digital hair removal in skin tumor images (Chapter 5)*

Following the promising results of our gap-removal method described in Chapter 4, and the positive feedback on the associated tumor segmentation method from dermatology specialists, we decided to refine our results in this direction. In Chapter 5, we proposed a new approach to digital hair removal (DHR) from dermoscopic images, based on a threshold-set image representation. For every threshold, we modify our gap-detection algorithm to find hairs, and merge results in a single mask image. Next, we detect hairs in this mask by a combination of morphological filters and medial-axis descriptors. We tested our method on over 300 dermatoscopy images, and compared it with six other state-of-the-art DHR methods. The obtained results show, both qualitatively and quantitatively, that our method produces superior results to the compared methods.

10.3 PART 2: REPAIRING THREE-DIMENSIONAL SHAPES

10.3.1 *3D Curve Skeletonization Comparison (Chapter 6)*

As outlined above, a precondition to using skeletons to repair 3D shapes is the availability of a good 3D skeletonization method. To find such a method, we performed a survey focused on several recent curve skeletonization methods. In particular, we consider the family of mesh-collapse skeletonization methods wich, at the time of our research, were deemed as extremely promising in the skeletonization community. As quality factors, we consider the well-accepted feature set described in the classical survey of Cornea *et al.* [59].

The performed comparison led to a number of surprising results. First and foremost, we observed that mesh-collapse skeletonization methods did not always perform as well as one would expect from the current literature, except for computational scalability. In particular, issues like detail preservation, centeredness, and smoothness were seen to be challenging for these methods. As such, we concluded that the more traditional voxel-based skeletonization methods may still be the more suitable method-class for our application.

10.3.2 *3D Voxel-based Skeletonization Comparison (Chapter 7)*

As outlined above, our comparison study did not find mesh-based skeletonization methods to be optimal for our application context from the perspective of the studied quality criteria. As such, a natural next step was to study and compare voxel-based skeletonization methods. In Chapter 7, we present such a study, which, in addition to the work in Chapter 6, also considers surface skeletons.

We compared six mesh-based curve-skeletonization methods and ten voxel-based curve- and surface-skeletonization methods along quality criteria proposed in [59]: homotopy, invariance, thinness, centeredness, smoothness, detail preservation, and resolution robustness. The comparison was done both qualitatively (by visually assessing how well skeletons fulfill the desirable properties), but also quantitatively, by the usage of two novel skeleton-distance metrics based on the Hausdorff distance.

The results of this work are twofold. First, our comparison showed that, similar to the mesh-based curve skeleton case, there is no clear winner method for the voxel-based curve (and especially surface) skeletonization field. In particular, having a method that supports multiscale regularization, ensures centered skeletons, and is computationally scalable was hard to find. The second result of our comparison was the construction of a public, open-access, benchmark database for skeleton-methods comparison. The database contains an extensive collection of 3D shapes, voxelization software, and skeletonization software. It can be used by any researchers interested to test side-by-side their new skeletonization methods vs existing established ones. To our knowledge, our initiative is the first public benchmark for 3D skeletonization.

Jointly, the work presented in Chapters 6 and 7 represent the largest qualitative and quantitative comparison of 3D skeletonization methods performed since the well-known similar survey of Cornea *et al.* [59] in 2006. Together with the above-mentioned open skeletonization database, we believe that our work lays the founda-

tions for an easier, and more open, way for researchers in skeletonization to evaluate their work.

10.3.3 *Unified Curve-and-Surface Skeletonization Framework (Chapter 8)*

As pointed out above, a 'winner method' that computes 2D and 3D curve and 3D surface skeletons with all desirable properties (thinness, homotopy preservation, centeredness, multiscale regularization, speed, and ease of use) was found to be lacking. In particular, we found that there are only a few multiscale skeletonization methods ([80, 168, 228] for 2D shapes, and [70, 109, 187] for 3D shapes). Interestingly, all these methods use the same general collapse-based principle to compute a skeletal multiscale.

As such, we decided to embark on the construction of a unified 2D and 3D skeletonization framework that would (a) produce skeletons compatible with all our requirements, and (b) use a single multiscale importance metric that generalizes and unifies the earlier work outlined above in this area. Our approach, described in Chapter 8, models the skeleton detection and regularization by a conservative mass transport process from the shape's boundary to its surface skeleton, next to its curve skeleton, and finally to the shape center. The resulting mass-density field can be thresholded to obtain a multiscale representation of progressively simplified surface, or curve, skeletons. We also proposed a numerical implementation of our framework which is demonstrably stable and has high computational efficiency. We tested our new method on the skeletonization benchmark created during our earlier comparisons, and confirmed that it produces both curve and surface skeletons that compare favorably with existing methods.

10.3.4 *3D Artifact Detection and Removal (Chapter 9)*

The last chapter of our work puts all of our earlier results together in the context of showing how skeletal-based descriptors can be used to perform repairing of 3D shapes. In this context, Chapter 9 describes two such applications.

In the first application, we aim to remove thin-and-elongated artifacts caused by metallic wire-like insertions present in anatomic volumes that are scanned with cone-beam computer tomography (CBCT). As well known, such metallic artifacts can cause severe reconstruction problems, ultimately leading to reconstructed 3D volumes which are unusable. To alleviate such problems, we adapt our 2D gap-removal algorithm presented in Chapter 4 to detect and remove the wire projections present in several 2D views (X-ray images) of the 3D data. After the wire projections are removed, standard CT reconstruction can take place, yielding artifact-free volumes that can be further easily explored by the interested medical researchers.

In our second application, we show how 3D curve-and-surface skeletons can be used to detect and remove crack-like gaps present in 3D binary volumetric shapes. For this, we adapt again our 2D gap-removal algorithm presented in Chapter 4 to detect gaps into details (to be preserved) and defects (to be removed) based on their position with respect to the simplified surface skeletons of the respective shapes. We show next how, in this context, our novel advection-based skeletonization method

(Chapter 8) produces better results than other skeletonization methods. Hereby, thus, we show the added-value of three-dimensional skeletons in the context of damage detection and repairing for three-dimensional shapes.

BIBLIOGRAPHY

[1] Q. Abbas, M. Celebi, and I. Garcia. Hair removal methods: A comparative study for dermoscopy images. *Biomedical Signal Processing and Control*, 6: 395–404, 2011.

[2] Q. Abbas, I. Fondon, and M. Rashid. Unsupervised skin lesions border detection via two-dimensional image analysis. *Comp. Meth. Prog. Biom.*, 104:1–15, 2011.

[3] AFP. Australian Federal Police, National Missing Persons Coordination Centre, 2012. URL http://www.missingpersons.gov.au.

[4] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. Sig. Process.*, 54(11):43–61, 2006.

[5] N. Ahuja and J. Chuang. Shape representation using a generalized potential field model. *IEEE TPAMI*, 19(2):169–176, 1997.

[6] N. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[7] V. Amaral and C. E. Thomaz. Normalização espacial de imagens frontais de face. Technical report, Dept. of Electrical Engineering, Univ. Center of FEI, Brazil, 2008. http://fei.edu.br/cet/relatorioTecnico_012008.pdf.

[8] N. Amenta, S. Choi, and R. Kolluri. The power crust. In *Proc. SMA*, pages 65–73, 2001.

[9] ANSI. US standard for digital image formats for use with the facial biometric (INCITS 385), May 2004.

[10] E. Aptoula and S. Lefèvre. A comparative study on multivariate mathematical morphology. *Pattern Recogn. Lett.*, 29(2):109–118, 2008.

[11] C. Arcelli, G. S. di Baja, and L. Serino. Distance-driven skeletonization in voxel images. *IEEE TPAMI*, 33(4):709–720, 2011.

[12] M. Ashikhmin. Synthesizing natural textures. In *Proc. Symp. Interactive 3D Graphics*, pages 217–226. ACM Press, 2001.

[13] C. Aslan, A. Erdem, E. Erdem, and S. Tari. Disconnected skeleton: Shape at its absolute scale. *IEEE TPAMI*, 30(12):2188–2203, 2008.

[14] A.Torsello and E. Hancock. Correcting curvature-density effects in the Hamilton-Jacobi skeleton. *IEEE TIP*, 15(4):877Ð891, 2006.

[15] O. K. C. Au, C. Tai, H. Chu, D. Cohen-Or, and T. Lee. Skeleton extraction by mesh contraction. In *Proc. ACM SIGGRAPH*, pages 441–449, 2008.

[16] O. Ayinde and Y. Yang. Face recognition approach based on rank correlation of Gabor-filtered images. *Pattern Recogn.*, 35(6):1275–1289, 2002.

[17] X. Bai and L. Latecki. Path similarity skeleton graph matching. *IEEE TPAMI*, 30(7):1282–1292, 2008.

[18] X. Bai, L. Latecki, and W. Y. Liu. Skeleton pruning by contour partitioning with discrete curve evolution. *IEEE TPAMI*, 3(29):449–462, 2007.

[19] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera. Filling-in by joint interpolation of vector fields and gray levels. *IEEE Trans. Imag. Proc.*, 10 (12):1200–1211, 2001.

[20] A. Belyaev, S. Yoshizawa, and H. P. Seidel. Skeleton-based variational mesh deformations. *CGF*, 26(3):255–264, 2007.

[21] G. H. Bendels, R. Schnabel, and R. Klein. Detail-preserving surface inpainting. In *In proceedings of The 6th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST), Eurographics Association*, pages 41–48, 2005.

[22] A. Bermano, A. Vaxman, and C. Gotsman. Online reconstruction of 3D objects from arbitrary cross-sections. In *ACM Transactions on Graphics (TOG)*, volume 30, 2011.

[23] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proc. ACM SIGGRAPH*, pages 417–424, 2000.

[24] M. Bertalmio, G. Sapiro, and A. Bertozzi. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proc. CVPR*, pages 355–362, 2001.

[25] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. *IEEE Trans. Imag. Process.*, 12(8):882–889, 2003.

[26] M. Bertalmio, V. Caselles, S. Masnou, and G. Sapiro. Inpainting. In *Encyclopedia of Computer Vision*. Springer, 2011.

[27] G. Bertrand. A parallel thinning algorithm for medial surfaces. *Pattern Recogn Lett*, 16(9):979–986, 1995.

[28] G. Bertrand and G. Malandain. A new characterization of three-dimensional simple points. *Pattern Recogn. Lett.*, 2(15):169–175, 1994.

[29] S. Beucher. Digital skeletons in Euclidean and geodesic spaces. *Signal Processing*, 38(1):127–141, 1994.

[30] T. Biben, K. Kassner, and C. Misbah. Phase-field approach to three-dimensional vesicle dynamics. *Phys. Rev. E*, 72:041921, 2005.

[31] V. Bismuth, R. Vaillant, H. Talbot, and L. Najman. Curvilinear structure enhancement with the polygonal path image – application to guide-wire segmentation in X-ray fluoroscopy. In *Proceedings of the 15th International Conference on Medical Image Computing and Computer-Assisted Intervention - Volume Part II*, MICCAI'12, pages 9–16. Springer-Verlag, 2012.

[32] B.Kim, J. Kim, and J. Park. Exemplar based inpainting in a multi-scaled space. *Optik - International Journal for Light and Electron Optics*, 2015. DOI 10.1016/j.ijleo.2015.07.168.

[33] H. Blum. *A transformation for extracting new descriptors of shape*. Models for the perception of speech and visual form. MIT Press, 1967.

[34] H. Blum. Biological shape and visual science. *J. Theor. Biol.*, 38:205–287, 1973.

[35] F. Bornemann and T. März. Fast image inpainting based on coherence transport. *J. Math. Imaging Vis*, 28:259–278, 2007.

[36] S. Bouix, K. Siddiqi, and A. Tannenbaum. Flux driven automatic centerline extraction. *Medical Image Analysis*, 9(3):209–221, 2005.

[37] A. Bugeau and M. Bertalmio. Combining texture synthesis and diffusion for image inpainting. In *Proc. VISAPP*, pages 26–33, 2009.

[38] A. Bugeau, M. Bertalmio, V. Caselles, and G. Sapiro. A unifying framework for image inpainting. Technical report, Intitute for Math. and Applications (IMA), 2009.

[39] W. Bussab and P. Morrettin. *Estátistica Básica*. Editora Saraiva, São Paulo, Brazil, 2002.

[40] J. Cao, A. Tagliasacchi, M. Olson, H. Zhang, and Z. Su. Point cloud skeletons via Laplacian based contraction. In *Proc. IEEE SMA*, pages 187–197, 2010.

[41] T. Cao, K. Tang, A. Mohamed, and T. Tan. Parallel banding algorithm to compute exact distance transform with the GPU. In *Proc. SIGGRAPH I3D Symp.*, pages 134–141, 2010.

[42] W. Casaca, M. Boaventura, M. de Almeida, and L. Nonato. Combining anisotropic diffusion, transport equation and texture synthesis for inpainting textured images. *Pattern Recognition Letters*, 36:36–45, 2014.

[43] V. Caselles, G. Haro, G. Sapiro, and J. Verdera. On geometric variational models for inpainting surface holes. *Computer Vision and Image Understanding*, 111(3):351–373, 2008.

[44] O. Castillo. Survey about facial image quality. Technical report, Fraunhofer IGD, TU Darmstadt, 2006.

[45] M. Celebi, H. Kingravi, B. Uddin, H. Iyatomi, A. Aslandogan, W. Stoecker, and R. Moss. A methodological approach to the classification of dermoscopy images. *Comput Med Imaging Graph*, 31(6):362–373, 2007.

[46] T. Chan and J. Shen. Mathematical models for local deterministic inpaintings. In *Tech. Report CAM 00-11*. Image Processing Group, UCLA, 2000.

[47] T. Chan and J. Shen. Non-texture inpainting by curvature driven diffusions. In *Tech. Report CAM 00-35*. Image Processing Group, UCLA, 2000.

[48] M. Chang, F. Leymarie, and B. Kimia. Surface reconstruction from point clouds by transforming the medial scaffold. *CVIU*, (113):1130–1146, 2009.

[49] J. Chaussard, M. Couprie, and H. Talbot. A discrete lambda-medial axis. In *Proc. DGCI*, pages 232–243. Springer, 2009.

[50] J. Chou, C. Yang, and S. Gong. Face-off: Automatic alteration of facial features. *Multimedia Tools and Applications*, 56(3):569–596, 2012.

[51] J. Christensen, M. Soerensen, Z. Linghui, S. Chen, and M. Jensen. Prediagnostic digital imaging prediction model to discriminate between malignant melanoma and benign pigmented skin lesion. *Skin Res. Technol.*, 16, 2010.

[52] M. Chuang and M. Kazhdan. Fast mean-curvature flow via finite-elements tracking. *CGF*, 30(6):1750–1760, 2011.

[53] B. Chung and C. Yim. Hybrid error concealment method combining exemplar-based image inpainting and spatial interpolation. *Signal Processing: Image Communication*, 29:1121–1137, 2014. DOI 10.1016/j.image.2014.09.009.

[54] U. Clarenz, U. Diewald, G. Dziuk, M. Rumpf, and R. Rusu. A finite element method for surface restoration with smooth boundary conditions. *CAGD*, 21 (5):427 – 445, 2004.

[55] U. Clarenz, M. Rumpf, and A. Telea. Surface processing methods for point sets using finite elements. *Computers & Graphics*, 28(6):851–868, 2004.

[56] F. Cokelaer, H. Talbot, and J. Chanussot. Efficient robust d-dimensional path operators. *IEEE J. Selected Topics in Signal Processing*, 6(7):830–839, 2012.

[57] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE TPAMI*, 24(5):603–619, 2002.

[58] N. Cornea, D. Silver, X. Yuan, and R. Balasubramanian. Computing hierarchical curve-skeletons of 3D objects. *Visual Comput.*, 21(11):945–955, 2005.

[59] N. Cornea, D. Silver, and P. Min. Curve-skeleton properties, applications, and algorithms. *IEEE TVCG*, 13(3):87–95, 2007.

[60] C. Cortes and V. Vapnik. Support-vector networks. *Mach Learn*, 20(3):273–297, 1995.

[61] L. Costa and R. Cesar. *Shape analysis and classification*. CRC Press, 2000.

[62] M. Couprie, D. Coeurjolly, and R. Zrour. Discrete bisector function and Euclidean skeleton in 2D and 3D. *Image Vision Comput.*, 25(10):1543–1556, 2007.

[63] M. Couprie, F. N. Bezerra, and G. Bertrand. Topological operators for grayscale image processing. *J. Electronic Imag.*, 10(4):1003–1015, 2001.

[64] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. Imag. Proc.*, 13(9):1200–1212, 2004.

[65] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. ACM SIGGRAPH*, pages 303–312, 1996.

[66] J. Damon. Global medial structure of regions in $\mathbb{R}^3$. *Geometry and Topology*, 10:2385–2429, 2006.

[67] J. Davis, S. Marschner, M. Garr, and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. In *Proc. 3D Data Processing Visualization and Transmission*, pages 428–438, 2002.

[68] T. Delame, C. Roudet, and D. Faudot. From a medial surface to a mesh. *CGF*, 31(5):1637–1646, August 2012.

[69] V. Devita, S. Hellman, and S. Rosenberg. *Cancer: Principles & Practice of Oncology*. Lippincott Williams & Wilkins, 2001.

[70] T. Dey and J. Sun. Defining and computing curve-skeletons with medial geodesic function. In *Proc. SGP*, pages 143–152, 2006.

[71] T. Dey and W. Zhao. Approximating the medial axis from the Voronoi diagram with a convergence guarantee. *Algorithmica*, 38:179–200, 2003.

[72] T. Dey and W. Zhao. Approximate medial axis as a Voronoi subcomplex. *Comp. Aided Design*, 36(2):195–202, 2004.

[73] I. Drori, D. Cohen-Or, and H. Yeshurun. Fragment-based image completion. *ACM TOG*, 22(3):303–312, 2003.

[74] X. Du, D. Cho, and T. Bui. Image segmentation and inpainting using hierarchical level set and texture mapping. *Signal Processing*, pages 852–863, March 2011. ISSN 01651684.

[75] L. Duan, H. Q. Yang, and D. Samaras. Shape reconstruction from 3d and 2d data using pde-based deformable surfaces. *Proc. Eur. Conf. Comput. Vis.*, pages 238–251, 2004.

[76] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proc. ICCV*, volume 2, pages 10–33, 1999.

[77] M. Elad, J. L. Starck, P. Querre, and D. L. Donoho. Simultaneous cartoon and texture image inpainting using morphological component analysis (mca). *Applied and Computational Harmonic Analysis*, 19(3):340–358, 2005.

[78] S. Esedoglu and J. Shen. Digital image inpainting by the Mumford-Shah-Euler image model. *European J. Appl. Math.*, 13:353–370, 2002.

[79] A. Falcão, L. Costa, and B. da Cunha. Multiscale skeletons by image foresting transform and its applications to neuromorphometry. *Pattern Recognition*, 35 (7):1571–1582, 2002.

[80] A. X. Falcão, J. Stolfi, and R. A. Lotufo. The image foresting transform: Theory, algorithms, and applications. *IEEE TPAMI*, 26(1):19–29, 2004.

[81] Z. Farbman, G. Hoffer, Y. Lipman, D. CohenOr, and D. Lischinski. Coordinates for instant image cloning. In *Proc. ACM SIGGRAPH*, pages 342–450, 2009.

[82] O. Faugeras and R. Keriven. Variational principles, surface evolution, PDE's, level set methods, and the stereo problem. *IEEE Trans. Image Process*, 7(3): 336–344, 1998.

[83] FGB. Federal Goverment of Brazil, Justice Ministry - Missing Persons, 2012. URL http://www.desaparecidos.mj.gov.br.

[84] M. Fiorese, E. Peserico, and A. Silletti. VirtualShave: automated hair removal from digital dermatoscopic images. In *Proc. EMBS*, pages 5145–5148, 2011.

[85] E. Flores and J. Scharcanski. Segmentation of pigmented melanocytic skin lesions based on learned dictionaries and normalized graph cuts. In *Graphics, Patterns and Images (SIBGRAPI), 2014 27th SIBGRAPI Conference on*, pages 33–40, Aug 2014. DOI 10.1109/SIBGRAPI.2014.42.

[86] M. Foskey, M. Lin, and D. Manocha. Efficient computation of a simplified medial axis. In *Proc. Shape Modeling*, pages 135–142, 2003.

[87] FotoFinder. Handyscope mobile dermatoscope specifications, 2015. www.handyscope.net.

[88] P. Frey. YAMS: a fully automatic adaptive isotropic surface remeshing procedure. tech. rep. 0252, INRIA, Nov. 2001. http://www.ann.jussieu.fr/~frey.

[89] R. Friedman, D. Rigel, and A. Kopf. Early detection of malignant melanoma: The role of physician examination and self-examination of the skin. *CA Cancer J Clin*, 35(3):130–151, 1985.

[90] N. Gagvani and D. Silver. Parameter-controlled volume thinning. *Graphical Models and Image Processing*, 61(3):149–164, 1999.

[91] Y. Ge and J. Fitzpatrick. On the generation of skeletons from discrete Euclidean distance maps. *IEEE TPAMI*, 18:1055–1066, 1996. ISSN 0162-8828.

[92] General Electrics. Vivid software platform, 2014. www3.gehealthcare.com/products/vivid.

[93] P. Giblin and B. Kimia. A formal classification of 3D medial axis points and their local geometry. *IEEE TPAMI*, 26(2):238–251, 2004.

[94] J. Giesen, B. Miklos, M. Pauly, and C. Wormser. The scale axis transform. In *Proc. Annual Symp. Comp. Geom.*, pages 106–115, 2009.

[95] B. Gunturk and X. Li. *Image Restoration: Fundamentals and Advances*. Digital Imaging and Computer Vision. CRC Press, 2012.

[96] F. Han and S. C. Zhu. Bayesian reconstruction of 3d shapes and scenes from a single image. In *Proc. First IEEE Intl. Workshop Higher-Level knowl. 3 D Model. Motion Anal. (HLK'03)*, pages 1–12, 2003.

[97] X. Han, C. Xu, and J. L. Prince. A topology preserving level set method for geometric deformable models. *IEEE TPAMI*, 25(6):755–768, 2003.

[98] M. Hassouna and A. Farag. Variational curve skeletons using gradient vector flow. *IEEE TPAMI*, 31(12):2257–2274, 2009.

[99] J. Hays and A. Efros. Scene completion using milions of photographs. *ACM Transactions on Graphics (SIGGRAPH 2007)*, pages 1 – 7, August 2007.

[100] H. Heijmans and J. Roerdink. *Mathematical Morphology and its Applications to Image and Signal Processing*, volume 12 of *Computational Imaging and Vision*. Kluwer, 1998.

[101] H. Heijmans. *Morphological Image Operators*. Harcourt, Brace and Company, 1994.

[102] W. Hesselink and J. Roerdink. Euclidean skeletons of digital image and volume data in linear time by the integer medial axis transform. *IEEE TPAMI*, 30(12): 2204–2217, 2008.

[103] E. Hjelmas and B. Low. Face detection: A survey. *CVIU*, 83(3):236–274, 2001.

[104] A. Huang, S. Kwan, W. Chang, M. Liu, M. Chi, and G. Chen. A robust hair segmentation and removal approach for clinical images of skin lesions. In *Proc. EMBS*, pages 3315–3318, 2013.

[105] H. Huang, S. Wu, D. Cohen-Or, M. Gong, H. Zhang, G. Li, and B. Chen. l_1 medial skeleton of point cloud. *ACM TOG*, 32(4):1–8, 2013.

[106] ISO. Final draft international standard of biometric data interchange formats (part 5, face image data, ISO–19794-5 FDIS), Nov. 2004.

[107] M. Iwanowski and P. Soille. Fast algorithm for order independent binary homotopic thinning. In *Proc. ICANNGA*, pages 606–615. Springer, 2007.

[108] H. Iyatomi, H. Oka, G. Celebi, M. Hashimoto, M. Hagiwara, M. Tanaka, and K. Ogawa. An improved internet-based melanoma screening system with dermatologist-like tumor area extraction algorithm. *Comp. Med. Imag. Graph.*, 32(7):566–579, 2008.

[109] A. Jalba, J. Kustra, and A. Telea. Surface and curve skeletonization of large 3D models on the GPU. *IEEE TPAMI*, 35(6):1495–1508, 2013.

[110] A. Jalba, A. Sobiecki, and A. Telea. An unified multiscale framework for planar, surface, and curve skeletonization. *IEEE TPAMI*, 2015. DOI 10.1109/TPAMI.2015.2414420.

[111] M. Janaszewski, M. Couprie, and L. Babout. Hole filling in 3d volumetric objects. *Pattern Recognition*, 43(10):3548–3559, 2010.

[112] S. Jeschke, D. Cline, and P. Wonka. A GPU laplacian solver for diffusion curves and poisson image editing. In *Proc. ACM SIGGRAPH Asia*, pages 1–8, 2009.

[113] J. Jia and C. Tang. Image repairing: Robust image synthesis by adaptive *n*D tensor voting. In *Proc. IEEE CVPR*, pages 643–650, 2003.

[114] H. Jin, S. Soatto, and A. J. Yezzi. Multi-view stereo beyond Lambert. In *Proc. IEEE CVPR*, pages 171–178, 2003.

[115] A. V. J.O.S. Junior, O. Bellon, and L. Silva. 3d reconstruction of cultural heritages: Challenges and advances on precise mesh integration. *Computer Vision and Image Understanding*, 116(12):1195–1207, 2012.

[116] N. Joshi, W. Matusik, E. Adelson, M. Csail, and D. Kriegman. Personal photo enhancement using example images. In *Proc. ACM SIGGRAPH*, pages 1–15, 2010.

[117] T. Ju, M. Baker, and W. Chiu. Computing a family of skeletons of volumetric models for shape description. *Comput. Aided Design*, 39(5):352–360, 2007.

[118] J.Wang, K.Lu, D. Pan, N. He, and B. Bao. Robust object removal with an exemplar-based image inpainting approach. *NeuroComputing*, 123:150–155, 2014. DOI 10.1016/j.neucom.2013.06.022.

[119] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, pages 321–331, 1987.

[120] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM TOG*, 22(3):954–961, 2003.

[121] N. Kawai, T. Sato, and N. Yokoya. Surface completion by minimizing energy based on similarity of shape. In *Proc. IEEE Int. Conf. on Image Processing (ICIP2008)*, pages 1532–1535, 2008.

[122] N. Kawai, T. Sato, and N. Yokoya. Efficient surface completion using principal curvature and its evaluation. In *IEEE Int. Conf. on Image Processing (ICIP2009)*, pages 521–524, 2009.

[123] N. Kawai, A. Zakhor, T. Sato, and N. Yokoya. Surface completion of shape and texture based on energy minimization. In *Proc. Int. Conf. on Image Processing (ICIP)*, pages 913–916, 2011.

[124] K. Kiani and A. Sharafat. E-shaver: An improved dullrazor for digitally removing dark and light-colored hairs in dermoscopic images. *Computers in Biology and Medicine*, 41:139–145, 2011.

[125] R. Kimmel, D. Shaked, N. Kiryati, and A. Bruckstein. Skeletonization via Distance Maps and Level Sets. *CVIU*, 62(3):382–391, 1995.

[126] J. Koehoorn, A. Sobiecki, D. Boda, A. Diaconeasa, A. Jalba, and A. Telea. Digital hair removal source code, 2014. www.cs.rug.nl/svcg/Shapes/HairRemoval.

[127] J. Koehoorn, A. Sobiecki, D. Boda, A. Diaconeasa, S. Doshi, S. Paisey, A. Jalba, and A. Telea. Automated digital hair removal by threshold decomposition and morphological analysis. In *Mathematical Morphology and Its Applications to Signal and Image Processing (Proc. ISMM)*, pages 15–26. Springer LNCS 9082, 2015.

[128] J. Koehoorn, A. Sobiecki, P. Rauber, A.Jalba, and A. Telea. Efficient and effective automated digital hair removal from dermoscopy images. *submitted*, 2015.

[129] T. Kohonen. Learning vector quantization. In *Self-Organizing Maps*, pages 203–217. Springer, 1997.

[130] N. Komodakis and G. Tziritas. Image completion using efficient belief propagation via priority scheduling and dynamic pruning. *IEEE Trans. Imag. Process.*, 16:26–49, 2007.

[131] K. N. Konemann. *The restoration of paintings*. Konemann Inc., 1999.

[132] K. Korotkov and R. Garcia. Methodological review: Computerized analysis of pigmented skin lesions: A review. *Artif. Intell. Med.*, 56(2):69–90, 2012.

[133] J. Kustra, A. Jalba, and A. Telea. Probabilistic view-based 3D curve skeleton computation on the GPU. In *Proc. VISAPP*, volume 2, pages 237–246, 2013.

[134] J. Kustra, A. Jalba, and A. Telea. Robust segmentation of multiple intersecting manifolds from unoriented noisy point clouds. *CGF*, 33(1):73–87, 2014.

[135] J. Kustra, A. Jalba, and A. Telea. Computing refined skeletal features from medial point clouds. *Patt Recog Lett*, 2015. DOI dx.doi.org/10.1016/j.patrec.2015.05.007.

[136] C. Lantuéjoul. *La squelettisation et son application aux mesures topologiques de mosaiques polycristallines*. PhD thesis, School of Mines, Paris, 1979.

[137] H. Y. Lee, H. K. Lee, T. Kim, and W. Park. Towards knowledge-based extraction of roads from 1m-resolution satellite images. In *Proc. SSIAI*, pages 171–178, 2000.

[138] T. Lee, V. Ng, R. Gallagher, A. Coldman, and D. McLean. Dullrazor®: A software approach to hair removal from images. *Comput. Biol. Med.*, 27(6):533–543, 1997.

[139] M. Lentine, J. Gretarsson, and R. Fedkiw. An unconditionally stable fully conservative semi-Lagrangian method. *J. Comp. Phys.*, 230(8):2857–2879, 2011.

[140] S. Lessard, C. Lau, D. Roy, G. Soulez, and J. de Guise. Wires segmentation in fluoroscopic images during cerebral aneurysm endovascular intervention. In *Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE International Symposium on*, pages 193–196, May 2008.

[141] F. Leymarie and B. Kimia. The medial scaffold of 3D unorganized point clouds. *IEEE TVCG*, 29(2):313–330, 2007.

[142] F. Leymarie and M. Levine. Simulating the grassfire transform using an active contour model. *IEEE TPAMI*, 14(1):56–75, jan 1992.

[143] C. Li, C. Xu, C. Gui, and M. D. Fox. Distance regularized level set evolution and its application to image segmentation. *IEEE TPAMI*, 19(12):3243–3254, 2010.

[144] H. Li, S. Wang, W. Zhang, and M. Wu. Image inpainting based on scene transform and color transfer. *Pattern. Recogn. Lett.*, 31(7):582–592, 2010.

[145] S. Li and M. Zhao. Image inpainting with salient structure completion and texture propagation. *Pattern Recognittion Letters*, pages 1256–1266, 2011.

[146] X. Li, T. Woon, T. Tan, and Z. Huang. Decomposing polygon meshes for interactive applications. In *Proc. I3D Symp.*, pages 35–42, 2001.

[147] Z. Li, X. M. Wu, and S. F. Chang. Segmentation using superpixels: A bipartite graph partitioning approach. In *Proc. CVPR*, pages 789–796, 2012.

[148] P. Liepa. Filling holes in meshes. In *Proc. SGP*, pages 200–205. Eurographics, 2003.

[149] L. Liu, E. Chambers, D. Letscher, and T. Ju. A simple and robust thinning algorithm on cell complexes. *CGF*, 29(7):2253–2260, 2010.

[150] M. Livesu, F. Guggeri, and R. Scateni. Reconstructing the curve-skeletons of 3D shapes using the visual hull. *IEEE TVCG*, 18(11):1891–1901, 2012.

[151] J. Ma, S. Bae, and S. Choi. 3D medial axis point approximation using nearest neighbors and the normal field. *Vis. Comput.*, 28(1):7–19, 2012.

[152] D. Macrini, K. Siddiqi, and S. Dickinson. From skeletons to bone graphs: Medial abstraction for object recognition. In *Proc. IEEE CVPR*, pages 1–8, 2008.

[153] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE Trans. Imag. Process.*, 17(1):53–63, 2008.

[154] J. Mairal, G. Sapiro, and M. Elad. Learning multiscale sparse representations for image and video restoration. *SIAM Multiscale Modeling and Simulation*, 7 (1):214–241, 2008.

[155] G. Malandain and S. Fernandez-Vidal. Euclidean skeletons. *Image and Vision Computing*, 16(5):317–327, 1998.

[156] P. Maragos and M. A. Butt. Curve evolution, differential morphology, and distance transforms applied to multiscale and eikonal problems. *Fundam. Inf.*, 41 (1,2):91–129, 2000.

[157] S. Masnou and J. M. Morel. Level lines based disocclusion. In *Proc. Intl. Conf. on Image Processing*, 1998.

[158] F. Meyer. Skeletons and perceptual graphs. *Signal Processing*, 16(4):335–363, 1989.

[159] B. Miklos, J. Giesen, and M. Pauly. Discrete scale axis representations for 3D geometry. In *Proc. ACM SIGGRAPH*, pages 394–493, 2010.

[160] M. K. M.J. Lyons, S. Akamatsu and J. Gyoba. Coding facial expressions with gabor wavelets. In *Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 200–204. IEEE, 1998.

[161] M. Mortara, G. Patanet, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Plumber: A method for multiscale decomposition of 3D shapes into tubular primitives and bodies. In *Proc. ACM SMA*, pages 339–344, 2004.

[162] MPO. Missing People from United Kingdom, 2012. URL http://www.missingpeople.org.uk.

[163] J. Mullikin and P. Verbeek. Surface area estimation of digitized planes. *Bioimaging*, 1(1):6–16, 1993.

[164] L. Neumann, B. Csébfalvi, A. König, and E. Gröller. Gradient estimation in volume data using 4D linear regression. *CGF*, 19(3):351–358, 2000.

[165] N. Nguyen, T. Lee, and M. Atkins. Segmentation of light and dark hair in dermoscopic images: a hybrid approach using a universal kernel. In *Proc. SPIE Med. Imaging*, pages 1–8, 2010.

[166] F. Nooruddin and G. Turk. Simplification and repair of polygonal models using volumetric techniques. *IEEE TVCG*, 9(2):191–205, 2003. see also www.cs.princeton.edu/~min/binvox.

[167] J. M. Ogden, E. H. Adelson, J. R. Bergen, and P. J. Burt. Pyramid-based computer graphics. *RCA Engineer*, 30(5):4–15, 1985.

[168] R. L. Ogniewicz and O. Kubler. Hierarchic Voronoi skeletons. *Pattern Recognition*, (28):343–359, 1995.

[169] M. M. Oliveira, B. Bowen, R. Mckenna, and Y. Chang. Fast digital image inpainting. In *Proc. VIIP*, pages 261–266, 2001.

[170] OpenCV. Open source library for computer vision, 2015. http://opencv.org.

[171] K. Pálagyi and A. Kuba. Directional 3D thinning using 8 subiterations. In *Proc. DGCI*, volume 1568, pages 325–336. Springer LNCS, 1999.

[172] A. Parolin, E. Herzer, and C. Jung. Semi-automated diagnosis of melanoma through the analysis of dermatological images. In *Proc. SIBGRAPI*, pages 71–78. IEEE Press, 2010.

[173] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. In *Proc. ACM SIGGRAPH*, pages 313–318, 2003.

[174] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE TPAMI*, 12(7):629–639, 1990.

[175] P. Peter and M. Breuß. Refined homotopic thinning algorithms and quality measures for skeletonisation methods. In *Innovations for Shape Analysis Mathematics and Visualization*, pages 77–92. Springer, 2013.

[176] S. Pizer, K. Siddiqi, G. Szekely, J. Damon, and S. Zucker. Multiscale medial loci and their properties. *IJCV*, 55(2-3):155–179, 2003.

[177] S. R. P.J. Phillips, M. Hyeonjoon and P. Rauss. The FERET evaluation methodology for face-recognition algorithms. *IEEE TPAMI*, 22(10):1090–1104, 2000.

[178] S. Prohaska and H. C. Hege. Fast visualization of plane-like structures in voxel data. In *Proc. IEEE Visualization*, pages 29–36, 2002.

[179] C. Pudney. Distance-ordered homotopic thinning: A skeletonization algorithm for 3D digital images. *CVIU*, 72(3):404–413, 1998.

[180] A. Rahimi. Fast connected components on images. alumni.media.mit.edu/~rahimi/connected, 2014.

[181] P. Rauber, R. da Silva, S. Feringa, M. Celebi, A. Falcao, and A. Telea. Interactive image feature selection aided by dimensionality reduction. In *Proc. EuroVA*, pages 322–328, 2015.

[182] P. Rauber, A. Falcao, T. Spina, and P. De Rezende. Interactive segmentation by image foresting transform on superpixel graphs. In *Graphics, Patterns and Images (SIBGRAPI), 2013 26th SIBGRAPI - Conference on*, pages 131–138, Aug 2013. DOI 10.1109/SIBGRAPI.2013.27.

[183] E. Remya and E. Thiel. Exact medial axis with Euclidean distance. *Image and Vision Computing*, (23):167–175, 2005.

[184] D. Reniers and A. Telea. Patch-type segmentation of voxel shapes using simplified surface skeletons. *CGF*, 27(7):1954–1962, 2008.

[185] D. Reniers and A. Telea. Part-type segmentation of articulated voxel-shapes using the junction rule. *CGF*, 27(7):1837–1844, 2008.

[186] D. Reniers, A. Jalba, and A. Telea. Robust classification and analysis of anatomical surfaces using 3D skeletons. In *Proc. VCBM*, pages 61–68, 2008.

[187] D. Reniers, J. J. van Wijk, and A. Telea. Computing multiscale skeletons of genus 0 objects using a global importance measure. *IEEE TVCG*, 14(2):355–368, 2008.

[188] L. Rossi and A. Torsello. An adaptive hierarchical approach to the extraction of high resolution medial surfaces. In *Proc. 3DIMPVT*, pages 371–378, 2012.

[189] M. Rumpf and A. Telea. A continuous skeletonization method based on level sets. In *Proc. VisSym*, pages 151–158, 2002.

[190] T. Ružić, B. Cornelis, L. Platiša, A. Pižurica, A. Dooms, W. Philips, M. Martens, M. D. Mey, and I. Daubechies. Virtual restoration of the Ghent altarpiece using crack detection and inpainting. In *Advanced Concepts for Intelligent Vision Systems*, pages 417–428. Springer LNCS 6915, 2011.

[191] P. K. Saha, G. Borgefors, and G. S. di Baja. A survey on skeletonization algorithms and their applications. In *Pattern Recognition Letters*. Elsevier, 2015.

[192] P. Saugeon, J. Guillod, and J. Thiran. Towards a computer-aided diagnosis system for pigmented skin lesions. *Comput. Med. Imag. Grap.*, 27:65–78, 2003.

[193] M. Schaap, C. Metz, T. van Walsum, A. van der Giessen, A. Weustinck, N. Mollet, C. Bauer, H. Bogunovic, C. Castro, X. Deng, E. Dikici, T. O'Donnell, M. Frenay, O. Friman, M. Hoyos, P. Kitslaar, A. Szymczak, H. Tek, C. Wang, S. Warfield, S. Zambal, Y. Zhang, G. Kresting, and W. Niessen. Standardized evaluation methodology and reference database for evaluating coronary artery centerline extraction algorithms. *Med Image Anal*, 13(5):701–714, 2009.

[194] J. Scharkanski and M. Celebi. *Computer Vision Techniques for the Diagnosis of Skin Cancer*. Springer, 2014.

[195] T. Sebastian, P. Klein, and B. Kimia. Recognition of shapes by editing their shock graphs. *IEEE TPAMI*, 26(5):550–571, 2004.

[196] H. Sellahewa and S. Jassim. Image-quality-based adaptive face recognition. *IEEE TIM*, 59(4):805–813, 2010.

[197] J. Serra. *Image analysis and mathematical morphology*. Acad. Press, 1982.

[198] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge Univ. Press, 2nd edition, 1999.

[199] D. Shaked and A. Bruckstein. Pruning medial axes. *CVIU*, 69(2):156–169, 1998.

[200] W. SHAO and Z. WEI. Edge-and-corner preserving regularization for image interpolation and reconstruction. *Image and Vision Computing*, pages 1591–1606, 2008.

[201] A. Sharf, T. Lewiner, A. Shamir, and L. Kobbelt. On-the-fly curve-skeleton computation for 3d shapes. *CGF*, 26(3):323–328, 2007.

[202] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE TPAMI*, 22(8):888–905, 2000.

[203] K. Siddiqi and S. Pizer. *Medial Representations: Mathematics, Algorithms and Applications*. Springer, 2009.

[204] K. Siddiqi, S. Bouix, A. Tannenbaum, and S. Zucker. Hamilton-Jacobi skeletons. *IJCV*, 48(3):215–231, 2002.

[205] A. Sobiecki, A. Jalba, and A. Telea. Qualitative and quantitative comparison of curve and surface skeletons – a state of the art review. In *ICTOpen, Eindhoven, The Netherlands*, 2013.

[206] A. Sobiecki, A. Telea, G. Giraldi, L. Neves, and C. Thomaz. Low-cost automatic inpainting for artifact suppression in facial images. In *Proc. 8^{th} International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*, pages 324–335, 2013.

[207] A. Sobiecki, H. Yasan, A. Jalba, and A. Telea. Qualitative comparison of contraction-based curve skeletonization methods. In *Mathematical Morphology and Its Applications to Signal and Image Processing (Proc. ISMM)*, pages 425–439. Springer LNCS 7883, 2013.

[208] A. Sobiecki, A. Jalba, D. Boda, A. Diaconeasa, and A. Telea. Gap-sensitive segmentation and restoration of digital images. In *Proc. Computer Graphics and Visual Computing (CGVC)*, pages 136–144. Eurographics, 2014.

[209] A. Sobiecki, A. Jalba, and A. Telea. Skeletonization benchmarking resources, 2014. www.cs.rug.nl/svcg/Shapes/SkelBenchmark.

[210] A. Sobiecki, A. Jalba, and A.C.Telea. Comparison of curve and surface skeletonization methods for voxel shapes. *Pattern Recogn. Lett.*, 47:147–156, October 2014.

[211] A. Sobiecki, A. Jalba, and A. Telea. Automatic gap restoration in 2D and 3D digital images. In *ICTOpen, Amersfoort, The Netherlands*, 2015.

[212] A. Sobiecki, J. Koehoorn, D. Boda, C. Solovan, A. Diaconeasa, A. Jalba, and A. Telea. A new efficient method for digital hair removal by dense threshold analysis. In *Proc. 4^{th} World Congress of Dermoscopy (IDS), Vienna, Austria*, 2015.

[213] M. Spiegel and L. Stephens. *Statistics*. Schaum's Outlines, 2008.

[214] S. Stolpner, S. Whitesides, and K. Siddiqi. Sampled medial loci and boundary differential geometry. In *Proc. IEEE 3DIM*, pages 87–95, 2009.

[215] S. Stolpner, S. Whitesides, and K. Siddiqi. Sampled medial loci for 3D shape representation. *CVIU*, 115(5):695–706, 2011.

[216] R. Strzodka and A. Telea. Generalized distance transforms and skeletons in graphics hardware. In *Proc. VisSym*, pages 221–230, 2004.

[217] A. Sud, M. Foskey, and D. Manocha. Homotopy-preserving medial axis simplification. In *Proc. SPM*, pages 103–110, 2005.

[218] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. In *Proc. SMI*, pages 130–137, 2003.

[219] S. Svensson. Reversible surface skeletons of 3D objects by iterative thinning of distance transforms. In *Proc. DGCI*, pages 400–411. Springer, 2001.

[220] A. Tagliasacchi, H. Zhang, and D. Cohen-Or. Curve skeleton extraction from incomplete point clouds. *ACM TOG*, 28(3):71:1–71:9, 2009.

[221] A. Tagliasacchi, I. Alhashim, M. Olson, and H. Zhang. Mean curvature skeletons. *CGF*, 31(5):1735–1744, 2012.

[222] Z. Tauber, Z. N. Li, and M. Drew. Review and preview: Disocclusion by inpainting for image-based rendering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(4):527–540, 2007.

[223] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proc. ICCV*, pages 902–907, 1995.

[224] H. Tek and B. Kimia. Boundary smoothing via symmetry transforms. *J. Math. Imag. Vis.*, 14:211–223, 2001.

[225] A. Telea. Feature preserving smoothing of shapes using saliency skeletons. In *Visualization in Medicine and Life Sciences*, pages 153–170. Springer, 2012.

[226] A. Telea. *Data Visualization – Principles and practice*. CRC Press, 2014. 2^{nd} edition.

[227] A. Telea and A. Jalba. Computing curve skeletons from medial surfaces of 3D shapes. In *Proc. Theory & Practice of Computer Graphics (TPCG)*, pages 137–145, 2012.

[228] A. Telea and J. J. van Wijk. An augmented fast marching method for computing skeletons and centerlines. In *Proc. VisSym*, 2002.

[229] A. C. Telea. An image inpainting technique based on the fast marching method. *J. Graphics. Tools*, 9(1):23–34, 2004.

[230] N. Thakare and V. Thakare. Biometrics standards and face image format for data interchange - a review. *Int. J. of Advances in Engineering and Technology*, 2(1):385–392, 2012.

[231] H. Thomas. Jordan's proof of the Jordan curve theorem. *Studies in Logic, Grammar and Rhetoric*, 10(23):567–582, 2007.

[232] C. Thomaz and G. Giraldi. A new ranking method for principal components analysis and its application to face image analysis. *Image Vision Comput.*, 28 (6):902–913, 2010.

[233] D. Tschumperl'e. Fast anisotropic smoothing of multi-valued images using curvature-preserving PDE's. *IJCV*, 68(1):65–82, 2006.

[234] Univ. of Pisa. *MeshLab* mesh processing toolkit, 2012. http://meshlab.sourceforge.net.

[235] J. Verdera, V. Caselles, M. Bertalmio, and G. Sapiro. Inpainting surface holes. In *Proc. ICIP*, pages 342–332, 2003.

[236] M. Wan, F. Dachille, and A. Kaufman. Distance-field based skeletons for virtual navigation. In *Proc. IEEE Visualization*, pages 239–246, 2001.

[237] J. Wang and M. Oliveira. Filling holes on locally smooth surfaces reconstructed from point clouds. *Image and Vision Computing*, 25(1):103–113, 2007.

[238] Z. Wang, L. Lu, and A. Bovik. Video quality assessment based on structural distortion measurement. *Sig. Proc.: Image Comm.*, 19(2):121–132, 2004.

[239] O. Whyte, J. Sivic, and A. Zisserman. Get out of my picture! internet based inpainting. *Proceedings of the 20th British Machine Vision Conference, London*, pages 1–11, September 2009.

[240] P. Wighton, T. Lee, and M. Atkins. Dermoscopic hair disocclusion using inpainting. In *Proc. SPIE Med. Imaging*, pages 144–151, 2008.

[241] F. Xie, S. Qin, Z. Jiang, and R. Meng. PDE-based unsupervised repair of hair-occluded information in dermoscopy images of melanoma. *Comp. Med. Imag. Graph.*, 33:275–282, 2009.

[242] H. Yasan. Contraction-based curve skeletonization of 3D meshes, 2012. MSc thesis, Dept. of Computer Science, TU Eindhoven, the Netherlands, http://repository.tue.nl/741246.

[243] A. Zamani, M. Awang, N. Omar, and S. Nazeer. Image quality assessments and restoration for face detection and recognition system images. In *Proc. AMS*, pages 505–510, 2008.

[244] W. Zhao and S. Gao, Lin. A robust hole-filling algorithm for triangular meshes. In *Visual Comput, Springer-Verlag*, 2007.

[245] W. Zhao, R. Chellappa, P. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys*, 35(4):399–458, 2003.

[246] M. van der Zwan, Y. Meiburg, and A. Telea. A dense medial descriptor for image analysis. In *Proc. IEEE VISAPP*, pages 133–140, 2013.

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor, professor Telea, for taking me as his PhD student. Thank you very much for being an enthusiastic researcher, and a good supervisor who has provided me with the needed orientation. I am glad to say that I learned a lot from him concerning both science and life in general. I appreciated much his explanatory abilities, and the huge amount of scientific topics that he deeply understands.

I would also like to thank my PhD co-supervisor Andrei Jalba and my colleague Jasper van der Gronde – thanks for the extra help, orientation, and discussions. Thanks go also to professor Reordink, the SVCG group leader, for managing the excellent context of the SVCG group. To the members of my doctoral committee – professors Walter Falcão, and Petkov – thank you for your comments and suggestions and time taken to read my thesis. They all helped me to improve my final manuscript. Thanks go also to professors Thomaz, Neves, and Giraldi, who have helped me with my preparations before starting my PhD, and supported my application.

Furhtermore, I would like to thank the Brazilian program Science Without Borders, together with CNPq, for the PhD scholarship they offered me; and the University of Groningen, for supporting me with all high-quality facilities I needed for my research.

Thanks to all our institute's 'secretaries, especially Esmee Elshof, Ineke Scheelhas, Desiree Hansen, Janieta de Jong-Schlukebir, and their colleagues, for their administrative support and help. Thanks go to all researchers I have collaborated with during my work, for sharing their ideas, knowledge, and algorithms with me. To all MSc, PhD, and professors from the University of Groningen and UMCG I have worked with, thank you for your friendship.

To all my friends, especially those that I have made here in Groningen, thanks for the company, for the parties and for the drinks enjoyed together. To all sports and student associations, thanks for so many nice trips and activities, I enjoyed them a lot. The student and sport associations AEGEE, MayDay, Aclo, Tjas, Surface, and ESN deserve a definite mention, I had a great time with all of them. These were the places where I could recover my mind and energy to pursue my research goals.

My sweet girlfriend, Lenja Vroom, she has appeared in the last months of my PhD journey, thanks for making my life more joyful. Thanks to all my family and relatives for always being great with me. I would like to mention, especially, my dear mother Everilda S. Sobiecki, my awesome brother Alexandre Sobiecki, my awesome sister Madalena Sobiecki; my great father Lino Sobiecki, who unfortunately passed away, but know all about my plans for the future; and my grandmother Jadviga O. Sobiecki. My deepest appreciation is for them.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Final Version as of January 26, 2016 (`classicthesis`).